Topological tools for discrete shape analysis

John Chaussard

UNIVERSITÉ PARIS-EST

PARIS-EST UNIVERSITY
DOCTORAL SCHOOL MSTIC

A thesis submitted in partial fulfilment for the degree of **Doctor of Philosophy** in Computer Science

presented by **John Chaussard**

advised by Michel Couprie

Topological tools for discrete shape analysis

September 2010[1]

Committee in charge:

J.O. Lachaud *(reviewer)*
C. Ronse *(reviewer)*
D. Attali
D. Bernard
G. Bertrand
M. Couprie
E. Thiel

1 latest revision:March 1, 2011

ABSTRACT

These last years, the domain of image analysis has drastically evolved. The progresses in technology brought higher resolution images and allowed to obtain three (even, in some cases, four) dimensional images. These progresses came with an important increase of the amount of information to analyse, and only tools capable of automatically extracting important characteristics from images can manage this flow of data. Digital topology offers a set of tools adapted to image analysis, especially the skeletonization process (also called homotopic thinning) which can simplify input data while keeping specific information untouched. In this thesis, we focus on how digital topology can help material image analysis.

In the first part, we study fluid flow simulation. In the images used for such simulations, it is necessary to remove small parts of the material not connected with the rest of the object (the grains). However, in some cases, the images are considered to be embedded in a toric space, where detection of grains can be hard. Thanks to the characterization of the fundamental group of the torus, we propose an original algorithm for detecting and removing these grains.

The rest of this thesis covers the field of skeletonization methods. The goal of a skeletonization process is to remove unnecessary information from an input, and provide a simplified object, called the skeleton, having the same characteristics as the original data. It is then possible to perform some computations on the skeleton and generalise their results to the original data. In the second part of this thesis, we propose some new tools for preserving, during skeletonization, important geometrical features of the original data, and obtain a skeleton recalling the visual aspect of the input. This method, which works in all dimensions, requires from the user a filtering parameter. We compare our method with other tools used for visual aspect preservation during homotopic thinning.

In the last part, we present the cubical complex framework, where objects are no more made only of voxels. We propose in this framework new skeletonization algorithms, some of them preserving automatically the visual aspect of the input during the thinning process (no filtering parameter from the user is required). These algorithms work in 2d and 3d, and we compare them to other thinning methods. We then show how a skeleton, in the cubical complexes framework, can be decomposed into basic parts, and we show some applications of these algorithms to material image analysis and medical image analysis.

KEYWORDS: TOPOLOGY, SKELETON, MEDIAL AXIS, VOXELS, CUBICAL COMPLEXES, FUNDAMENTAL GROUP, TORIC SPACE, LAMBDA-MEDIAL AXIS, PARALLEL SKELETONIZATION, SKELETON DECOMPOSITION

RÉSUMÉ

L'analyse d'images est devenue ces dernières années une discipline de plus en plus riche de l'informatique. L'accroissement de la taille des données à traiter, due à l'amélioration de la résolution des dispositifs de capture d'images ainsi qu'à l'utilisation de plus en plus fréquente de la troisième (voir même, dans certains cas, de la quatrième) dimension, nécessite des outils plus élaborés afin de pouvoir extraire des informations pertinentes de ces images, et simplifier le travail des chercheurs en aval. La topologie discrète propose un panel d'outils incontournables dans le traitement d'images, notamment grâce à l'outil du squelette, qui permet de simplifier des objets tout en conservant certaines informations intactes. Cette thèse étudie comment certains outils de la topologie discrète peuvent être utilisés pour le traitement d'images de matériaux.

Dans un premier temps, nous nous intéressons à la simulation d'écoulements de fluide. Dans les images où se déroulent ces simulations, il est nécessaire de retirer les morceaux du matériau qui ne sont pas connectés au reste du matériau (des graines). Dans certains cas, les images sont plongées dans un espace dit "torique", rendant la détection des graines difficile. Grâce à la caractérisation du groupe fondamental du tore, nous avons élaboré un algorithme permettant de détecter ces graines, afin de pouvoir les retirer.

La suite de la thèse est dédiée aux squelettes. Le squelette d'un objet peut être vu comme une simplification d'un objet, possédant certaines caractéristiques identiques à celles de l'objet original. Il est alors possible d'étudier un squelette et de généraliser certains résultats à l'objet entier. Dans la seconde partie de la thèse, nous proposons une nouvelle méthode pour conserver, dans un squelette, certaines caractéristiques géométriques de l'objet original (méthode nécessitant un paramètre de filtrage de la part de l'utilisateur) et obtenir ainsi un squelette possédant le même aspect que l'objet original. Notre méthode, qui fonctionne en toute dimension, est comparée à d'autres méthodes déjà existantes.

Enfin, la dernière partie propose de ne plus travailler avec des objets constitués de voxels, mais avec des objets constitués de complexes cubiques. Dans ce nouveau cadre, nous proposons de nouveaux algorithmes de squelettisation, dont certains permettent de conserver certaines caractéristiques géométriques de l'objet de départ dans le squelette, de façon automatique (aucun paramètre de filtrage ne doit être donné par l'utilisateur). Nos algorithmes, fonctionnant en 2d et en 3d, seront comparés à d'autres algorithmes de squelettisation. Nous montrerons ensuite comment un squelette, dans le cadre des complexes cubiques, peut être décomposé en différentes parties. Enfin, nous montrerons nos résultats sur différentes applications, allant de l'étude des matériaux à l'imagerie médicale.

MOTS CLEFS : TOPOLOGIE, SQUELETTE, AXE MÉDIAN, VOXELS, COMPLEXES CUBIQUES, GROUPE FONDAMENTAL, ESPACE TORIQUE, LAMBDA-MEDIAL AXIS, SQUELETTISATION PARALLÈLE, DÉCOMPOSITION DU SQUELETTE

ACKNOWLEDGEMENTS / REMERCIEMENTS

Je souhaiterais avant tout dire un très grand merci à Michel Couprie, mon directeur de thèse, qui m'a accompagné pendant toutes ces années. Il a toujours su montrer une grande patience face à ma grande inertie pour écrire des articles, et a toujours attentivement écouté mes différentes idées, dont certaines ne méritaient pas d'écoute attentive... Je souhaiterai aussi remercier Gilles Bertrand, directeur du laboratoire A3SI, qui m'a accompagné pendant ma thèse, m'a souvent conseillé sur différents choix, a toujours su écouter mes questions et y répondre malgré les grandes charges de travail qui lui incombaient, et qui fut mon principal fournisseur de Schoko-Bons.

Je souhaite remercier aussi les autres membres de mon jury de thèse, c'est à dire Jacques-Olivier Lachaud, Christian Ronse, Dominique Attali, Dominique Bernard et Edouard Thiel, qui ont accepté de consacrer du temps à la relecture de mon manuscrit, et qui ont aussi fait l'expérience de ma grande inertie à écrire cette thèse. Un second merci à Dominique Bernard, qui m'a souvent accueilli dans son laboratoire à Bordeaux, et m'a fait découvrir le merveilleux mur d'écran (dont je n'ai malheureusement pas pu emporter un petit morceau chez moi).

Ma thèse s'est déroulée au sein du laboratoire A3SI, où l'équipe qui m'a accueillie m'a très vite considéré comme l'un des ses membres, et m'a fait me sentir au bureau comme dans une seconde maison (simplement un peu moins chauffée en hiver). Merci pour toutes les discussions à la cantine, autour du café, ou dans les couloirs, et pour tous ces échanges d'idées si fructueuses (notamment l'épique discussion sur le problème des Sophie). Merci donc à Hugues, Yukiko, Denis, Xavier, Laurent, Michel, Gilles, Venceslas, Vincent, Dror, Marcin, les deux Jean, Mohammed et Thierry. Évidemment, je n'oublie pas mes compagnons de guerre, les autres doctorants (et bientôt docteurs) du laboratoire A3SI. Les moments partagés avec vous, dans le bureau ou hors du bureau (ou sur canal IRC) ont été réellement formidables et furent souvent l'occasion de joindre la procrastination à l'élévation de ma culture générale (surtout grâce à Quizyou). Un très grand merci à vous tous, Olena, Benjamin, Fabrice, Camille, Adrien, Nadine, Anthony. Un merci très spécial s'adresse à mes compagnons de bureau, qui m'ont accompagné (et supporté) pendant presque toutes ces quatre années: merci à Emilie qui est aujourd'hui docteur (avec mention spéciale du jury pour avoir tenu trois ans dans le même bureau que moi et Yohan), merci à Mohammed qui fit un passage court mais empli de rires au bureau, et Michal. J'adresse aussi un remercient très spécial à mon compagnon de thèse Yohan, qui m'accompagna dans le même bureau pendant presque quatre ans, travaillant sur un sujet assez différent du mien: sans lui, la thèse aurait peut-être été un tout petit peu plus productive (et j'aurai économisé de l'argent en n'achetant pas les tanks télécommandés), mais beaucoup plus austère... Merci aussi à David Menotti-Gomes, André Vital Saúde, et Nicolas Passat, qui sont partis depuis vers d'autres laboratoires, et avec qui j'ai pu échanger beaucoup d'idées lors de leur passage.

Je tiens aussi à remercier tous les acteurs de l'ombre qui m'ont aidé (plus sur le plan moral que sur les problèmes de fond de la thèse): ma famille. Merci à mes parents, mes grands-parents, mon frère et ma soeur (et leurs familles respectives) pour leur soutien quotidien et leur compréhension lors de mes longs silence, pendant les dîners, où je

réfléchissais à mes problèmes. Même si j'use de moins de mots pour vous remercier, je n'en suis pas moins reconnaissant pour tout le soutien que vous m'avez apporté. Un très très grand merci à ma femme, Sophia, qui pendant ces quatre années m'a accompagné au quotidien dans mes joies et mes peines, a su rendre les moments difficiles plus simples à passer, et a sacrifié ses vacances pour rester auprès de moi pendant l'été au travail. Un grand merci aussi à ma belle famille, en Corée du Sud, et spécialement ma belle-maman, qui m'a toujours encouragé (avec des mots que je ne comprenais pas forcément) pendant ces années.

Enfin, merci à tous mes autres amis (Eric, Olivier, Frédéric, Ehan, Damien, Nicolas, Emilie, Benoît, Valérie, Christophe, Angeline, Stéphane) qui, autour d'un verre, d'une pizza ou d'une manette de jeu, m'ont souvent écouté expliquer mes idées et mes avancées, tout en cachant leur faible intérêt pour mon discours.

De façon plus générale, merci à toutes les personnes de mon entourage et toutes les personnes rencontrées pendant cette thèse, "avec qui mes rapports furent aussi divers qu'enrichissants".

PUBLICATIONS

Some of the work explained in this thesis was already presented in the following publications:

CONFERENCES

- [CBC08] *Characterizing and detecting toric loops in n-dimensional discrete toric spaces*, by John Chaussard, Michel Couprie and Gilles Bertrand, in DGCI 2008, pages 129-140.
- [CCT09] *A discrete lambda-medial axis*, by John Chaussard, Michel Couprie and Hugues Talbot, in DGCI 2009, pages 421-433.
- [CC09] *Surface thinning in 3d cubical complexes*, by John Chaussard and Michel Couprie, in IWCIA 2009, pages 135-148.

JOURNALS

- [CBC10] *Characterization and Detection of Toric Loops in n-Dimensional Discrete Toric Spaces*, by John Chaussard, Michel Couprie and Gilles Bertrand, in Journal of Mathematical Image and Vision, volume 36, pages 111-124 (2010).
- [CCT10] *Robust skeletonization using the discrete lambda-medial axis*, by John Chaussard, Michel Couprie and Hugues Talbot, in Pattern Recognition Letters, accepted (2010).

CONTENTS

1 INTRODUCTION

1.1 GENERAL SETTING OF THIS WORK

1.1.1 DIGITAL IMAGE PROCESSING AND TOPOLOGY

In the 60s, it became realistic that computers could be used in order to automatically process data from digital images. The bank industry was mostly interested by these techniques in order to perform automatic text recognition in order to speed up the processing of checks. Text recognition is easily performed by human brain in every day life (less easily when reading a doctor's prescription), but automatically processing hand-written text with a computer is a difficult task, specifically because two different persons don't have the same writing.

A letter can be recognized by looking at various characteristics, such as how many pieces it is made of (for example, the letter "i" is made of two pieces), how many cavity it has (the letter "o" has one cavity), and its general shape ("a" has one cavity and is small, while "q" has also one cavity but is more elongated. Moreover, if one looks the result of a same letter written by different persons, he or she will realize that each letter is different from the other, although they define the same alphabetical symbol. In order to recognize a hand-written letter, an algorithm must therefore be insensitive to slight deformations of a shape.

Topology is the domain of mathematics studying the invariant properties of objects under continuous deformations, and is able to provide tools in order to solve such a problem. Among these tools, *skeletonization* is widespread in image processing: it consists in removing some data from the input image, in order to obtain at the end a simplified object with the same topological properties than the input (and, sometimes, the same shape). This powerful method found applications in many other domains, such as medical image processing ([MS96], [BC02], [DC02], [DLPB99]), material image analysis ([JAB+10], [Plo09], [Com]), environmental sciences ([FDM06]), fingerprint analysis ([CBB01]), polyhedrization ([BM03]), motion capture ([BMV10]), etc.

In the following, we leap back in time to the origins of topology...

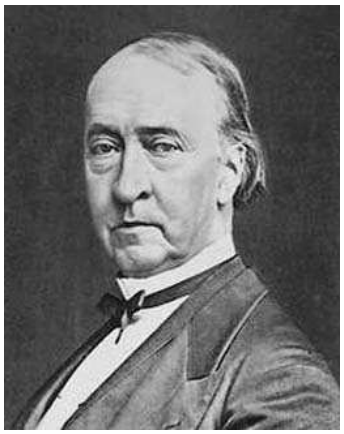
1.1.2 PRESENTATION OF TOPOLOGY

1.1.2.1 (HI)STORY OF TOPOLOGY

In 1736, Euler solved the famous problem of the *Seven Bridges of Königsberg* ([Eul41]), by building the basis of what will become later the *graph theory* (although Euler did not mention any graph in his preliminary solution [HW07]). The problem consisted in knowing if, given a set of islands and a set of bridges connecting these islands, it was possible to walk through each bridge once and only once, and come back to the starting point. The elegant solution proposed by Euler expressed that, in order to solve this problem, no metric measure of the bridges (length of the bridges, angles between

the bridges) was necessary: only the number of bridges connecting each island was to be taken into account.

Leonhard Euler
(1707 - 1783)



Johann Benedict Listing
(1808 - 1882)

Jules Henri Poincaré
(1854 - 1912)

Euler continued to work on the field of what he called *geometria situs* (geometry of position), where the basic information contained in the shapes of geometric figures is solely studied. The term *topology* was introduced for the first time by Johann Benedict Listing ([Lis47]) and slowly replaced the term geometria situs. Poincaré defined, in the end of the 19th century, the fundamental notions of homotopy, homology and fundamental group ([Poi95]). Popularity of topology grew larger during the 20th century, when it appeared that this new domain of mathematics was in fact a root of mathematics at the base of many new concepts.

1.1.2.2 CONTINUOUS DEFORMATION AND TOPOLOGICAL INVARIANTS

In topology, two objects are equivalent if one can be obtained from the other by *continuous deformation*: intuitively, a continuous deformation is any transformation that does not involve cutting or piercing the object, or gluing together separate parts. Some examples of continuous deformations are stretching, bending, torsion. For this reason, topology is also nicknamed the geometry of plasticine, as topologists often consider objects as if they were made of deformable material.

Continuous deformation were formalized by Poincaré through three main concepts: *homotopy of functions*, *homotopy of spaces* and *homeomorphism*. Two objects which are equivalent by continuous deformation are said to be homeomorphic (homotopy is a more general kind of deformation which allows infinite pinching of some parts of the objects). Knowing if two objects are homotopic or homeomorphic is generally undecidable ([Mar60]).

Based on this, it appeared that, under continuous deformation, some characteristics of an object stay invariant. Among these characteristics appear, in all dimension, the number of connected components (number of pieces) of the object: whatever the continuous deformation applied to an object, its number of connected components stay the same (although their relative size and shape might change). In two dimensions and more, another topological invariant appear: the number of cavities of the object (number of connected components of the complement). In three dimensions and more,

the number of tunnels (or holes) of the object appears to be another topological invariant. Indeed, one can say that, in n-dimensions, there exists n "types" of invariants to take into account in an object.

Topological invariants of an object are intrinsic properties of the object, and should not depend of the ambient space where the object is embedded. For this reason, all topological invariants of objects were defined "inside" objects. For example, in 2d, the cavities of objects are not defined as connected components of the non-object, but rather as families of loops, inside the object, that cannot be continuously distorted into a single point. In 3d, the cavities are defined as families of 2 dimensional manifolds inside the object that cannot be distorted to a single point, while tunnels are defined as families of loops that cannot be reduced to a point. In 4d, the cavities are families of 3 dimensional manifolds that cannot be reduced to a single point. This has been formalized by Poincaré through the notions of *homotopy group* and n*-dimensional cycles*.

1.1.2.3 DEFORMATION RETRACT AND SKELETON

Deformation retract is another type of deformation, which can qualify two homotopic spaces such that one of them is a subset of the other: this continuous deformation can be seen as a simplification of an object which preserves the topological invariants. As said previously, knowing if an object is the deformation retract of another is generally undecidable.

In the 60s, Blum introduced the notion of *skeleton* ([Blu62],[Blu67]) in order to compute a particular deformation retract from a shape. First defined as the meeting point of a grassfire propagating from the border of a shape, it was then defined as the set of centers of maximal balls contained in the object. Other definitions (from various domains, such as mathematical morphology [Ser82], [SM93], [Soi99] and computational geometry [OK95] [AM96] [NSK+97] [AL01]) were given, leading to various efficient methods for computing (homotopic) skeletons of a shape.

In an n-dimensional space, the skeleton of a shape is at most an $(n-1)$-dimensional shape: we say that the skeleton is *thin*.

1.1.2.4 DIGITAL TOPOLOGY

In order to apply topological concepts to digital image processing, it was necessary to propose discrete versions of the main fundamental topological concepts. This lead some authors in proposing discrete versions of homotopy, fundamental group, deformation retract, etc, such as Rosenfeld ([Ros73], [Ros81]), Kong ([Kon89], [KRR92]), Malgouyres ([Malo0])...

Skeletonization algorithms were also proposed in the digital topology framework ([DP81], [Vin91], [TV92], [Pud98]). A more extensive description of skeletonization algorithms in the discrete framework is given in Sec. 3. In this work, we focus on homotopic thinning: basically, it consists in removing points from a discrete object, in order to obtain, at each step, a new object homotopic to the previous one, until stability is reached.

In applications, discrete skeletons are used to simplify a shape in the two or three dimensional discrete space, while preserving its topological characteristics. It is more easy to study the skeleton of a shape rather than the shape itself, as it contains less data. In the discrete framework, skeletons are not always thin (of course, it depends on what definition of thinness is used in the discrete framework).

1.2 CONTRIBUTIONS AND CONTENTS OF THIS THESIS

This thesis initially aimed at studying how topological tools can help material analysis. All along the chapters of this work, images of materials will be used as examples and results of new algorithms. Moreover, results will be also presented on other types of shapes, in order to prove that the methods exposed here are not aimed at only one domain.

- The first part will be about fluid flow simulation. The global goal of the project surrounding this work was to decide if it is possible to reduce fluid flow simulation calculation of a material to the skeleton of the pore space of this material, and still obtain precise measures on the tested material. In this work, the skeleton should be a curvilinear object, and the sample of the tested material is considered as embedded in a toric space. Detecting and removing parts of the material which do not wrap around the toric space allows to remove a "source" of surfaces in the skeleton of the pore space.
  - In order to do so, we first define basic notions in the toric space, such has neighbourhood, homotopy and loops. Our definition of loops allow to avoid some problems that might occur in small toric spaces.
  - Then, we define the *wrapping vector* of loop, which is invariant under homotopic deformation. Based on this, we find some well-known results about the fundamental group of the torus.
  - Finally, we propose an algorithm which allows to detect if some connected components of an object are wrapped around the torus or not. We show also that our algorithm allows to precisely understand how a component wraps around the torus.
- The second part of this thesis is about a new shape descriptor, called the *discrete λ-medial axis*, in the discrete framework. In some applications, the topology of a shape is not sufficient for characterizing the shape: it is also necessary to study its geometry. This new shape descriptor can be used during skeletonization algorithm in order to retain, during the thinning, important visual features of an object. Moreover, in the continuous framework, it possesses good stability to contour deformation of the shape: we evaluate this stability in the discrete framework by comparing the discrete version of the λ-medial axis to other medial axes.
  - We first define the λ-medial axis as it was originally defined in the continuous framework ([CL05]), and then adapt this definition to the discrete framework.
  - We then give an algorithm for computing the discrete λ-medial axis and, as the proposed algorithm is not linear in time (depending on the number of points of the object), we propose a second shape descriptor, called the discrete λ′-medial axis, which can be computed in linear time.
  - We perform a series of tests, in order to compare these two shape descriptors with other medial axes (the Euclidean medial axis and the integer medial axis): we compare the stability of each axis to rotation and to noise.
- The last part is taking place in the cubical complex framework. In this framework, we show that decomposition of a skeleton is straightforward and has properties, close to the ones of the continuous framework.

  - After introducing some basic notions about cubical complexes, we propose a parallel-thinning scheme for performing homotopic thinning of objects embedded in the cubical complex framework. We also give a methodology for using any shape descriptor of the discrete framework in this new framework (and preserve, during homotopic thinning, important visual information on the shape). Finally, we prove that, under some conditions, the results of this thinning is thin.
  - We give an algorithm, in the cubical complex framework, for performing homotopic thinning which preserves some visual features of the object and which does not require any user input. After proving that the result of this algorithm produces thin result, we compare this algorithm with other "parameter-free" skeletonization algorithms of the discrete framework. To do so, we give a measure of the visual quality of a skeleton.
  - We finally explain how a skeleton can be decomposed easily into components in the cubical complex framework, and we study the results of the decomposition.

- In the appendix, we give some basic definitions about digital topology and image analysis, as well as extra algorithms that were isolated from the rest of this work in order to keep fluent the reading of this thesis.

Part I

TOPOLOGY IN TORIC SPACES

2

DETECTING OBJECT WRAPPED INSIDE TORIC SPACES

In this chapter, we study how topology can help solving problems in fluid flow simulation. In such applications, materials are often considered as embedded inside a toric space. In this case, it can be difficult to detect grains of the material, which are components that should be filtered out from the image. Thanks to a complete characterization of the digital fundamental group of the torus, we propose an algorithm capable of detecting objects wrapping around the torus. Based on this, we can detect and remove grains from an image.

This work was presented in the DGCI 2008 conference in Lyon, France ([CBC08]), and an extended version was published ([CBC10]).

Contents

2.1 HOW ARE MATERIAL ANALYSIS AND TORIC SPACES RELATED?

2.1.1 POROUS MATERIALS AND SKELETONS

A *percolating porous material* is "a solid punctured by voids, through which, when connected to the exterior, a fluid or gas can flow" ([Plo09]). These materials are used in many applications, such as car industry where ceramics are used in the manufacturing of particle filters, petrology where studying sandstones can help extracting more oil from wells, agriculture where understanding soils structures can help predicting water retention process, etc.

The set of the voids of a porous material is called the *porosity*, or the *pore space*. Many applications involving porous materials require measuring the *permeability* of the material, which tells how easily a fluid can flow through the pore space of a material. The permeability can be obtained by computing the flow map of the fluid through the pore space (to each point of the pore space, a vector giving the direction and velocity of the flow is associated). Computation of such map can be performed on the image of the material and the result is an image where a vector is associated to each voxel of the porous space ([ABE94], [BV00], [BNSC05]).

As pointed out in [Plo09], the computation of the flow map is "computationally intensive", and can therefore be long. For this reason, it is necessary to use simplified models of the pore space in order to compute the flow map.

The skeletonization operation allows to simplify an object's structure while preserving important topological information of the original data. The main motivation of this work is to test if the skeleton of the pore space could be used as a valid simplified model in order to compute a flow map.

The work presented hereafter focuses on obtaining a curvilinear skeleton from the pore space of a material's image. Computation of a flow map based on the skeleton and testing the validity of the results will be presented in [Com].

2.1.1.1 FROM GREY SCALE IMAGES TO BINARY IMAGES

The work presented here starts with a grey-level image obtained with 3d microtomography imagery. The first step is to perform a binarisation (also called segmentationn) of the image, meaning that each voxel of the image must be labelled as part of the material (the foreground) or part of the pore space (the background). The most popular segmentation methods involve thresholding ([SS01]), level sets ([OS88], [ZCMO96], [OP03]), region growing([PL90], [CL94], [FZBH05]), graph partitioning ([Gra06], [WL93], [Zah71]), watershed ([BM92], [CB97], [CBNC10]), split and merge algorithms ([HP76]) or energy minimization ([LV01], [BVZ01]).

In our case, the grey-level images of porous material are segmented using a filtering step based on anisotropic diffusion, followed by a threshold ([PM90], [MN01]).

2.1.2 EMBEDDING THE IMAGE IN A TORIC SPACE

When simulating a fluid flow through a porous material, we only possess the image of a sample of this material. The simulation calculation, in order to produce correct results, should be performed on the whole material, and not only on one of its sample. However, it is not usually possible to obtain the image of the whole piece of material

(which is very big). Therefore, the whole material is approximated by the tessellation of the space made up by copies of the sample we have, under the condition that the sample's volume exceeds the Representative Elementary Volume[1] of the material ([Bea72], [Sch74], [VCAB07]).

When the whole Euclidean space is tiled this way, the results of the fluid flow simulation is itself a tessellation of the local flow obtained inside any copy of the sample (see Fig. 2a). When considering the flow obtained inside any copy of the sample, it appears that the flow leaving the sample by one side comes back inside the sample by the opposite side (see Fig. 2b). Thus, it is possible to perform the fluid flow simulation only on the sample (and not on the tessellation), under the condition that its opposite sides are joined: with this construction, the sample is embedded inside a toric space ([Hat02], [Sti80]). As depicted on the sequence shown on Fig. 1, when joining the opposite's border of an image, one obtains a torus. In the following chapter, we consider all image-spaces as toric spaces.

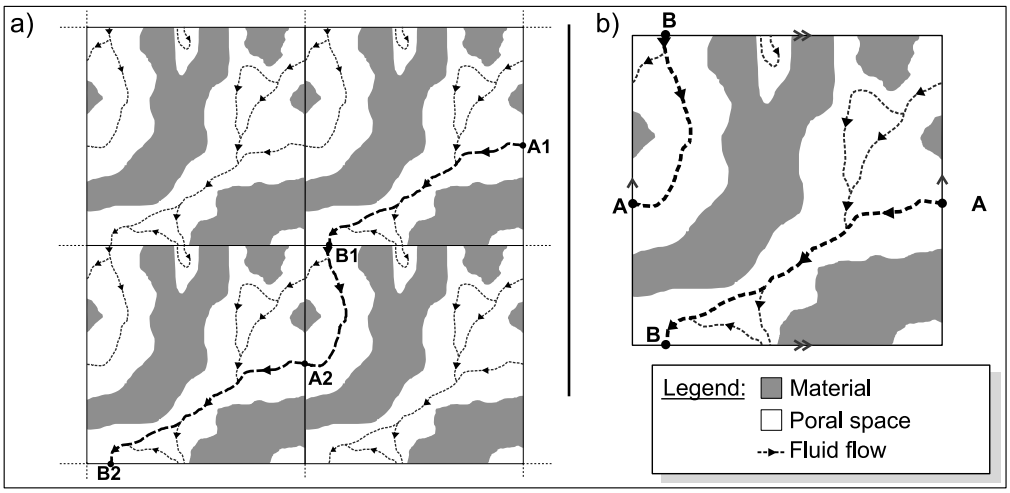


Figure 1: **Simulating a fluid flow -** When simulating a fluid flow, a porous material (in gray) can be approximated by the tessellation of one of its samples (see **a**). When the results of the simulation are obtained (the dotted lines), one can see that the fluid flow through the mosaic is the tessellation of the fluid flow simulation results obtained in one sample. For example, one can look at the bold dotted line in **a)**: the flow going from A1 to B1 is the same than the flow going from A2 to B2. It is therefore possible to perform the fluid flow simulation through only one sample and, in order to obtain the same results than in **a**, connect the opposite sides of the sample (see **b**): the sample is embedded inside a toric space.

2.1.3 REMOVING THE GRAINS FROM AN IMAGE: PROBLEM IN TORIC SPACE

2.1.3.1 WHAT IS A GRAIN/CAVITY IN A TORIC SPACE

In a real fluid flow, grains of a material (pieces of the material which are not connected with the borders of the sample) do not have any effect on the final results, as these grains eventually either evacuate the object with the flow or get blocked and connect with the rest of the material. Thus, before performing a fluid flow simulation on a

1 The Representative Elementary Volume of a material is the minimal volume that a sample of this material should have in order for a measurement performed on the sample be representative of the whole material.

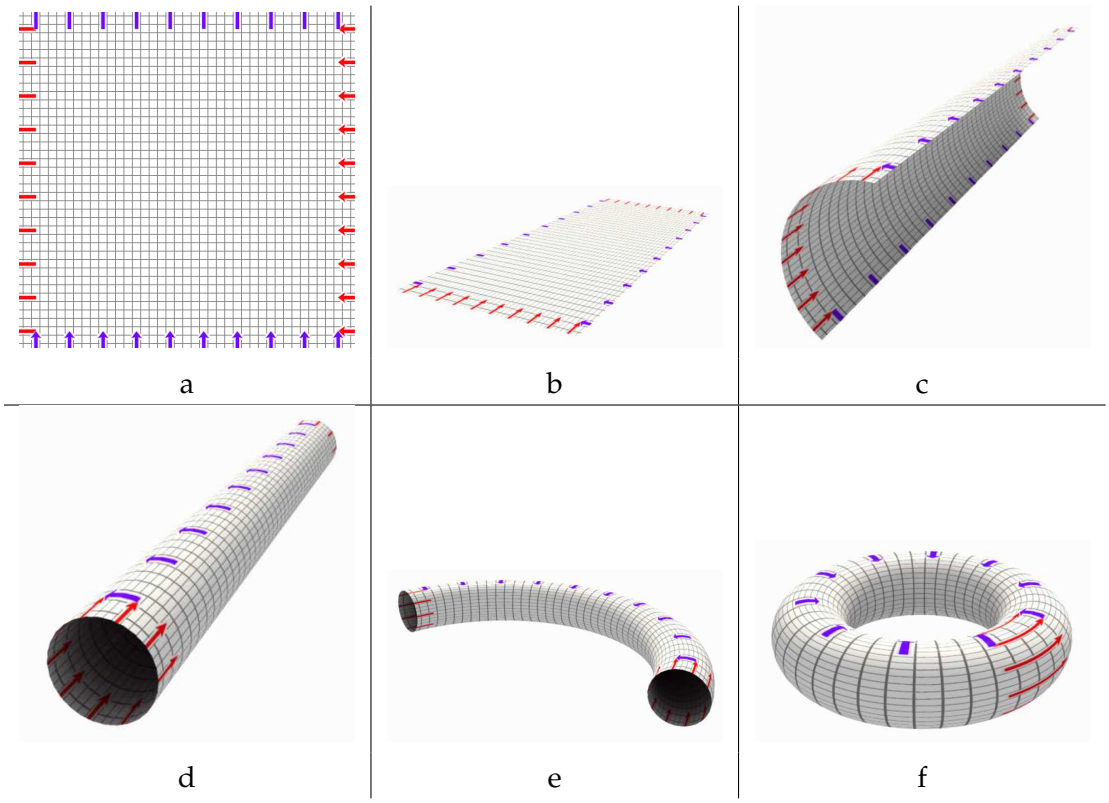


Figure 2: **Making a toric space -** When considering a 2d image (in **a**), and folding it in order to join the opposite sides together, one obtains a torus (in **f**).

sample, it is necessary to remove its grains. Typically, in a finite Euclidean space, a grain is a connected component which does not "touch" the borders of the space).

However, considering the sample inside a toric space leads to new difficulties. Indeed, such a space does not possess any border, therefore, the definition of a grain cannot rely on the space's borders. In order to understand what is a grain (or cavity) in a toric space, let us consider an image and let us tessellate the entire infinite space with it. Intuitively, a grain is a component which, in the tessellation, is contained in a finite bounding box; in other words, in the (infinite) tessellation, the grain does not "spread to the infinite" and does not touch the borders of the space. Any component which is not a grain spreads to the infinite of the space.

For example, when the whole Euclidean space is tessellated with the image presented on Fig. 4a (on the left), then the piece of material can be "enclosed inside a curve" (the dotted lines on the right side) and produces a grain in the tessellation. On the other hand, when the whole Euclidean space is tessellated with the image presented on Fig. 4b, the piece of material "spreads to the infinite" and cannot be considered as a grain in the tessellation.

This intuitive characterization requires to tessellate the infinite space with the image, and cannot be used directly for characterizing efficiently grains and cavities in toric spaces. We need to find another definition in order to produce an algorithm detecting grains in a toric space.

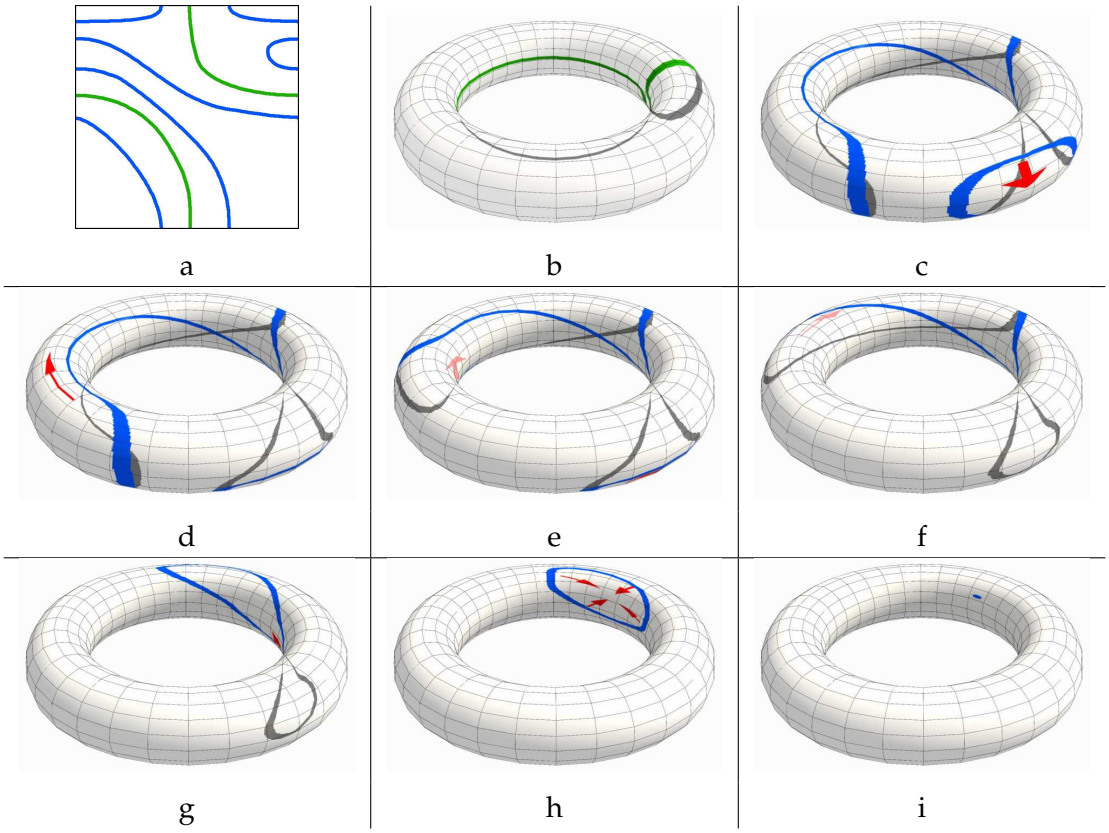


Figure 3: **Loops in toric space -** In a toric space, some loops cannot be reduced to a single point while other can. The green loop on **a** cannot be reduced to a point by continuous deformation, as it wraps around the torus (see **b**). The blue loop on **a** be continuously distorted into a single point (see **c** to **i**).

2.1.3.2 TORIC SPACES ARE NOT SIMPLY CONNECTED

A space is said to be *simply connected* if every loop inside the space is homotopic to a single point (in other words, it can be continuously distorted into a point). The fundamental group of the space is said to be trivial. In a toric space, some loops are not homotopic to a point: these loops wrap around the torus and cannot be continuously distorted into a point: they are called *toric loops*. The figure 3a, which must be seen as a 2d toric space (opposite sides joint together), provides an example of a toric loop (in green) and an example of a loop homotopic to a point. Indeed, the green loop wraps around the torus (see Fig. 3b) while the blue loop can be distorted into a single point (see Fig. 3c to i).

2.1.3.3 CHARACTERIZING GRAINS IN A TORIC SPACE

A grain is a component which does not wrap completely around the toric space, and can therefore be enclosed in a finite bounding box when tessellating the Euclidean space with the image. On the opposite, a non-grain wraps around the toric space and produces infinite components when tessellating the Euclidean space with the image. In other words, a non-grain contains a toric loop, while a grain does not.

One can see, on Fig. 4a, that any closed curve drawn inside the piece of material can be reduced to a point: the piece is a grain in the toric space. However, on Fig. 4b, there exists some curves inside the piece of material (the dotted line) which cannot be reduced to a point: the piece contains a toric loop and is not a grain.

Generally, in digital topology, deciding if two objects are homotopic is most of the time undecidable. However, deciding if a loop in discrete toric spaces can be continuously reduced to a points is possible. We build, in the following, a simple framework adapted to discrete toric spaces leading to an algorithm allowing to know if a component contains a toric loop (and therefore, allowing to decide if it is a grain or not).

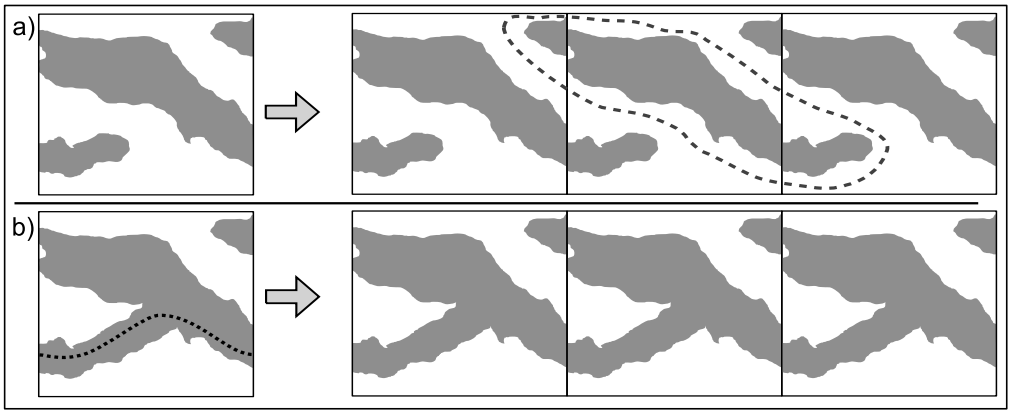


Figure 4: **Grains in toric spaces -** The image in **a** contains no grain based on the 'border criterion'; when the Euclidean space is tessellated with copies of the image, grains appear (the circled connected component is an example of grain). In **b**, the connected component has toric loops (e.g. the dotted line) and when the Euclidean space is tessellated with copies of the image, no grain appear.

2.2 BASIC NOTIONS IN TORIC SPACES

In order to define formally a *toric loop*, we give additional definitions related to toric spaces.

2.2.1 DISCRETE TORIC SPACES

An n-dimensional torus is classically defined as the direct product of n circles (see [Hato2]). In the following, we give a discrete definition of toric space, based on modular arithmetic (see [GKP94]).

Given d a positive integer, we set $\mathbb{Z}_d = \mathbb{Z}/d\mathbb{Z} = \{0, ..., d-1\}$, and we denote by $\oplus_d$ the operation such that for all $a, b \in \mathbb{Z}$, $(a \oplus_d b)$ is the element of $\mathbb{Z}_d$ congruent to $(a + b)$ modulo d. We point out that $(\mathbb{Z}_d, \oplus_d)$ is the cyclic group of order d.

Let n be a positive integer, $\mathbf{d} = (d_1, ..., d_n) \in \mathbb{N}^n$, and $\mathbb{T}^n = \mathbb{Z}_{d_1} \times ... \times \mathbb{Z}_{d_n}$, we denote by $\oplus_{\mathbf{d}}$ the operation such that for all $\mathbf{a} = (a_1, ..., a_n) \in \mathbb{Z}^n$ and $\mathbf{b} = (b_1, ..., b_n) \in \mathbb{Z}^n$, $\mathbf{a} \oplus_{\mathbf{d}} \mathbf{b} = (a_1 \oplus_{d_1} b_1, ..., a_n \oplus_{d_n} b_n)$. The group $(\mathbb{T}^n, \oplus)$ is the direct product of the n groups $(\mathbb{Z}_{d_i}, \oplus_{d_i})_{(1 \leqslant i \leqslant n)}$, and is an *n-dimensional discrete toric space* [Hato2].

The scalar d_i is *the size of the i-th dimension* of $\mathbb{T}^n$, and $\mathbf{d}$ is the *size (vector)* of $\mathbb{T}^n$. For simplicity, the operation $\oplus_{\mathbf{d}}$ will be also denoted by $\oplus$.

2.2.2 NEIGHBOURHOOD IN TORIC SPACE

As in $\mathbb{Z}^n$ (see appendix, Sec. 9.1, p. 204), various neighbourhood relations can be defined in a toric space.

Definition 2.2.1 *An* m-step $(0 < m \leqslant n)$ *is a vector* $\mathbf{s} = (s_1, ..., s_n)$ *of* $\mathbb{Z}^n$ *such that, for all* $i \in [1; n]$, $s_i \in \{-1, 0, 1\}$ *and* $|s_1| + ... + |s_n| \leqslant m$.

Two points $\mathbf{a}, \mathbf{b} \in \mathbb{T}^n$ *are* m-adjacent *if there exists an* m*-step* $\mathbf{s}$ *such that* $\mathbf{a} \oplus \mathbf{s} = \mathbf{b}$.

Note that the steps must not be considered as elements of $\mathbb{T}^n$, but rather as elements of $\mathbb{Z}^n$. From this definition, we derive the m*-neighbourhood* of a point x as the set of all points which are m-adjacent to x.

In 2D, the 1- and 2-adjacency relations respectively correspond to the 4- and 8-neighbourhood [KR89] (defined in Sec. 9.1, p. 204) adapted to two-dimensional toric spaces. In 3D, the 1-, 2- and 3-adjacency relations can be respectively seen as the 6-, 18- and 26-neighbourhood [KR89] adapted to three-dimensional toric spaces (defined in Sec. 9.1, p. 204).

If the coordinates of the size vector of $\mathbb{T}^n$ are all strictly greater than 2, then the m-neighbourhood of any element of $\mathbb{T}^n$ is isomorphic to the m-neighbourhood of any element of $\mathbb{Z}^n$.

Based on the m*-adjacency relation* previously defined, we introduce the notion of m*-connectivity*.

Definition 2.2.2 *A set of points* X *of* $\mathbb{T}^n$ *is* m-connected *if, for all* $\mathbf{a}, \mathbf{b} \in X$, *there exists a sequence* $(\mathbf{x_1}, ..., \mathbf{x_k})$ *of elements of* X *such that* $\mathbf{x_1} = \mathbf{a}$, $\mathbf{x_k} = \mathbf{b}$ *and for all* $i \in [1; k-1]$, $\mathbf{x_i}$ *and* $\mathbf{x_{i+1}}$ *are m-adjacent.*

2.2.3 LOOPS IN TORIC SPACES

Classically, in $\mathbb{Z}^n$, an m-loop is defined as a sequence of m-adjacent points such that the first point and the last point of the sequence are equal [Kon89]. In this paper, we define a loop as a sequence of m-steps, which describes the direction followed by the loop in the toric space. This new definition allows us to give simple intermediate properties and proofs leading to our main theorem (Th. 2.4.11).

Definition 2.2.3 *Given* $\mathbf{p} \in \mathbb{T}^n$, *an* m-loop (of base point $\mathbf{p}$) *is a pair* $\mathcal{L} = (\mathbf{p}, V)$, *where* $V = (\mathbf{v_1}, ..., \mathbf{v_k})$ *is a sequence of* m*-steps such that* $(\mathbf{p} \oplus \mathbf{v_1} \oplus ... \oplus \mathbf{v_k}) = \mathbf{p}$.

The number k *is the* length of $\mathcal{L}$. *We call* i-th point of $\mathcal{L}$, *with* $1 \leqslant i \leqslant k+1$, *the point* $(\mathbf{p} \oplus \mathbf{v_1} \oplus ... \oplus \mathbf{v_{i-1}})$. *The loop* $(\mathbf{p}, ())$ *is called the* trivial loop of base point $\mathbf{p}$.

Remark In this definition, the $(k+1)$-th point of $\mathcal{L}$ is $\mathbf{p}$, and has been defined in order to make some propositions and proofs more simple.

Remark This definition of loops in toric space allows to remove an ambiguity which can exist in small toric spaces. Indeed, when considering loops as a sequence of m-adjacent points, an ambiguity arises in toric spaces where one dimension has a size equal to 1 or 2. For example, let us consider the two-dimensional toric space $\mathbb{T}^2 = \mathbb{Z}_3 \times \mathbb{Z}_2$, and the 2-adjacency relation on $\mathbb{T}^2$. Let us also consider $\mathbf{x_1} = (1, 0)$ and $\mathbf{x_2} = (1, 1)$ in $\mathbb{T}^2$, and let us consider the sequence of points $\mathcal{L} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_1})$.

The sequence $\mathcal{L}$ could either be the loop passing by $\mathbf{x_1}$ and $\mathbf{x_2}$ and doing a 'u-turn' to come back to $\mathbf{x_1}$, or either be the loop passing by $\mathbf{x_1}$ and $\mathbf{x_2}$, and 'wrapping around' the toric space in order to reach $\mathbf{x_1}$ without making any 'u-turn', as shown on Fig. 5. In toric spaces of small size, defining a loop as a sequence of m-adjacent points may lead to such ambiguity.

However, considering a loop as a sequence of m-steps removes the ambiguity: let $\mathbf{v}$ be the vector $(0,1)$, the loop passing by $\mathbf{x_1}$ and $\mathbf{x_2}$ and making a u-turn is $(\mathbf{x_1},(\mathbf{v},-\mathbf{v}))$ (see Fig. 5-**b**), while the loop wrapping around the toric space is $(\mathbf{x_1},(\mathbf{v},\mathbf{v}))$ (see Fig. 5-**c**). Since m-steps are elements of $\mathbb{Z}^n$, we have $\mathbf{v} \neq -\mathbf{v}$.

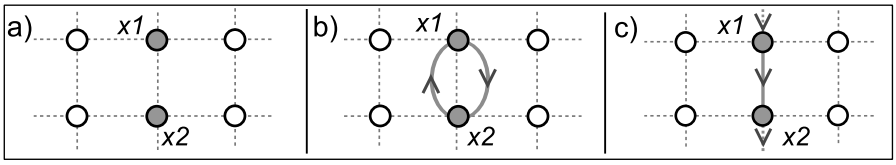


Figure 5: **Loops in small toric spaces -** In the toric space $\mathbb{Z}_3 \times \mathbb{Z}_2$ (see **a**), the sequence of points $(\mathbf{x_1},\mathbf{x_2},\mathbf{x_1})$ can be interpreted in two different ways: **b)** and **c)**.

2.3 LOOP HOMOTOPY IN TORIC SPACES

2.3.1 HOMOTOPIC LOOPS

We now define an equivalence relation between loops, corresponding to an homotopy, inside a discrete toric space. An equivalence relation between loops inside $\mathbb{Z}^2$ and $\mathbb{Z}^3$ was defined in [Kon89], however, it cannot be adapted to discrete toric spaces. Observe that the following definition does not constrain the loops to lie in a subset of the space, on the contrary of the definition given in [Kon89].

Definition 2.3.1 *Let $\mathcal{K} = (\mathbf{p}, U)$ and $\mathcal{R} = (\mathbf{p}, V)$ be two m-loops of base point $\mathbf{p} \in \mathbb{T}^n$, with $U = (\mathbf{u_1},...,\mathbf{u_k})$ and $V = (\mathbf{v_1},...,\mathbf{v_r})$. The two m-loops $\mathcal{K}$ and $\mathcal{R}$ are directly homotopic if one of the three following conditions is satisfied:*

1. *There exists $j \in [1;r]$ such that $\mathbf{v_j} = 0$ and $U = (\mathbf{v_1},...,\mathbf{v_{j-1}},\mathbf{v_{j+1}},...,\mathbf{v_r})$.*
2. *There exists $j \in [1;k]$ such that $\mathbf{u_j} = 0$ and $V = (\mathbf{u_1},...,\mathbf{u_{j-1}},\mathbf{u_{j+1}},...,\mathbf{u_k})$.*
3. *There exists $j \in [1;k-1]$ such that*
   - *$V = (\mathbf{u_1},...,\mathbf{u_{j-1}},\mathbf{v_j},\mathbf{v_{j+1}},\mathbf{u_{j+2}},...,\mathbf{u_k})$, and*
   - *$\mathbf{u_j} + \mathbf{u_{j+1}} = \mathbf{v_j} + \mathbf{v_{j+1}}$, and*
   - *$(\mathbf{u_j} - \mathbf{v_j})$ is an n-step.*

Remark In the case 1 (resp. 2 and 3), we have $k = r - 1$ (resp. $(r = k-1)$ and $(r = k)$).

Remark It may be observed that in the above definition, the parameter m is used to specify that we consider m-loops, but it is not taken into account in order to decide if two m-loops are directly homotopic.

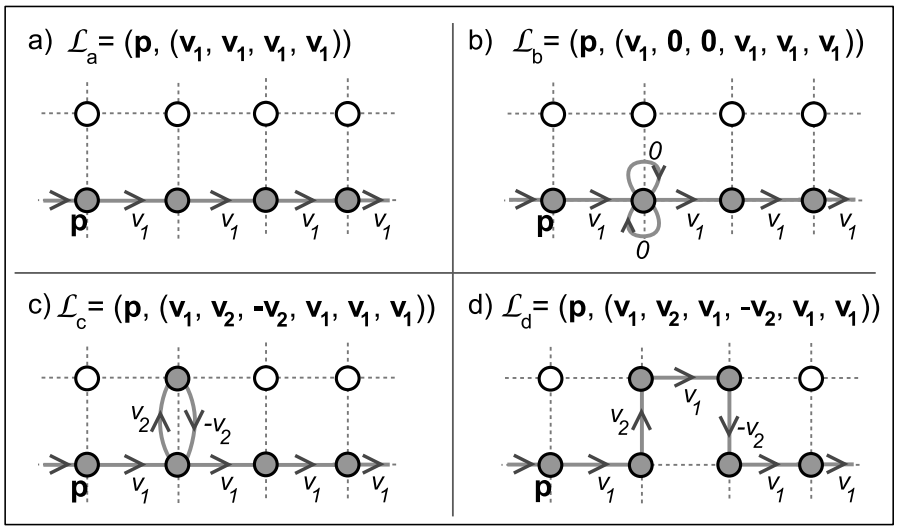


Figure 6: **Homotopic loops -** The 1-loops $\mathcal{L}_a, \mathcal{L}_b, \mathcal{L}_c$ and $\mathcal{L}_d$ are homotopic.

Definition 2.3.2 *Two* m*-loops* $\mathcal{K}$ *and* $\mathcal{R}$ *of base point* $\mathbf{p} \in \mathbb{T}^n$ are homotopic *if there exists a sequence of* m*-loops* $(\mathcal{C}_1, ..., \mathcal{C}_q)$ *such that* $\mathcal{C}_1 = \mathcal{K}$, $\mathcal{C}_q = \mathcal{R}$ *and for all* $j \in [1; q-1]$, $\mathcal{C}_j$ *and* $\mathcal{C}_{j+1}$ *are directly homotopic.*

Example In the toric space $\mathbb{Z}_4 \times \mathbb{Z}_2$, let us consider the point $\mathbf{p} = (0,0)$, the 1-steps $\mathbf{v_1} = (1,0)$ and $\mathbf{v_2} = (0,1)$, and the 1-loops $\mathcal{L}_a$, $\mathcal{L}_b$, $\mathcal{L}_c$ and $\mathcal{L}_d$ (see Fig. 6). The loops $\mathcal{L}_a$ and $\mathcal{L}_b$ are homotopic (two insertions of null vector have been performed), the loops $\mathcal{L}_b$ and $\mathcal{L}_c$ are directly homotopic, and the loops $\mathcal{L}_c$ and $\mathcal{L}_d$ are also directly homotopic. Thus, $\mathcal{L}_a$ and $\mathcal{L}_d$ are homotopic.

On the other hand, it may be seen that the 1-loops depicted on Fig. 5-**b** and on Fig. 5-**c** are not directly homotopic.

2.3.2 FUNDAMENTAL GROUP

Initially defined in the continuous space by Henri Poincaré in 1895 [Poi95], the fundamental group is an essential concept of topology, based on the homotopy relation, which has been transferred into different discrete frameworks (see e.g. [Kon89], [Mal01], [BCP09]).

Given two m-loops $\mathcal{K} = (\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_k}))$ and $\mathcal{R} = (\mathbf{p}, (\mathbf{v_1}, ..., \mathbf{v_r}))$ of same base point $\mathbf{p} \in \mathbb{T}^n$, the *product of* $\mathcal{K}$ *and* $\mathcal{R}$ is the m-loop $\mathcal{K}.\mathcal{R} = (\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_k}, \mathbf{v_1}, ..., \mathbf{v_r}))$. The identity element of this product operation is the trivial loop $(\mathbf{p}, ())$, and for each m-loop $\mathcal{K} = (\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_k}))$, we define the inverse of $\mathcal{K}$ as the m-loop $\mathcal{K}^{-1} = (\mathbf{p}, (-\mathbf{u_k}, ..., -\mathbf{u_1}))$.

The symbol $\prod$ will be used for the iteration of the product operation on loops. Given a positive integer w, and an m-loop $\mathcal{K}$ of base point $\mathbf{p}$, we set $\mathcal{K}^w = \prod_{i=1}^{w} \mathcal{K}$ and $\mathcal{K}^{-w} = \prod_{i=1}^{w} \mathcal{K}^{-1}$. We also define $\mathcal{K}^0 = (\mathbf{p}, ())$.

The homotopy of m-loops is a reflexive, symmetric and transitive relation: it is therefore an equivalence relation and the equivalence class, called *homotopy class*, of an

m-loop $\mathcal{R}$ is denoted by $[\mathcal{R}]$. The product operation can be extended to the homotopy classes of m-loops of same base point: the product of $[\mathcal{K}]$ and $[\mathcal{R}]$ is $[\mathcal{K}].[\mathcal{R}] = [\mathcal{K}.\mathcal{R}]$. It may be easily seen that this binary operation is well defined since, if $\mathcal{K}' \in [\mathcal{K}]$ and $\mathcal{R}' \in [\mathcal{R}]$, then $\mathcal{K}'.\mathcal{R}' \in [\mathcal{K}.\mathcal{R}]$.

We now define the fundamental group of $\mathbb{T}^n$.

Definition 2.3.3 *Given an* m*-adjacency relation on* $\mathbb{T}^n$ *and a point* $\mathbf{p} \in \mathbb{T}^n$*, the* m-fundamental group of $\mathbb{T}^n$ with base point $\mathbf{p}$ *is the group formed by the homotopy classes of all m-loops of base point* $\mathbf{p} \in \mathbb{T}^n$ *under the product operation.*

The identity element of this group is the homotopy class of the trivial loop, and for each m-loop $\mathcal{K}$ of base point $\mathbf{p}$, the inverse of $[\mathcal{K}]$ is $[\mathcal{K}^{-1}]$, since $[\mathcal{K}.\mathcal{K}^{-1}] = [(\mathbf{p}, ())]$.

The choice of the base point leads to different fundamental groups which are isomorphic to each other ([Mau96], Th. 3.2.16). Thus, in the following, we sometimes talk about the m-fundamental group of $\mathbb{T}^n$, without specifying the base point.

2.4 TORIC LOOPS: CHARACTERIZATION AND DETECTION

The toric loops, informally evoked in the introduction, can now be formalised using the definitions given in the previous sections.

Definition 2.4.1 *In* $\mathbb{T}^n$*, we say that an m-loop is a* toric m-loop *if it does not belong to the homotopy class of a trivial loop.*

A connected subset of $\mathbb{T}^n$ *is* wrapped in $\mathbb{T}^n$ *if it contains a toric m-loop.*

The notion of grain introduced informally in Sec. 2.1.3.3 may now be defined:

Definition 2.4.2 *A connected component of* $\mathbb{T}^n$ *is* a grain *if it is not wrapped in* $\mathbb{T}^n$*.*

2.4.1 ALGORITHM FOR DETECTING WRAPPED SUBSETS IN A TORIC SPACE

In order to know whether a connected subset of $\mathbb{T}^n$ is wrapped or not, it is not necessary to build all the m-loops which can be found in the subset: the Wrapped Subset Detector (WSD) algorithm (see Alg. 1) answers this question in linear time (more precisely, in $O(N.M)$, where N is the number of points of $\mathbb{T}^n$, and M is the number of distinct m-steps, under the condition of choosing the right data structures, such as arrays for the image and for the map HasCoord), as stated by the following proposition.

Proposition 2.4.3 *Let* $\mathbb{T}^n$ *be an n-dimensional toric space of size vector* $\mathbf{d}$*. A non-empty m-connected subset X of* $\mathbb{T}^n$ *is wrapped in* $\mathbb{T}^n$ *if and only if WSD(n,m,* $\mathbb{T}^n$*,*$\mathbf{d}$*,X) is non-empty (see Alg. 1).*

Remark In Alg. 1, the division operation performed on line 11 is a 'coordinate by coordinate' division between elements of $\mathbb{Z}^n$.

To summarize, Alg. 1 'tries to embed' the subset X of $\mathbb{T}^n$ in $\mathbb{Z}^n$: if some incompatible coordinates are detected by the test achieved on l. 10 of Alg. 1, then the object has

Algorithm 1: WSD(n,m,$\mathbb{T}^n$,**d**,X)

Data: An n-dimensional toric space $\mathbb{T}^n$ of dimension vector **d** and a non-empty m-connected subset X of $\mathbb{T}^n$.
Result: A set B of elements of $\mathbb{Z}^n$

```
1  Let p ∈ X; Coord(p) = 0^n; S = {p }; B = ∅;
2  Let HasCoord be a map from T^n to {true, false};
3  foreach x ∈ X do HasCoord(x) = false;
4  HasCoord(p) = true;
5  while there exists x ∈ S do
6      S = S \{x};
7      foreach non-null n-dimensional m-step v do
8          y = x ⊕_d v;
9          if y ∈ X and HasCoord(y) = true then
10             if Coord(y) ≠ Coord(x) + v then
11                 B = B ∪ ((Coord(x) + v - Coord(y))/ d);
12             end
13         end
14         else if y ∈ X and HasCoord(y) = false then
15             Coord(y) = Coord(x) + v;
16             S = S ∪{y};
17             HasCoord(y) = true;
18         end
19     end
20 end
21 return B
```

a feature (a toric loop) which is incompatible with $\mathbb{Z}^n$. A toric 2-loop lying in X is depicted in Fig. 7-**f**.

Before proving Prop.2.4.3 (see Sec. 2.4.2.4), new definitions and theorems must be given: in particular, Th. 2.4.11 establishes an important result on homotopic loops in toric spaces. Before, let us study an example of execution of Alg. 1.

Example Let us consider a subset X of points of $\mathbb{Z}_4 \times \mathbb{Z}_4$ (see Fig. 7-**a**) and the 2-adjacency relation. In Fig. 7-**a**, one element of X is chosen as **p** and is given the coordinates of the origin (see l. 1 of Alg. 1); then we set $\mathbf{x} = \mathbf{p}$. In Fig. 7-**b**, every neighbour **y** of **x** (l. 7,8) which is in X (l. 14) is given coordinates depending on its position relative to **x** (l. 15) and is added to the set S (l. 16).

Then, in Fig. 7-**c**, one element of S is chosen as **x** (l. 5). Every neighbour **y** of **x** is scanned (l. 7,8). If **y** is in X and has already been given some coordinates (l. 9), it is compared with **x**: as the coordinates of **x** and **y** are compatible in $\mathbb{Z}^2$ (the test achieved l. 10 returns false), the set B remains empty. Else, if **y** is in X and has not previously been given coordinates (l. 14) (see Fig. 7-**d**), then it is given coordinates depending on its position relative to **x** (l. 15) and added to the set S.

Finally, in Fig. 7-**e**, another element of S is chosen as **x**. The algorithm tests one of the neighbours **y** of **x** (the left neighbour) which is in X and has already some coordinates (l. 9). As the coordinates of **y** and **x** are incompatible in $\mathbb{Z}^2$ (the points $(-1,1)$ and $(2,1)$

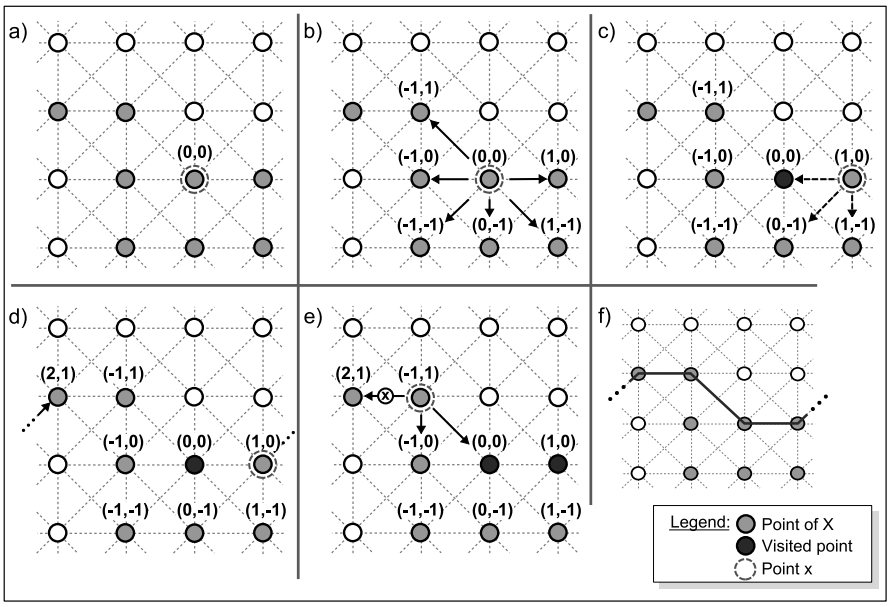


Figure 7: **Example of execution of WSD -** see Ex. 21 for a detailed description.

are not neighbours in $\mathbb{Z}^2$), the algorithm adds $\frac{(-1,1)+(-1,0)-(2,1)}{4} = (-1,0)$ to B (l. 11): according to Prop. 2.4.3, the subset X is wrapped in $\mathbb{T}^n$.

2.4.2 WRAPPING VECTOR AND HOMOTOPY CLASSES IN TORIC SPACES

Deciding if two loops $\mathcal{L}_1$ and $\mathcal{L}_2$ belong to the same homotopy class can be difficult if one attempts to do this by building a sequence of directly homotopic loops which 'links' $\mathcal{L}_1$ and $\mathcal{L}_2$. However, this problem may be solved using the *wrapping vector*, a characteristic which can be easily computed on each loop.

2.4.2.1 WRAPPING VECTOR OF A LOOP

The *wrapping vector* of a loop is the sum of all the elements of the m-step sequence associated to the loop.

Definition 2.4.4 *Let* $\mathcal{L} = (\mathbf{p}, V)$ *be an m-loop, with* $V = (\mathbf{v_1}, ..., \mathbf{v_k})$. *Then the* wrapping vector of $\mathcal{L}$ *is* $\mathbf{w}_\mathcal{L} = \sum_{i=1}^{k} \mathbf{v_i}$.

Remark In Def. 2.4.4, the symbol $\sum$ stands for the iteration of the classical addition operation on $\mathbb{Z}^n$, not of the operation $\oplus$ defined in Sec. 2.2.1.

Example In $\mathbb{T}^2 = \mathbb{Z}_4 \times \mathbb{Z}_4$, depicted on Fig. 8, the loop $\mathcal{K} = (\mathbf{p}, (\mathbf{v_3}, \mathbf{v_2}, \mathbf{v_3}, \mathbf{v_1}, \mathbf{v_3}))$ (see Fig. 8-**a**) has a wrapping vector equal to $(4,4)$, while the loop $\mathcal{L} = (\mathbf{p}, (\mathbf{v_3}, \mathbf{v_1}, \mathbf{v_1}, -\mathbf{v_2}, -\mathbf{v_1}, -\mathbf{v_3}, -\mathbf{v_3}, -\mathbf{v_1}, -\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_1}, \mathbf{v_1}, \mathbf{v_2}))$ has a wrapping vector equal to $(0,0)$ (see Fig. 8-**b**).

We now define the notion of 'basic loops', which will be used for the proof of Prop. 2.4.6 and for building, in Def. 2.4.9, a canonical loop for a given wrapping vector.

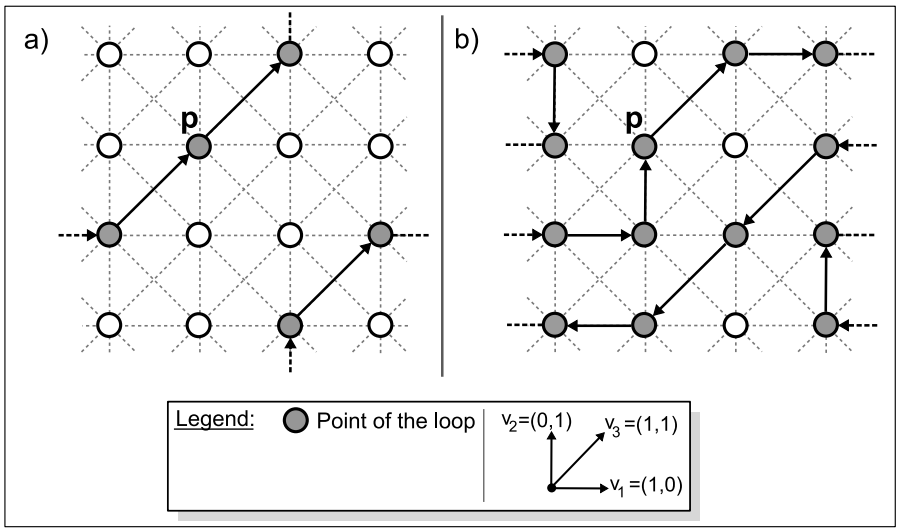


Figure 8: **Wrapping vector -** In $\mathbb{T}^2 = \mathbb{Z}_4 \times \mathbb{Z}_4$, the 2-loop in **a)** has a wrapping vector equal to $(4,4)$, and the 2-loop in **b)** has a wrapping vector equal to $(0,0)$.

Definition 2.4.5 *Let $\mathbb{T}^n$ be an n-dimensional toric space of size vector $\mathbf{d} = (d_1, ..., d_n)$. We denote, for each $i \in [1;n]$, by $\mathbf{b_i}$ the 1-step whose i-th coordinate is equal to 1 and all the others are null, and by B_i the 1-step sequence composed of exactly d_i 1-steps $\mathbf{b_i}$.*

Given $\mathbf{p} \in \mathbb{T}^n$, for all $i \in [1;n]$, we define the i-th basic loop of base point $\mathbf{p}$ *as the 1-loop* $(\mathbf{p}, B_i)$.

Remark For all $i \in [1;n]$, the wrapping vector of the i-th basic loop of base point $\mathbf{p}$ is equal to $(d_i.\mathbf{b_i})$.

The next property establishes that the wrapping vector of any m-loop can only take specific values in $\mathbb{Z}^n$. The proof may be found in Sec. 2.4.3.

Proposition 2.4.6 *Let $\mathbb{T}^n$ be an n-dimensional toric space of size vector $\mathbf{d} = (d_1, ..., d_n)$. A vector $\mathbf{w} = (w_1, ..., w_n)$ of $\mathbb{Z}^n$ is the wrapping vector of an m-loop of $\mathbb{T}^n$ if and only if, for all $i \in [1;n]$, w_i is a multiple of d_i.*

Thanks to Prop. 2.4.6, we can now define the *normalized wrapping vector* of an m-loop.

Definition 2.4.7 *Given $\mathbb{T}^n$ of size vector $\mathbf{d} = (d_1, ..., d_n)$, let $\mathcal{L}$ be an m-loop of wrapping vector $\mathbf{w} = (w_1, ..., w_n)$. The* normalized wrapping vector *of $\mathcal{L}$ is $\mathbf{w}^* = (w_1/d_1, ..., w_n/d_n)$.*

Remark It may be pointed out that, in Alg. 1, the set B contains the normalized wrapping vectors of loops contained in a set X (see Prop. 2.4.14).

Example The wrapping vector and the normalized wrapping vector give information on how a loop 'wraps around' each dimension of a toric space before 'coming back to its starting point'. For example, let $\mathbb{T}^3 = \mathbb{Z}_2 \times \mathbb{Z}_5 \times \mathbb{Z}_7$ (hence, the size vector of $\mathbb{T}^3$ is $(2,5,7)$). A loop with wrapping vector (4,5,0) has a normalized wrapping vector equal to (2,1,0): it wraps two times in the first dimension, one time in the second, and does not wrap in the third dimension.

On Fig. 8, the normalized wrapping vector of loop $\mathcal{K}$ (see Ex. 2.4.2.1), depicted on Fig. 8-**a**, is equal to $(1,1)$ (the loop $\mathcal{K}$ loops one time in each dimension), while the

normalized wrapping vector of $\mathcal{L}$ (see Ex. 2.4.2.1), depicted on Fig. 8-**b**, is equal to $(0,0)$ (the loop $\mathcal{L}$ does not wrap in the toric space).

It may easily be seen that, in $\mathbb{T}^n$, for each $i \in [1;n]$, the normalized wrapping vector of the i-th basic loop of any base point is equal to $\mathbf{b_i}$ (see Def. 2.4.5).

2.4.2.2 EQUIVALENCE BETWEEN HOMOTOPY CLASSES AND WRAPPING VECTOR

It can be seen that two directly homotopic m-loops have the same wrapping vector, as their associated m-step sequences have the same sum (see Def. 2.3.1). Therefore, we have the following property.

Proposition 2.4.8 *Two homotopic m-loops of $\mathbb{T}^n$ have the same wrapping vector.*

The following definition is necessary in order to understand Prop. 2.4.10 and its demonstration, leading to the main theorem of this chapter.

Definition 2.4.9 *Let $\mathbf{p}$ be an element of $\mathbb{T}^n$, and $\mathbf{w}^* = (w_1^*, ..., w_n^*) \in \mathbb{Z}^n$.*
The canonical loop of base point $\mathbf{p}$ and normalized wrapping vector $\mathbf{w}^*$ *is the 1-loop* $\prod_{i=1}^{n} (\mathbf{p}, B_i)^{w_i^*}$, *where $(\mathbf{p}, B_i)$ is the i-th basic loop of base point $\mathbf{p}$.*

Example Consider $\mathbb{T}^3 = \mathbb{Z}_3 \times \mathbb{Z}_2 \times \mathbb{Z}_2$, $\mathbf{w}^* = (1,1,-2)$ and $\mathbf{p} = (0,0,0)$. The canonical loop of base point $\mathbf{p}$ and normalized wrapping vector $\mathbf{w}^*$ is the 1-loop $(\mathbf{p}, V)$ with:

$$V=\left(\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}1\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\end{pmatrix},\begin{pmatrix}0\\0\\-1\end{pmatrix},\begin{pmatrix}0\\0\\-1\end{pmatrix},\begin{pmatrix}0\\0\\-1\end{pmatrix},\begin{pmatrix}0\\0\\-1\end{pmatrix}\right)$$

Proposition 2.4.10 *Any m-loop of base point $\mathbf{p} \in \mathbb{T}^n$ and of normalized wrapping vector $\mathbf{w}^* \in \mathbb{Z}^n$ is homotopic to the canonical loop of base point $\mathbf{p}$ and of normalized wrapping vector $\mathbf{w}^*$.*

The proof of this proposition may be found in Sec. 2.4.3.

This proposition shows that the canonical loop of base point $\mathbf{p}$ and of normalized wrapping vector $\mathbf{w}^*$ can be seen as a canonical form for all loops of base point $\mathbf{p}$ and normalized wrapping vector $\mathbf{w}^*$.

From this, we deduce that two m-loops of same base point $\mathbf{p}$ and same normalized wrapping vector $\mathbf{w}^*$ are homotopic, as they both belong to the homotopy class of the canonical loop of base point $\mathbf{p}$ and of normalized wrapping vector $\mathbf{w}^*$.

Example In $\mathbb{T}^2 = \mathbb{Z}^4 \times \mathbb{Z}^4$, let $\mathcal{L}$ be the 2-loop of base point $\mathbf{p}$ represented on Fig. 9-**a**. It can be seen that the normalized wrapping vector of $\mathcal{L}$ is equal to $(1,-1)$: this means that the loop wraps 1 time around the first dimension, with a "clockwise" direction, and one time around the second dimension, with an "anti-clockwise". The canonical loop of base point $\mathbf{p}$ and of normalized wrapping vector $(1,-1)$, represented on Fig. 9-**b**, belongs to the same homotopy class as $\mathcal{L}$ (Prop. 2.4.10).

We can now state the main theorem of this article, which is a direct consequence of Prop. 2.4.8 and Prop. 2.4.10.

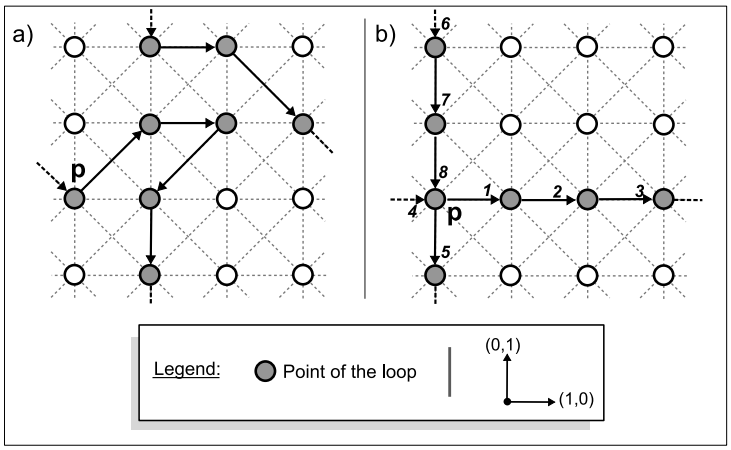


Figure 9: **Canonical loops -** In $\mathbb{T}^2 = \mathbb{Z}_4 \times \mathbb{Z}_4$, the 2-loop in **a)** has a normalized wrapping vector equal to $(1,-1)$. The 1-loop in **b)** is the canonical loop of base point $\mathbf{p}$ and normalized wrapping vector $(1,-1)$. On **b)**, the numbers represent the positions of the 1-steps in the 1-step sequence associated to the loop.

Theorem 2.4.11 *Two m-loops of $\mathbb{T}^n$ of same base point are homotopic if and only if their wrapping vectors are equal.*

Remark According to Th. 2.4.11, the homotopy class of the trivial loop $(\mathbf{p}, ())$ is the set of all m-loops of base point $\mathbf{p}$ that have a null wrapping vector.

Thus, the loop depicted on Fig. 8-**b** belongs to the homotopy class of the trivial loop.

2.4.2.3 WRAPPING VECTOR AND FUNDAMENTAL GROUP

Given a point $\mathbf{p} \in \mathbb{T}^n$, we set $\Omega = \{\mathbf{w}^* \in \mathbb{Z}^n / \textit{ there exists an m-loop in } \mathbb{T}^n \textit{ of base point } \mathbf{p} \textit{ and of normalized wrapping vector } \mathbf{w}^*\}$ (see Def. 2.4.7). From Prop. 2.4.6, it is plain that $\Omega = \mathbb{Z}^n$. Therefore, $(\Omega, +)$ is precisely $(\mathbb{Z}^n, +)$

Theorem 2.4.11 states that there exists a bijection between the set of the homotopy classes of all m-loops of base point $\mathbf{p}$ and Ω. The product (see Sec. 2.3.2) of two m-loops $\mathcal{K}$ and $\mathcal{L}$ of same base point $\mathbf{p}$ and of respective wrapping vectors $\mathbf{w_k}$ and $\mathbf{w_l}$ is the loop $(\mathcal{K}.\mathcal{L})$ of base point $\mathbf{p}$ and of wrapping vector $(\mathbf{w_k} + \mathbf{w_l})$. Therefore we can state that there exists an isomorphism between the fundamental group of $\mathbb{T}^n$ and $(\Omega, +)$.

Consequently, we retrieve in our discrete framework a well-known property of the fundamental group of toric spaces [Hato2].

Proposition 2.4.12 *The fundamental group of $\mathbb{T}^n$ is isomorphic to $(\mathbb{Z}^n, +)$.*

2.4.2.4 PROOF OF ALGORITHM WSD

Proof *(of Prop. 2.4.3)* For all $\mathbf{y} \in X$ such that $\mathbf{y} \neq \mathbf{p}$, there exists a point $\mathbf{x}$ such that the test performed on l. 14 of Alg. 1 is true: we call $\mathbf{x}$ *the label predecessor of* $\mathbf{y}$.

• At the end of the execution of Alg. 1, if the set B is empty, then the test performed l. 10 was never true. Let $\mathcal{L} = (\mathbf{p}, V)$ be an m-loop contained in X, with $V = (\mathbf{v_1}, ..., \mathbf{v_k})$, and let us denote by $\mathbf{x_i}$ the i-th point of $\mathcal{L}$. As the test performed l. 10 was always false, we have the following:

$$\begin{cases} \text{for all } i \in [1; k-1], \mathbf{v_i} = \text{Coord}(\mathbf{x_{i+1}}) - \text{Coord}(\mathbf{x_i}) \\ \mathbf{v_k} = \text{Coord}(\mathbf{x_1}) - \text{Coord}(\mathbf{x_k}) \end{cases}$$

The wrapping vector of $\mathcal{L}$ is

$$\begin{aligned}\mathbf{w} &= \sum_{i=1}^{k-1}(\mathrm{Coord}(\mathbf{x_{i+1}}) - \mathrm{Coord}(\mathbf{x_i})) \\ &\quad + \mathrm{Coord}(\mathbf{x_1}) - \mathrm{Coord}(\mathbf{x_k}) = \mathbf{0}\end{aligned}$$

Thus, if the algorithm returns false, each m-loop of X has a null wrapping vector and, according to Th. 2.4.11, belongs to the homotopy class of a trivial loop: there is no toric m-loop in X which is therefore not wrapped in $\mathbb{T}^n$.

• If B is not empty, then, there exists $\mathbf{x}, \mathbf{y} \in X$ and an m-step $\mathbf{a}$ such that $\mathbf{x} \oplus \mathbf{a} = \mathbf{y}$ and $\mathrm{Coord}(\mathbf{y}) - \mathrm{Coord}(\mathbf{x}) \neq \mathbf{a}$.

It is therefore possible to find two sequences γ_x and γ_y of m-adjacent points in X, with $\gamma_x = (\mathbf{p} = \mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_q} = \mathbf{x})$ and $\gamma_y = (\mathbf{y} = \mathbf{y_t}, ..., \mathbf{y_2}, \mathbf{y_1} = \mathbf{p})$, such that, for all $i \in [1; q-1]$, $\mathbf{x_i}$ is the label predecessor of $\mathbf{x_{i+1}}$, and for all $i \in [1; t-1]$, $\mathbf{y_i}$ is the label predecessor of $\mathbf{y_{i+1}}$. Therefore, we can set

$$\begin{cases} \text{. for all } i \in [1; q-1], \mathbf{u_i} = \mathrm{Coord}(\mathbf{x_{i+1}}) - \mathrm{Coord}(\mathbf{x_i}) \\ \quad \text{is an m-step such that } \mathbf{x_i} \oplus \mathbf{u_i} = \mathbf{x_{i+1}} \\ \text{. for all } i \in [1; t-1], \mathbf{v_i} = \mathrm{Coord}(\mathbf{y_i}) - \mathrm{Coord}(\mathbf{y_{i+1}}) \\ \quad \text{is an m-step such that } \mathbf{y_{i+1}} \oplus \mathbf{v_i} = \mathbf{y_i} \end{cases}$$

Let $\mathcal{N} = (\mathbf{p}, V)$ be the m-loop such that $V = (\mathbf{u_1}, ..., \mathbf{u_{q-1}}, \mathbf{a}, \mathbf{v_{t-1}}, ..., \mathbf{v_1})$. The m-loop $\mathcal{N}$ is lying in X and its wrapping vector $\mathbf{w}$ is equal to:

$$\begin{aligned}\mathbf{w} &= \sum_{i=1}^{q-1}\mathbf{u_i} + \mathbf{a} + \sum_{i=1}^{t-1}\mathbf{v_i} = \mathbf{a} - (\mathrm{Coord}(\mathbf{y}) - \mathrm{Coord}(\mathbf{x})) \\ &\neq \mathbf{0}\end{aligned}$$

Thus, when the algorithm returns a non-empty set, it is possible to find, inside X, an m-loop with a non-null wrapping vector: by Th. 2.4.11, there is a toric m-loop in X which is therefore wrapped in $\mathbb{T}^n$. □

2.4.2.5 COMPUTING A BASIS FOR TORIC LOOPS IN A SUBSET OF A TORIC SPACE

In this part, we show that Alg. 1 builds a basis for all normalized wrapping vectors of all toric m-loops contained in a subset of $\mathbb{T}^n$.

Given $\mathbb{T}^n$ of size vector $\mathbf{d}$ and an m-connected subset X of $\mathbb{T}^n$, we consider having run WSD($n, m, \mathbb{T}^n, \mathbf{d}, X$), and we will use Coord, the function built on l. 15 of Alg. 1.

Given an m-step $\mathbf{v}$ and two points $\mathbf{x}, \mathbf{y} \in X$ such that $\mathbf{x} \oplus \mathbf{v} = \mathbf{y}$, *the points* $\mathbf{x}$ *and* $\mathbf{y}$ *are conflictive through* $\mathbf{v}$ if $\mathrm{Coord}(\mathbf{x}) + \mathbf{v} \neq \mathrm{Coord}(\mathbf{y})$. Observe that, for all conflictive pairs of points x, y through v contained in the subset X of $\mathbb{T}^n$, the vector $((\mathrm{Coord}(\mathbf{x}) + \mathbf{v} - \mathrm{Coord}(\mathbf{y}))/\mathbf{d})$ is added to the set B built on l. 11 of Alg. 1.

The next lemma establishes that, in order to calculate the wrapping vector of an m-loop (and therefore, its homotopy class, as stated by Th. 2.4.11), only the conflictive pairs of points in the loop need to be considered:

Lemma 2.4.13 *Let $\mathbf{p} \in X$ and let $\mathcal{K} = (\mathbf{p}, V)$ be an m-loop in X, with $V = (\mathbf{v_1}, ..., \mathbf{v_k})$. For all $i \in [1; k+1]$, we denote by $\mathbf{x_i}$ the i-th point of $\mathcal{K}$, and we set $C = \{i \in [1; k] | \mathbf{x_i}$ and $\mathbf{x_{i+1}}$ are conflictive through $\mathbf{v_i}\}$. Let $\mathbf{w}$ be the wrapping vector of $\mathcal{K}$. We have:*

$$\mathbf{w} = \sum_{j \in C} (\mathrm{Coord}(\mathbf{x_j}) + \mathbf{v_j} - \mathrm{Coord}(\mathbf{x_{j+1}}))$$

Proof The wrapping vector w of $\mathcal{K}$ is by definition:

$$\begin{aligned}
\mathbf{w} &= \sum_{j=1}^{k} \mathbf{v_j} = \sum_{j \notin C} \mathbf{v_j} + \sum_{j \in C} \mathbf{v_j} \\
&= \sum_{j \notin C} (\mathrm{Coord}(\mathbf{x_{j+1}}) - \mathrm{Coord}(\mathbf{x_j})) + \sum_{j \in C} \mathbf{v_j} \\
&= \sum_{j=1}^{k} (\mathrm{Coord}(\mathbf{x_{j+1}}) - \mathrm{Coord}(\mathbf{x_j})) \\
&- \sum_{j \in C} (\mathrm{Coord}(\mathbf{x_{j+1}}) - \mathrm{Coord}(\mathbf{x_j})) + \sum_{j \in C} \mathbf{v_j}
\end{aligned}$$

As $\sum_{j=1}^{k} (\mathrm{Coord}(\mathbf{x_{j+1}}) - \mathrm{Coord}(\mathbf{x_j}))) = \mathrm{Coord}(\mathbf{x_{k+1}}) - \mathrm{Coord}(\mathbf{x_1}) = 0$, we get the lemma proved. □

We now focus on the set B, result of WSD(n, m, $\mathbb{T}^n$, **d**, X). For all **x**, **y** ∈ X that are conflictive through an m-step **v**, the vector $((\mathrm{Coord}(\mathbf{x}) + \mathbf{v} - \mathrm{Coord}(\mathbf{y}))/\mathbf{d})$ is in B. The next proposition states that B can be seen as a generating set for all (normalized) wrapping vectors of all m-loops of X.

Proposition 2.4.14 *Let the set $B = \{\mathbf{w_1}, ..., \mathbf{w_k}\}$ be the result of WSD(n,m, $\mathbb{T}^n$,**d**, X). A vector $\mathbf{w}^* \in \mathbb{Z}^n$ is the normalized wrapping vector of an m-loop of X if and only if there exists k non-negative integers $\alpha_1, ..., \alpha_k$ such that*

$$\mathbf{w}^* = \sum_{i=1}^{k} \alpha_i . \mathbf{w_i} \tag{2.1}$$

Remark If **x** and **y** are conflictive through **v**, then **y** and **x** are conflictive through (−**v**): therefore, if **u** belongs to B, then −**u** also belongs to B. This is why it is possible, in Prop. 2.4.14, to restrain the choice of the coefficients $\alpha_1, ..., \alpha_k$ to the set of non-negative integers.

Proof If $\mathcal{L}$ is an m-loop in X of normalized wrapping vector $\mathbf{w}^*$, then, by Lem. 2.4.13 and by construction of B, we deduce that $\mathbf{w}^*$ satisfies Equ. 2.1.

Now, let $\mathbf{w}^*$ be a vector which satisfies Equ. 2.1. For each **b** ∈ B, there exists **x** and **y** in X and an m-step **a** such that **x** and **y** are conflictive through **a** and such that $\mathbf{b} = (\mathrm{Coord}(\mathbf{x}) + \mathbf{a} - \mathrm{Coord}(\mathbf{y}))/(\mathbf{d})$. Consider the m-loop $\mathcal{N}$ (see the second part of proof of Prop. 2.4.3), lying inside X, and whose wrapping vector is equal to $(\mathrm{Coord}(\mathbf{x}) + \mathbf{a} - \mathrm{Coord}(\mathbf{y}))$: the normalized wrapping vector of $\mathcal{N}$ is **b**.

Therefore, for each **b** ∈ B, there exists an m-loop $\mathcal{L}_\mathbf{b}$ inside X, whose normalized wrapping vector is equal to **b**. Let $\mathcal{L}^* = \prod_{i=1}^{k} (\mathcal{L}_{\mathbf{w_i}})^{\alpha_i}$. By construction, $\mathcal{L}^*$ is contained in X, and its wrapping vector is equal to $\mathbf{w}^*$. □

Thus, algorithm 1 builds a (non-minimal) basis allowing to compute the normalized wrapping vector of any m-loop of X: the normalized wrapping vector of any m-loop lying inside X is the linear combination of elements of B. The set B, result of Alg. 1, allows to get information on how X wraps inside the toric space.

2.4.3 ADDITIONAL PROOFS AND LEMMAS

In the following, we give some proofs and lemmas that were useful for establishing proof of Alg. 1. We decided to move these proofs in this section in order to keep the reading of section 2.4 fluent.

Proof *of Prop. 2.4.6* - First, let $\mathcal{L} = (\mathbf{p}, V)$ be an m-loop of wrapping vector $\mathbf{w} = (w_1, ..., w_n)$, with $\mathbf{p} = (p_1, ..., p_n)$. As $\mathcal{L}$ is a loop, for all $i \in [1; n]$, $p_i \oplus_{d_i} w_i = p_i$. Hence, for all $i \in [1; n]$, $w_i \equiv 0 (mod\ d_i)$, proving that w_i is a multiple of d_i for all $i \in [1; n]$.

Let $\mathbf{w} = (w_1, ..., w_n)$ be a vector of $\mathbb{Z}^n$ such that for all $i \in [1; n]$, w_i is a multiple of d_i. If we denote by $(\mathbf{p}, B_i)$ the i-th basic loop of base point $\mathbf{p}$, we see that $(\prod_{i=1}^{n} (\mathbf{p}, B_i)^{w_i/d_i})$ is an m-loop whose wrapping vector is equal to $\mathbf{w}$. □

Lemma 2.4.15 *Any m-loop $\mathcal{L} = (\mathbf{p}, V)$ is homotopic to a 1-loop.*

Proof Let us write $V = (\mathbf{v_1}, ..., \mathbf{v_k})$ and let $j \in [1; n]$ be such that $\mathbf{v_j}$ is not a 1-step. The m-loop $\mathcal{L}$ is directly homotopic to $\mathcal{L}_1 = (\mathbf{p}, V_1)$, with $V_1 = (\mathbf{v_1}, ..., \mathbf{v_{j-1}}, \mathbf{v_j}, \mathbf{0}, \mathbf{v_{j+1}}, ..., \mathbf{v_k})$. As $\mathbf{v_j}$ is not a 1-step, there exists an (m-1)-step $\mathbf{v'_j}$ and a 1-step $\mathbf{v_{j_1}}$ such that $\mathbf{v_j} = (\mathbf{v_{j_1}} + \mathbf{v'_j})$. The m-loop $\mathcal{L}_1$ is directly homotopic to $\mathcal{L}_2 = (\mathbf{p}, V_2)$, with $V_2 = (\mathbf{v_1}, ..., \mathbf{v_{j-1}}, \mathbf{v_{j_1}}, \mathbf{v'_j}, \mathbf{v_{j+1}}, ..., \mathbf{v_k})$. By iteration, it is shown that $\mathcal{L}$ is homotopic to a 1-loop. □

Lemma 2.4.16 *Let $\mathcal{L}_A = (\mathbf{p}, V_A)$ and $\mathcal{L}_B = (\mathbf{p}, V_B)$ be two m-loops such that $V_A = (\mathbf{v_1}, ..., \mathbf{v_{j-1}}, \mathbf{v_{j_1}}, \mathbf{v_{j_2}}, \mathbf{v_{j+1}}, ..., \mathbf{v_k})$ and $V_B = (\mathbf{v_1}, ..., \mathbf{v_{j-1}}, \mathbf{v_{j_2}}, \mathbf{v_{j_1}}, \mathbf{v_{j+1}}, ..., \mathbf{v_k})$ where $\mathbf{v_{j_1}}$ and $\mathbf{v_{j_2}}$ are 1-steps. Then, $\mathcal{L}_A$ and $\mathcal{L}_B$ are homotopic.*

Proof As $\mathbf{v_{j_1}}$ and $\mathbf{v_{j_2}}$ are 1-steps, they have at most one non-null coordinate. If $(\mathbf{v_{j_1}} - \mathbf{v_{j_2}})$ is an n-step, the two loops are directly homotopic. If $(\mathbf{v_{j_1}} - \mathbf{v_{j_2}})$ is not an n-step, then necessarily $\mathbf{v_{j_1}} = (-\mathbf{v_{j_2}})$. Therefore, $\mathcal{L}_A$ is directly homotopic to $\mathcal{L}_C = (\mathbf{p}, V_C)$, with $V_C = (\mathbf{v_1}, ..., \mathbf{v_{j-1}}, \mathbf{0}, \mathbf{0}, \mathbf{v_{j+1}}, ..., \mathbf{v_k})$. Furthermore, $\mathcal{L}_C$ is also directly homotopic to $\mathcal{L}_B$. □

Proof *of Prop. 2.4.10* - Let $\mathbf{a}$ and $\mathbf{b}$ be two non-null 1-steps. Let i (resp. j) be the index of the non-null coordinate of $\mathbf{a}$ (resp $\mathbf{b}$). We say that $\mathbf{a}$ *is index-smaller than* $\mathbf{b}$ if $i < j$.

Let $\mathcal{L} = (\mathbf{p}, V)$ be an m-loop of normalized wrapping vector $\mathbf{w}^* \in \mathbb{Z}^n$.

- **1 -** The m-loop $\mathcal{L}$ is homotopic to a 1-loop $\mathcal{L}_1 = (\mathbf{p}, V_1)$ (see Lem. 2.4.15).
- **2 -** By Def. 2.3.1 and 2.3.2, the 1-loop $\mathcal{L}_1$ is homotopic to a 1-loop $\mathcal{L}_2 = (\mathbf{p}, V_2)$, where V_2 contains no null vector.
- **3 -** Let $\mathcal{L}_3 = (\mathbf{p}, V_3)$ be such that V_3 is obtained by iteratively permuting all pairs of consecutive 1-steps $(\mathbf{v_j}, \mathbf{v_{j+1}})$ in V_2 such that $\mathbf{v_{j+1}}$ is index-smaller than $\mathbf{v_j}$. Thanks to Lem. 2.4.16, $\mathcal{L}_3$ is homotopic to $\mathcal{L}_2$.

- **4 -** Consider $\mathcal{L}_4 = (\mathbf{p}, V_4)$, where V_4 is obtained by iteratively replacing all pairs of consecutive 1-steps $(\mathbf{v_j}, \mathbf{v_{j+1}})$ in V_3 such that $\mathbf{v_{j+1}} = (-\mathbf{v_j})$ by two null vectors, and then removing these two null vectors. The loop $\mathcal{L}_4$ is homotopic to $\mathcal{L}_3$ (see Def. 2.3.1).

The 1-loop $\mathcal{L}_4$ is homotopic to $\mathcal{L}$, it has therefore the same normalized wrapping vector $\mathbf{w}^* = (w_1^*, ..., w_n^*)$ (see Prop. 2.4.8). By construction, each pair of consecutive 1-steps $(\mathbf{v_j}, \mathbf{v_{j+1}})$ of V_4 is such that $\mathbf{v_j}$ and $\mathbf{v_{j+1}}$ are non-null and either $\mathbf{v_j} = \mathbf{v_{j+1}}$ or $\mathbf{v_j}$ is index-smaller than $\mathbf{v_{j+1}}$.

Let $\mathbf{d} = (d_1, ..., d_n)$ be the size vector of $\mathbb{T}^n$. As the normalized wrapping vector of $\mathcal{L}_4$ is equal to $\mathbf{w}^*$, we deduce that the $(d_1.|w_1^*|)$ first elements of V_4 are equal to $(\frac{w_1^*}{|w_1^*|}.\mathbf{b_1})$ (see Def. 2.4.5). Moreover, the $(d_2.|w_2^*|)$ next elements are equal to $(\frac{w_2^*}{|w_2^*|}.\mathbf{b_2})$, etc. Therefore, we have $\mathcal{L}_4 = (\prod_{i=1}^{n} (\mathbf{p}, B_i)^{w_i^*})$. □

2.5 COMPARING LOOP HOMOTOPY AND LOOP EQUIVALENCE IN z^2 AND z^3

We propose an adaptation of our definition of loop homotopy to $\mathbb{Z}^2$ and $\mathbb{Z}^3$, and we show that the resulting definition is equivalent to the definition of loop equivalence given in [Kon89]. In the following definition, a loop is a sequence of point $(x_1, ..., x_k)$ of $\mathbb{Z}^n$ (with $n = 2$ or $n = 3$) such that $x_1 = x_k$ and, for all $i \in [1; k]$, x_i and x_{i+1} are adjacent (the definition of a loop depends on a chosen adjacency relation).

Definition 2.5.1 *[Kon89] In $\mathbb{Z}^n$ (with $n = 2$ or $n = 3$), two loops $C_1 = (x_1, ..., x_k)$ and $C_2 = (y_1, ..., y_j)$ differ only in a set K if $k = j$, and for all $i \in [1; k]$, $p_i = q_i$ iff $q_i \notin K$, and $p_i \in K$ iff $q_i \in K$.*

In $\mathbb{Z}^n$ (with $n = 2$ or $n = 3$), two loops $C_1 = (x_1, ..., x_k)$ and $C_2 = (y_1, ..., y_j)$ are directly equivalent if

- *The loops are equal after removing all consecutive duplicate points in each loops, or*
- *The two loops only differ in a unit lattice square or a unit lattice cube.*

The previous definition is a simplified version of the original definition given in [Kon89], where the author is concerned by homotopy inside objects (this part of the definition has been removed as it irrelevant for the following study).

It is possible to adapt all definitions given previously in this thesis to $\mathbb{Z}^n$, by replacing the operation '$\oplus$' by the usual operation '$+$'. This way, we define the direct homotopy of m-loops in $\mathbb{Z}^n$: two m-loops of same base point $p \in \mathbb{Z}^n$ *are directly homotopic* if they are directly homotopic in the sense of definition 2.3.1 adapted to $\mathbb{Z}^n$.

We accordingly define the homotopy of m-loops in $\mathbb{Z}^n$: two m-loops $\mathcal{K}$ and $\mathcal{R}$ of same base point $p \in \mathbb{Z}^n$ *are homotopic* if there exists a sequence $(\mathcal{K} = \mathcal{C}_1, ..., \mathcal{C}_i = \mathcal{R})$ of m-loops such that, for all $j \in [1; i-1]$, $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ are directly homotopic.

A non self-intersecting loop is an m-loop $(\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_k}))$ such that, for all $i \in [1; k[$, and for all $j \in]i; k]$, with $(i, j) \neq (1; k)$, $\sum_{h=i}^{j} \mathbf{u_h} \neq 0$. We now introduce a lemma which will be used in the proof of the forthcoming proposition.

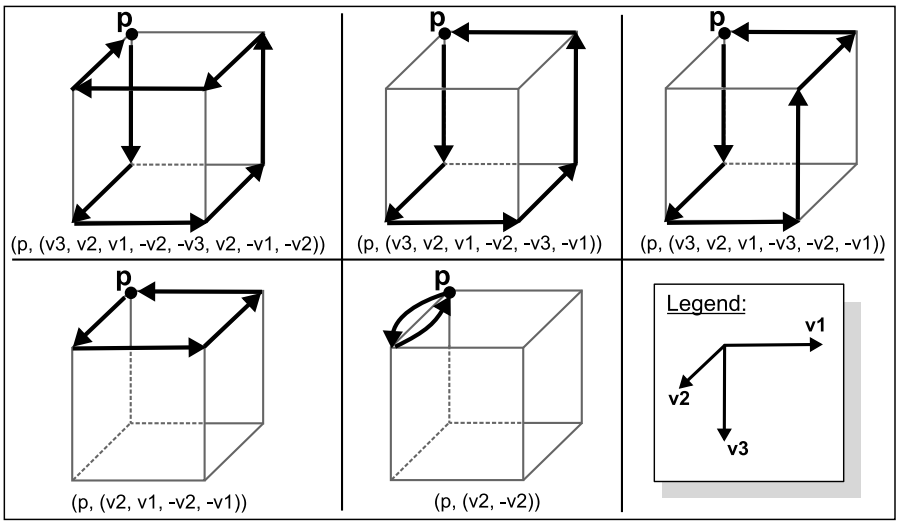


Figure 10: **Proving Lem. 2.5.2 -** In a unit lattice cube, when considering all possible symmetries, only 5 different non self-intersecting 1-loops can be built. If the cube does not contain two diametrically opposite points not belonging to the loop, the 1-loops are all equivalent to a trivial loop.

Lemma 2.5.2 *In $\mathbb{Z}^3$, any non self-intersecting 1-loop contained in a unit lattice cube which does not contain two diametrically opposite points not belonging to the loop, is homotopic to a trivial loop.*

Proof A non self-intersecting 1-loop $(\mathbf{p}, U)$ holding inside a unit lattice cube is such that $|U| \in \{0, 2, 4, 6, 8\}$.

When considering all possible symmetries and rotations in the unit lattice cube of $\mathbb{Z}^3$, only 5 kinds of non self-intersecting (and non trivial) 1-loops can be built (as shown on Fig. 10). For example, only one kind of 1-loop composed of eight 1-steps can be built (see Fig. 10a): let us call it $(\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_8}))$. It is plain that, in order to pass by each of the cube's eight vertices once and only once, we must have $\{\mathbf{u_1}, ..., \mathbf{u_8}\} = \{\mathbf{v_a}, \mathbf{v_a}, -\mathbf{v_a}, -\mathbf{v_a}, \mathbf{v_b}, -\mathbf{v_b}, \mathbf{v_c}, -\mathbf{v_c}\}$, with $\mathbf{v_a}, \mathbf{v_b}, \mathbf{v_c}$ being 1-steps of $\mathbb{Z}^3$ such that $\mathbf{v_a} \neq \pm\mathbf{v_b}$, $\mathbf{v_a} \neq \pm\mathbf{v_c}$ and $\mathbf{v_b} \neq \pm\mathbf{v_c}$. In order to avoid the loop to self-intersect, we must have $\mathbf{u_1} = \mathbf{v_a}$ or $\mathbf{u_2} = \mathbf{v_a}$.

If we choose $\mathbf{u_2} = \mathbf{v_a}$, then, in order to avoid the loop to self-intersect, we need $\mathbf{u_6} = \mathbf{v_a}$ and $\mathbf{u_4} = \mathbf{u_8} = -\mathbf{v_a}$. Then, we set $\mathbf{u_1} = \mathbf{v_b}$, and consequently, $\mathbf{u_3} = \mathbf{v_c}$, $\mathbf{u_5} = -\mathbf{v_b}$, and $\mathbf{u_7} = -\mathbf{v_c}$. Choosing $\mathbf{u_1} = \mathbf{v_a}$ leads to a symmetrical loop.

A similar reasoning, in the case $|U| = 6$, shows that the two loops $(\mathbf{p}, (\mathbf{v_a}, \mathbf{v_b}, \mathbf{v_c}, -\mathbf{v_a}, -\mathbf{v_b}, -\mathbf{v_c}))$ (Fig. 10b) and $(\mathbf{p}, (\mathbf{v_a}, \mathbf{v_b}, -\mathbf{v_a}, \mathbf{v_c}, -\mathbf{v_b}, -\mathbf{v_c}))$ (see Fig. 10c) are the only configurations of non self-intersecting loops that can be built in the unit lattice cube of $\mathbb{Z}^3$. The cases $|U| = 4$ and $|U| = 2$ are even simpler, each with one possible configuration of non-self intersecting loop: $(\mathbf{p}, (\mathbf{v_a}, \mathbf{v_b}, -\mathbf{v_a}, -\mathbf{v_b}))$ (see Fig. 10d) and $(\mathbf{p}, (\mathbf{v_a}, -\mathbf{v_a}))$ (see Fig. 10e).

The five kinds of non self-intersecting 1-loop which can exist in a unit lattice cube are represented on Fig.10. It is plain that each of these loops can be reduced to a trivial loop if the cube does not contain two diametrically opposite points not belonging to the loop. □

The next proposition establishes that, in $\mathbb{Z}^n$, m-loop homotopy and m-loop equivalence defined in [Kon89] (see Def. 2.3.2) are equivalent.

Proposition 2.5.3 *Two m-loops $\mathcal{K} = (\mathbf{p}, U)$ and $\mathcal{R} = (\mathbf{p}, V)$ in $\mathbb{Z}^n$ (with $(n, m) \in \{(2,1); (2,2); (3,1); (3,3)\}$) are equivalent if and only if they are homotopic.*

Proof In the following proof, we set $U = (\mathbf{u_1}, ..., \mathbf{u_k})$, $V = (\mathbf{v_1}, ..., \mathbf{v_r})$, and, for all $i \in [1; k+1]$, we denote by $\mathbf{x_i}$ the i-th point of $\mathcal{K}$, and for all $i \in [1; r+1]$, we denote by $\mathbf{y_i}$ the i-th point of $\mathcal{R}$. We have, for all $i \in [1; k]$, $\mathbf{u_i} = (\mathbf{x_{i+1}} - \mathbf{x_i})$ and, for all $i \in [1; r]$, $\mathbf{v_i} = (\mathbf{y_{i+1}} - \mathbf{y_i})$.

If $k \neq r$, then it can be easily seen that $\mathcal{K}$ and $\mathcal{R}$ are directly equivalent if and only if they are homotopic.

Now, let us consider the case where $k = r$. We define

$$\begin{aligned} D_K &= \{\mathbf{x} \in \mathcal{K} | \text{ there exists } i \in [2; k] \text{ such that } \mathbf{x_i} \neq \mathbf{y_i}\} \\ D_R &= \{\mathbf{y} \in \mathcal{R} | \text{ there exists } i \in [2; k] \text{ such that } \mathbf{y_i} \neq \mathbf{x_i}\}. \end{aligned}$$

Note that, for all $i \in [2; k]$, $x_i \in D_K$ if and only if $y_i \in D_R$. Thus, $(D_K \cup D_R)$ is the set of all points of $\mathcal{K}$ which differ from the corresponding point of $\mathcal{R}$, and vice versa.

i) Suppose that $\mathcal{K}$ and $\mathcal{R}$ are directly homotopic (see Def. 2.3.1, case 3), then there exists $j \in [1; k-1]$ such that $V = (\mathbf{u_1}, ..., \mathbf{u_{j-1}}, \mathbf{v_j}, \mathbf{v_{j+1}}, \mathbf{u_{j+2}}, ..., \mathbf{u_k})$, where $(u_j - v_j)$ is an n-step and $(u_j + u_{j+1} = v_j + v_{j+1})$. Obviously, we have $(D_K \cup D_R) = \{\mathbf{x_{j+1}}, \mathbf{y_{j+1}}\}$.

As $(\mathbf{x_{j+1}} - \mathbf{y_{j+1}}) = (\mathbf{u_j} - \mathbf{v_j})$, the points $\mathbf{x_{j+1}}$ and $\mathbf{y_{j+1}}$ lie in a same unit lattice square or cube. Furthermore, if $n = 3$ and $m = 1$, $(\mathbf{u_j} - \mathbf{v_j})$ is a 2-step, proving that $\mathbf{x_{j+1}}$ and $\mathbf{y_{j+1}}$ lie in a same unit lattice square (no diametrically opposite points to matter): the m-loops $\mathcal{K}$ and $\mathcal{R}$ are directly equivalent.

ii) Reciprocally, suppose that $\mathcal{K}$ and $\mathcal{R}$ are directly equivalent.

• In the case where $(n, m) \in \{(2,2), (3,3)\}$, we set, for all $h \in [1; k]$, $S_h = (\mathbf{v_1}, ..., \mathbf{v_{h-1}}, \mathbf{x_{h+1}} - \mathbf{y_h}, \mathbf{u_{h+1}}, ..., \mathbf{u_k})$ and $\mathcal{C}_h = (p, S_h)$.

First, we prove that for all $h \in [1; k]$, $\mathcal{C}_h$ is an m-loop of base point p, by proving that $(\mathbf{x_{h+1}} - \mathbf{y_h})$ is an m-step. As $\mathcal{K}$ and $\mathcal{R}$ are directly equivalent, we either have $\mathbf{x_h} = \mathbf{y_h}$ or $\mathbf{x_{h+1}} = \mathbf{y_{h+1}}$ (the result is then directly obtained), or we have $\mathbf{x_h}, \mathbf{y_h}, \mathbf{x_{h+1}}$ and $\mathbf{y_{h+1}}$ lying in a same unit lattice cube or square: $(\mathbf{x_{h+1}} - \mathbf{y_h})$ is therefore an n-step, and also an m-step since $n = m$.

We are going to prove that for all $h \in [1; k-1]$, $\mathcal{C}_h$ and $\mathcal{C}_{h+1}$ are directly homotopic by proving that they match the case 3 of Def. 2.3.1. We set $S_h = (\mathbf{a_1}, ..., \mathbf{a_k})$, and $S_{h+1} = (\mathbf{b_1}, ..., \mathbf{b_k})$:

- $S_{h+1} = (\mathbf{a_1}, ..., \mathbf{a_{h-1}}, \mathbf{b_h}, \mathbf{b_{h+1}}, \mathbf{a_{h+2}}, ..., \mathbf{a_k})$,
- $(\mathbf{a_h} + \mathbf{a_{h+1}}) = \mathbf{x_{h+1}} - \mathbf{y_h} + \mathbf{u_{h+1}} = \mathbf{x_{h+2}} - \mathbf{y_h}$, and $(\mathbf{b_h} + \mathbf{b_{h+1}}) = \mathbf{v_h} + \mathbf{x_{h+2}} - \mathbf{y_{h+1}} = \mathbf{x_{h+2}} - \mathbf{y_h}$,
- $(\mathbf{a_h} - \mathbf{b_h}) = \mathbf{x_{h+1}} - \mathbf{y_h} - \mathbf{v_h} = \mathbf{x_{h+1}} - \mathbf{y_{h+1}}$ is an n-step, as either $\mathbf{x_{h+1}} = \mathbf{y_{h+1}}$ or $\mathbf{x_{h+1}}$ and $\mathbf{y_{h+1}}$ belong to a same unit lattice cube or square, and also an m-step since $n = m$.

Finally, by pointing out that $\mathcal{C}_1$ is equal to $\mathcal{K}$ and that $\mathcal{C}_k$ is equal to $\mathcal{R}$, we conclude that $\mathcal{K}$ and $\mathcal{R}$ are homotopic.

• In the case where $(n, m) = (3, 1)$, let us assume that the set D_K (resp. D_R) contains only consecutive points of the loop $\mathcal{K}$ (resp. $\mathcal{R}$): if it was not the case, the following

reasoning could still be performed on each consecutive elements of D_K and D_R in order to obtain the same result.

Thus, there exists $i \in [2; k]$ and $j \in [i; k]$ such that $(D_K \cup D_R) = \{\mathbf{x_i}, ..., \mathbf{x_j}, \mathbf{y_i}, ..., \mathbf{y_j}\}$ is included in a unit lattice square or a unit lattice cube which does not contain two diametrically opposite points not belonging to $(D_K \cup D_R)$. Therefore, we have $V = (\mathbf{u_1}, ..., \mathbf{u_{i-2}}, \mathbf{v_{i-1}}, ..., \mathbf{v_j}, \mathbf{u_{j+1}}, ..., \mathbf{u_k})$. It is possible to simplify the problem in two ways:

- As $m = 1$, $\mathbf{y_i} - \mathbf{x_{i-1}}$ and $\mathbf{x_i} - \mathbf{x_{i-1}}$ are 1-steps. Therefore, $\mathbf{x_{i-1}}, \mathbf{x_i}$ and $\mathbf{y_i}$ are in a same unit lattice square and, as $\mathbf{x_i} \neq \mathbf{y_i}$, we find that $\mathbf{x_{i-1}}$ lie in the same unit lattice cube or square than the elements of $(D_K \cup D_R)$. The same way, we prove that $\mathbf{x_{j+1}}$ lie in the same unit lattice cube or square than the elements of $(D_K \cup D_R)$.

  It may be seen that $\mathcal{K}$ is homotopic to the 1-loop $\mathcal{K}' = (\mathbf{p}, (\mathbf{u_1}, ..., \mathbf{u_j}, -\mathbf{v_j}, ..., -\mathbf{v_{i-1}}, \mathbf{v_{i-1}}, ..., \mathbf{v_j}, \mathbf{u_{j+1}}, ..., \mathbf{u_k}))$.

  Hence, proving that $\mathcal{K}'$ and $\mathcal{R}$ are homotopic can be achieved by proving that the 1-loop $(\mathbf{x_{i-1}}, (\mathbf{u_{i-1}}, ..., \mathbf{u_j}, -\mathbf{v_j}, ..., -\mathbf{v_{i-1}}))$, whose points are contained inside the same unit lattice cube or square than $(D_K \cup D_R)$, is homotopic to the trivial loop $(\mathbf{x_{i-1}}, ())$.

- Let $\mathcal{C} = (\mathbf{p}, (\mathbf{w_1}, ..., \mathbf{w_i}, ..., \mathbf{w_j}, ..., \mathbf{w_k}))$ be a self intersecting 1-loop such that $\mathbf{p} + \mathbf{w_1} + ... + \mathbf{w_i} = \mathbf{p} + \mathbf{w_1} + ... + \mathbf{w_j}$. The problem of showing that $\mathcal{C}$ is homotopic to $(\mathbf{p}, ())$ can be decomposed into two smaller problems: to prove that $\mathcal{C}' = (\mathbf{p} + \mathbf{w_1} + ... + \mathbf{w_i}, (\mathbf{w_{i+1}}, ..., \mathbf{w_j}))$ is homotopic to $(\mathbf{p} + \mathbf{w_1} + ... + \mathbf{w_i}, ())$, and then to prove that $\mathcal{C}'' = (\mathbf{p}, (\mathbf{w_1}, ..., \mathbf{w_i}, \mathbf{w_{j+1}}, ..., \mathbf{w_k}))$ is homotopic to $(\mathbf{p}, ())$. Therefore, in order to prove that a 1-loop is homotopic to a trivial loop, we can consider only, without loss of generality, non self-intersecting 1-loops.

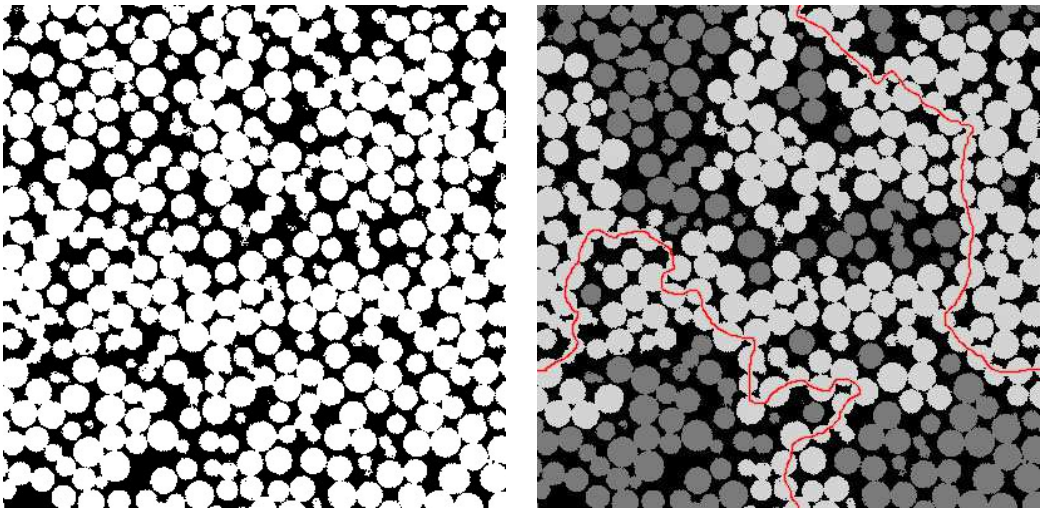
Therefore, in order to prove that the two 1-loops $\mathcal{K}$ and $\mathcal{L}$ are homotopic, it is sufficient to prove that any non self-intersecting 1-loop, contained in a unit lattice cube which does not contain two diametrically opposite points not belonging to the loop, is homotopic to a trivial loop: this is established by Lem. 2.5.2.

As the case $(n, m) = (2, 1)$ is included in the case $(n, m) = (3, 1)$, it can be concluded that $\mathcal{K}$ and $\mathcal{R}$ are homotopic. □

2.6 RESULTS AND CONCLUSION

In this chapter, we gave a formal definition of loops and homotopy, which suits all dimensions, inside discrete toric spaces in order to define various notions such as the fundamental group and the wrapping vector. Moreover, we show that wrapping vectors completely characterize toric loops (see Th. 2.4.11) and lead to a linear time algorithm for the detection of such loops in a subset X of $\mathbb{T}^n$. In addition, this algorithm allows to build, for each subset X of $\mathbb{T}^n$, a basis of vectors which characterizes all toric loops contained in X and describes how X wraps around $\mathbb{T}^n$.

In Sec. 2.1, we have seen that detecting toric loops is important in order to filter grains from a material's sample and perform a fluid flow simulation on the sample. The WSD algorithm proposed in this article detects which subsets of a sample, embedded inside a toric space, will create grains and should be removed. The WSD algorithm can also be used to validate the sample extracted from the material and used for the simulation: if

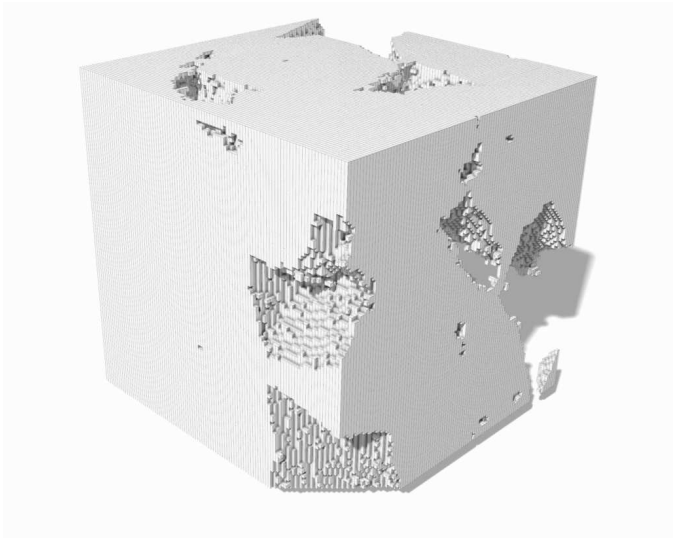


a) b)

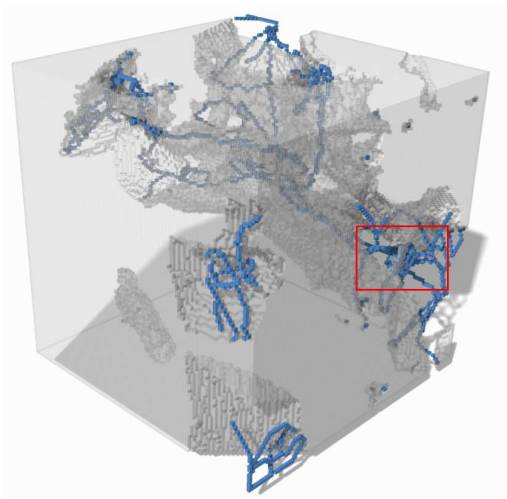
Figure 11: **Results of WSD algorithm in 2d -** The WSD algorithm is used on **a**, where it detects a toric loop for one connected component, highlighted in **b**.

an important portion of the sample is classified as grain, then deleting the grain would remove a big portion of material. In this case, one can say that the sample was not correctly chosen, and another sample should be taken.

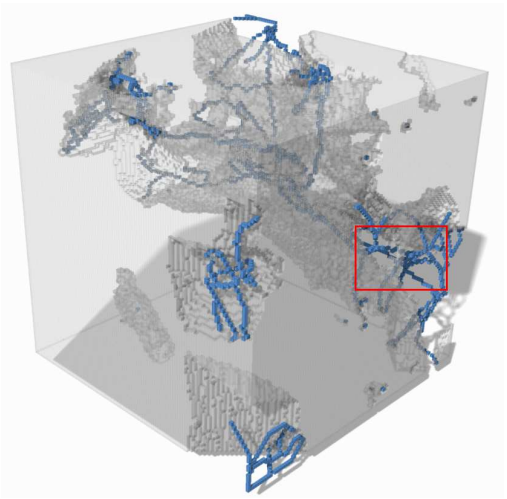
An example of toric loops detection, in 2d, is shown on Fig. 11: the WSD algorithm is used on Fig. 11a, and detects a connected component containing a toric loop (highlighted on Fig. 11). On Fig. 12, we show how, in 3d, toric loops detection allows to obtain a surface-free skeleton of the porous space of a material.



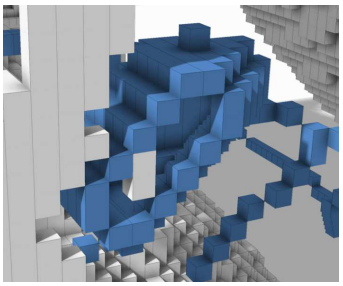
a)



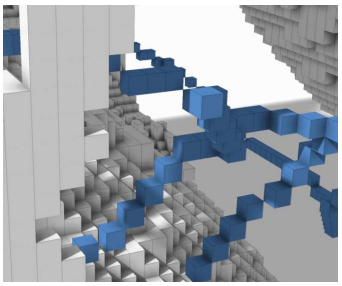
b)



c)



d)



e)

Figure 12: **Results of WSD algorithm in 3d -** On **a**, we consider a sample of porous material, embedded in a toric space. The toric skeleton of the porous space contains a surface patch, as shown on **b** (the red square shows the zone where the surface appears, a close-up of this zone is displayed on **d**). After removing components of the material which do not contain a toric cycle, the toric skeleton of the filtered porous space does not contain any surface patch, as shown on **c** (the red square shows the zone where the surface appeared before, a close-up of this zone is displayed on **e**).

Part II

SKELETON IN THE VOXEL FRAMEWORK

3

INTRODUCTION TO THE DIGITAL TOPOLOGY FRAMEWORK

In this chapter, we give a global overview of the state of the art of skeletonization in the digital topology framework (also called, sometimes, voxel framework), at the time when this thesis was written. We talk about skeleton computation, skeleton analysis and preservation of the visual aspect (medial axes, ...) of the input during skeletonization.

Contents

3.1 THINNING METHODOLOGIES

The skeletons were originally defined by Blum ([Blu62]) through an analogy with a grassfire. Imagine an object as made of grass; if you set on fire the contour of the object, then the meeting points of the flame fronts would constitute the skeleton of the object. In the continuous framework, this definition is equivalent to saying that the skeleton if the set of points which are centers of maximal balls (balls included in the object, and not strictly included in any other such ball - see Sec. 9.1 p.204) ([Cal65]).

In 1969, Hilditch gave four properties that a skeleton in a bidimensional space should possess ([Hil69]). Adapted to the general case of n-dimensional skeletons, these properties are that a skeleton should be homotopic to the original object, it should be thin (in an n-dimensional space, the skeleton of an object should be at most $(n-1)$-dimensional), it should be centered in the original object, and skeletonizing a skeleton should not change anything.

In the continuous framework, the set of centers of maximal balls, called the *medial axis*, holds these properties [Lie03]. In the discrete framework $\mathbb{Z}^n$, the discrete medial axis does not hold two of these properties: it is not always homotopic to the original object, and it is not always thin (see, for example, Fig. 30b).

Various methods have been developed for performing skeletonization of a discrete object, and we will have an overview of these methods in this chapter. According to [Pal08], discrete skeletons can be computed using four types of methods: Voronoi-based transformations([BA92], [NSK+97], [AM97], [AL99],[AL01]), distance-based transformations ([ST96], [BNS99], [TM01]), general-field methods ([AC97], [RT02b]) and thinning. In this thesis, we will focus only on thinning methods.

3.1.1 SIMPLE POINTS IN 2D AND 3D

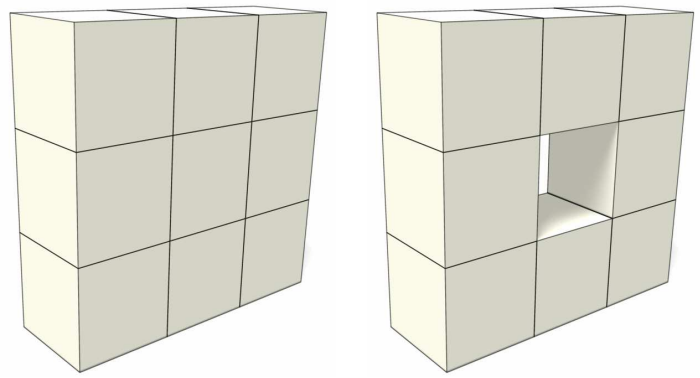
In the world of thinning, the atom is the *simple point*. Intuitively, a point is simple if it can be removed from an object without changing its topology. In the digital topology (DT) framework, the topology of an object depends on the chosen adjacency relation; for this reason, when considering a k-connected object, we will talk about k-simple points. The notion of simple point is central for homotopic thinning in the digital framework: a skeleton is obtained by removing iteratively simple points from an object.

According to [LLS92], in the 60s, 2d simple points were characterized based on connectivity: a point p is k-simple for an object X if its removal does not change the number of k-connected components of X nor the number of $\overline{k}$-connected components of $\overline{X}$ (where $\overline{k}$ is the usual adjacency for $\overline{X}$ when k is chosen as adjacency relation for X) ([BDM65], [Gol69]). This definition does not yield efficient algorithms: indeed, in order to test if a single point is simple, it requires to scan the whole object in order to enumerate its connected components. Fortunately, local characterizations of deletable points in 2d began to appear in the mid 60s, based on crossing numbers ([Rut66], [Hil69]), connectivity numbers ([YTF75]) and simplicity ([Ros70]).

All these works established that, in order to decide whether a point is deletable or not, it is only necessary to look at the configuration of the point's neighbourhood (no need to count the number of connected components of the whole object). Consequently, in 2d, deciding if a point is simple can be done in constant time.

Proposition 3.1.1 *[Ros79] Let* $X \subset \mathbb{Z}^2$, $p \in X$, *and* $N(k)$ *be the adjacency relation chosen for* X *(see Sec. 9.1, p. 204 for notations). If* $X \cap \Gamma_8^*(p)$ *has the same number of* k*-connected components as* $X \cap \Gamma_8(p)$, *and that* $\overline{X} \cap \Gamma_4^*(p) \neq \emptyset$, *then* p is simple for X.

In 3d, the removal of a point may not only change the number of connected components of the object or its complementary, but may also change the number of tunnels of the object (see, for example, Fig. 13). As in the 2d case, 3d simple points can be locally characterized ([Mor81]). Further work on 3d simple points have established only connectivity of X and $\overline{X}$ is sufficient in order to characterize 3d simple points ([MB92], [BM94], [Ber96], [SCCM94], [SC94]). As in 2d, deciding if a point is simple can be done in constant time in 3d. In order to do so, Bertrand and Malandain introduce *topological numbers* T_6 and T_{26}:



a) b)

Figure 13: **Tunnels in 3d -** In the set of nine voxels shown in **a**, removing the central voxel as shown in **b** does not change the number of connected components of the object, neither the number of connected components of its complementary, yet the topology of the object has changed: it now has a tunnel (hole).

Proposition 3.1.2 *[BM94] Let* $X \subseteq \mathbb{Z}^3$ *and* $x \in X$, *let* $T_{26}(x, X)$ *be the number of 26-connected components of* $(X \cap \Gamma_{26}^*(x))$, *and let* $T_6(x, X)$ *be the number of 6-connected components of* $(X \cap \Gamma_{18}^*(x))$.

In 26-connectivity, x *is simple for* X *iff* $T_{26}(x, X) = 1$ *and* $T_6(x, \overline{X}) = 1$.

In 6-connectivity, x *is simple for* X *iff* $T_6(x, X) = 1$ *and* $T_{26}(x, \overline{X}) = 1$

Studies of simple points in 4-dimensions have also been achieved, leading once more to a local characterization of such points ([Kon97] [CB09]). Thanks to these works, characterization of simple points in 4d can be done once again in constant time. Local characterization of simple points exist in higher dimension, but efficient (constant in time) methods for computing them has not yet been proposed.

3.1.2 SIMPLE SETS

In general, an object possesses more than one simple point. When a simple point is removed from an object, three events can take place: non-simple points can become simple, simple points can become non-simple, or nothing changes. Notice that removing

two or more simple points simultaneously from an object may lead in obtaining a set non homotopic to the original object (as shown, for example, on Fig 14). Parallel thinning (removing simultaneously many simple points) is possible but must be performed under certain conditions if topology is to be preserved. In the following, we will give an overview of the breakthroughs performed in the theoretical aspects of simple sets.

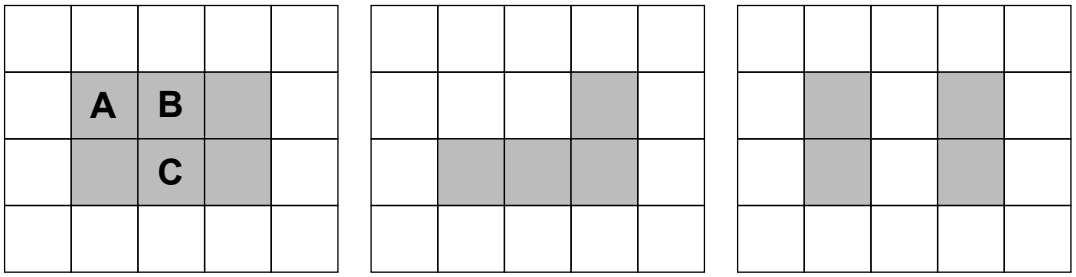


Figure 14: **Removing simple points in parallel -** On **a**, the initial shape is a rectangle made of six pixels. Pixels A, B and C are simple: removing one of them from the shape preserves the homotopy of the shape. On **b**, removing A and B at the same type preserves the homotopy of the shape. On **c**, removing A and C at the same time "breaks" in two the shape: it gives a result non-homotopic to the initial shape.

Given an object $X \subset \mathbb{R}^n$, we say that $D \subset X$ *is* k*-simple for* X if D consists only of k-simple points of X and that there exists a discrete thinning process allowing to transform X into $X \backslash D$. In 2d, this definition is equivalent to saying that X and $X \backslash D$ have the same number of connected components (with regards to the adjacency relation of X) and $\overline{X}$ and $\overline{X} \cup D$ have the same number of connected components (with regards to the adjacency relation of $\overline{X}$).

3.1.2.1 MINIMAL NON-DELETABLE SETS

Minimal non-deletable sets were introduced by C. Ronse ([Ron88]), in order to characterize under which conditions could simple points be removed simultaneously from a 2d object without "breaking" the topology. Minimal non-deletable sets rely on the notion of strong deletability, previously introduced by the same author in [Ron86].

Definition 3.1.3 *[Ron88] Let* $X \subset \mathbb{Z}^2$*, and let* k *be equal to 4 or 8. A minimal non-*k*-deletable set* $U \subseteq X$ *from* X *is a set that is not* k*-simple for* X *but every proper non void subset of* U *is* k*-simple for* X*.*

This definition is not the exact version presented in [Ron88] (for example, it does not take into account the frame of the image which is considered by Ronse). The following allows to locally characterize sets of simple points which can be removed in parallel while preserving the object's topology.

Proposition 3.1.4 *[Ron88] Let* $X \subset \mathbb{Z}^2$*, and let* k *be equal to 4 or 8. A set* $U \subseteq X$ *is a minimal non-*k*-deletable set from* X *iff one of the following holds:*

- U *consists of a single pixel that is not* k*-simple for* X *(in the sense of the* k*-connectivity).*

- U *consists of a pair of* 8*-adjacent pixels which are* k*-simple for* X*, but* U *is not* k*-simple for* X*.*
- *If* $k = 8$*,* U *consists of a triple or quadruple of pairwise* 8*-adjacent pixels, and* U *is an* 8*-connected component of* X*.*

Thanks to this last proposition, minimal non-deletable sets define "forbidden features" that should not appear in a set of pixels in order for it to be simple. Minimal non-deletable sets were designed in order to prove that 2d parallel thinning algorithms were topology preserving (and therefore valid) by testing only a small number of configurations of points. A computer-based implementation of these tests was later proposed in [Hal92], a 3d implementation of these sets was proposed in [Kon93], [Ma94] and [Kon95], and a 4d implementation was proposed in [GK03] and [KG04].

3.1.2.2 P-SIMPLE POINTS

In 1995, Bertrand introduced the P-simple points in order to characterize, in 3d, which simple points could be removed simultaneously ([Ber95b]). To do so, given $X \subset \mathbb{Z}^3$ and $x \in X$, he sets the *geodesic* n*-neighbourhood of order* k *of* x *inside* X (n being equal to 6 or 26) as the set $\Gamma_n^k(x, X) = \cup\{\Gamma_n(y) \cap \Gamma_{26}^*(x) \cap X | y \in \Gamma_n^{k-1}(x, X)\}$, with $\Gamma_n^1(x, X) = \Gamma_n^*(x) \cap X$.

Definition 3.1.5 *[Ber95c] Let* $X \in \mathbb{Z}^3$*,* $P \subset X$*,* $x \in P$ *and* n *equals to 26 or 6. The point* x *is* P_n*-simple if, for all* $S \subset (P \backslash \{x\})$*,* x *is* n*-simple for* $X \backslash S$*.*

Let $S_n(P)$ *be the set of all* P_n*-simple points. A set* D *is* P_n*-simple if* $D \subset S_n(P)$*.*

Based on this definition, one can see that, given the definition of P-simple points, if a set D is P-simple, then X and $X \backslash D$ are homotopic. The P-simple points allow to define sets of points that can be removed at once from an object during homotopic thinning. Let $G_6(x, X) = \Gamma_6^2(x, X)$ and $G_{26}(x, X) = \Gamma_{26}^1(x, X)$, the next proposition allows to locally characterize P-simple points:

Proposition 3.1.6 *[Ber95c] Let* $X \subset \mathbb{Z}^3$*,* $P \subset X$*,* $x \in P$*,* n *be equal to 6 or 26 and* $\overline{n}$ *be equal to* $32 - n$*.*

$$x \text{ is } P_n\text{-simple iff} \begin{cases} \text{The number of } n\text{-connected components of } G_n(x, X \backslash P) \text{ is equal to 1, and} \\ \text{The number of } \overline{n}\text{-connected components of } G_{\overline{n}}(x, \overline{X}) \text{ is equal to 1, and} \\ \text{For all } y \in \Gamma_n^*(x) \cap P, \Gamma_n^*(y) \cap G_n(x, X \backslash P) \text{ is not void, and} \\ \text{For all } y \in \Gamma_{\overline{n}}^*(x) \cap P, \Gamma_{\overline{n}}^*(y) \cap G_{\overline{n}}(x, \overline{X}) \text{ is not void} \end{cases}$$

As with the minimal non-deletable sets, the P-simple points allow to check if existing parallel 3d thinning algorithms work: indeed, in [Ber95c], the author gives a method for checking, based on the P-simple points framework, the topological validity of thinning algorithms. Moreover, P-simple points were widely used in order to propose new parallel 3d thinning algorithms (an example of a new algorithm is given in [Ber95c]).

3.1.2.3 CRITICAL KERNELS

Critical kernels, introduced in [Ber07], constitute a new framework for performing parallel thinning in 2d, 3d and 4d (see [BC06] and [BC08]). In relation with the DT framework, a new definition of simple points in 2d, 3d and 4d (see [CB09]) has been proposed, and links between this framework, P-simple points and minimal non-deletable

sets were established in [CB08]. Critical kernels were also used to prove that some thinning algorithms were valid, while others were not correct ([Cou06]).

Although they have applications in the voxel framework, the critical kernels rely on the cubical complex framework which is presented in Sec. 5.2.

3.1.3 SIMPLE PAIRS AND NON-SIMPLE POINTS

In order to conclude this overview of simple points, let us quickly talk of recent developments showing that simple points are not the only interesting elements in homotopic thinning. Recent work from Passat et al. ([PCB08]) is based on cubical complex framework in order to exhibit a new concept called Minimal Simple Sets, an example of which is the simple pair: in such pair of points, none of the point is simple but the pair itself is simple and can be removed without changing topology of the object.

Other works have shown that, in 3d, some points are not simple (relying on the local characterization given previously) but can still be removed without changing the topology of the input [BCP09] [Mor81].

3.1.4 SIMPLE POINTS AND MULTI-LABEL IMAGES

In some applications, an object can be decomposed in multiple parts: each voxel of the object has a label depending on the part to which it belongs. Recent work [DDL09] proposed to define *multi label simple points*: such a point can be switched from one label to some other label, without changing the topology of the different parts of the object.

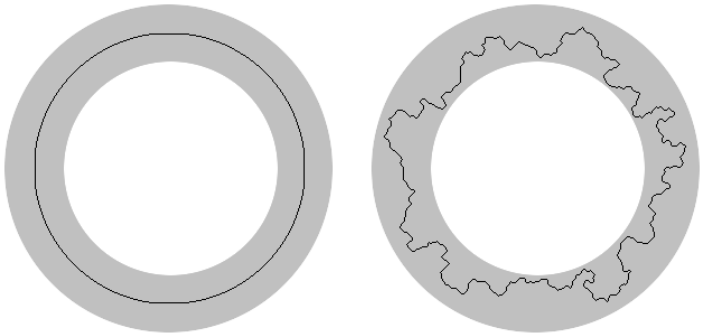
3.2 THINNING PROCESS: SIMPLE POINTS REMOVAL

Homotopic thinning in the digital framework consists of removing simple points from an object, until either no more simple point can be found, or a satisfactory subset of voxels has been reached: this method will reduce a ring into a circle, or a ball into a single point.

Two main strategies are possible for removing simple points: sequential removal and parallel removal. In the following, when giving pseudo-code for algorithms or definitions, unless the contrary is explicitly said, the objects will be considered as 8-connected in 2d, and 26-connected in 3d.

3.2.1 SEQUENTIAL REMOVAL OF SIMPLE POINTS

Sequential removal of simple points can be achieved by iteratively detecting a simple point in an object, and removing it, until no more simple point can be found. Such basic strategy does not guarantee the result to be centered in the original object, and does not preserve the "visual characteristics" of the object during thinning. It is important, when designing a sequential thinning algorithm, to decide of a removal order of simple points, and of a strategy for preserving interesting visual features of the object (this last part is studied in Sec. 3.3).



a) b)

Figure 15: **Various removal order of simple points** produce various skeletons (shown in black): on the left, the simple points were removed from the borders of the shape (shown in grey) towards the inside, while on the right, they were removed randomly.

In order to obtain a centered skeleton, one must define a particular order of removal of simple points (generally, choosing randomly the simple points to remove does not yield good results, as shown on Fig. 15). Usually, in order to get a centered skeleton, it is necessary to delete simple points "layer by layer", from the outer layer to the most inner one. Many strategies have been proposed in 2d for deciding of a removal order (a very exhaustive survey of thinning methods in 2d before 1992 can be found in [LLS92]) : for example, in the 80s, it was proposed to follow an object's contour in order to find and remove simple points "layer by layer" ([Arc81], [Pav80]). This strategy was also used in 3d in [SU79]. A typical thinning algorithm consists of looking at all simple points situated at the border of the object (see Sec. 9.1 p.204), and remove them. Taking as input an object X, this basic thinning scheme is described on Alg. 2.

Algorithm 2: Basic Thinning(X)

```
S = {p ∈ X|p belongs to the border of X}
while S ≠ ∅ do
    foreach p ∈ S do
        S = S\{p}
        if p is still simple for X then
            X = X\{p}
        end
    end
    S = {p ∈ X|p belongs to the border of X}
end
```

One can see that the order of removal of points p from the input X is important: once an element p of S has been removed from X, another element of S might no more be simple: this is called an order-dependent thinning algorithm. Sequential thinning is usually order-dependent, as it implies removing simple points one after the other, and as one removal might prevent another removal.

Order-independent thinning has been proposed in [RS02] and [KNP09]. In the first work, although the authors actually proposed a fully parallel thinning algorithm, they introduce the basic definitions of order-independency. An order-independent thinning

algorithm must give the same result whatever the scanning order of the input's pixels. Such algorithm should therefore remove only simple points whose removal does not change other simple points to non-simple points. In the second work, the authors propose a sequential order-independent thinning algorithm. Such thinning scheme hardly generalizes to 3d or more.

A widely used strategy to obtain a centered skeleton with a sequential thinning process consists of computing a priority function on the object and removing the simple points of X according to the value of this function ([DP81]) : at each step, the simple point that is removed is one with the lowest value. The Euclidean distance map is widely used as priority function in order to remove points "layer by layer" ([TV92], [CCZ07], [MFV98]). Other works use discrete distances, such as the chamfer distance ([Pud98]), to decide of a removal order.

Algorithm 3: Basic Thinning with Priority(X, W, D, k)

Data: A k-connected shape X, a priority function D and a subset $W \subseteq X$
Result: A skeleton of X
while *there exists a* k*-simple point in* $X \backslash W$ **do**
  $A = \{y \in X \backslash W | y \text{ is k-simple for } X\}$
  $B = \{x \in A | \text{ for all } y \in A, D(x) \leqslant D(y)\}$
  Let $z \in B$
  $X = X \backslash \{z\}$
end
return X

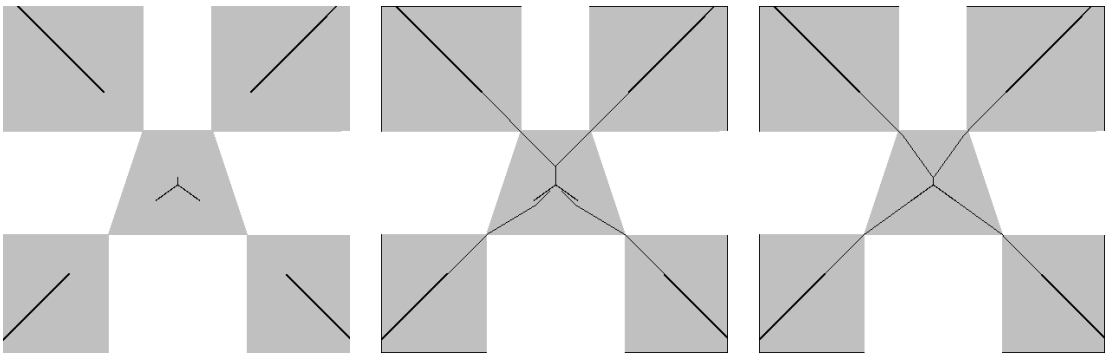
Algorithm 3 shows the basic thinning scheme based on a priority function. The set W, called *inhibitor set*, is a set of points of the input object which should be in the resulting skeleton, and D is the priority function used to decide of an order of points removal (here, the lower priority means faster removal, so it is possible to use an Euclidean distance map as priority function and obtain satisfying results, as shown on Fig. 15a).

An inhibitor set allows to choose "anchor points" for the skeleton, and therefore preserve the visual aspect of the original object in the skeleton, with an appropriate choice of anchor points (see Sec. 3.3): for example, in [DP81], the authors use the Euclidean medial axis as inhibitor set. However, when performing a thinning guided by an Euclidean distance map, the points of the inhibitor set and the directions of thinning followed by the algorithm are not always "compatible" (see Fig. 16b). In [TV92], the authors use a thinning algorithm where the slope of the priority function is used to dynamically add points to the constraint set. In [CCZ07], the authors propose to merge the slope calculation with the priority function, leading to a new priority function and a new thinning algorithm which works in 2d and 3d (see Fig. 16c).

3.2.2 PARALLEL REMOVAL OF SIMPLE POINTS

As previously explained in section 3.1.2, removing simple points simultaneously from an object usually "breaks" the topology. However, some theories have been elaborated in order to characterize sets of simple points which can be removed at the same time.

In a parallel thinning scheme, the decision of removing a point during an iteration must be completely independent of other removal decisions taken during the same



a) b) c)

Figure 16: **Priority function and inhibitor sets - a)** The original shape (in grey), and the inhibitor set (in black). **b)** The result of the homotopic thinning of the image, using the Euclidean distance map as priority function: "spurious branches" appeared in the center of the shape. **c)** The result of the homotopic thinning of the image, using the method described in [CCZ07] for building an efficient priority function.

iteration: during a same iteration, all point removals could be performed by independent processors.

When performing parallel thinning, priority function are rarely used as points are naturally removed "layer by layer". According to Palágyi ([Pal08]), parallel thinning algorithms can be divided into three categories. In *directional thinning algorithms*, the main loop is divided into sub-iterations, and the thinning operator (the considered configurations of simple points) is changed from one sub-iteration to another. In *subfield-based thinning algorithms*, the points of the grid are decomposed into subsets, and at a given iteration of the algorithm, only simple points in a given subset are studied. Finally, in *fully parallel thinning algorithms*, no sub-iteration takes place : the same thinning operations (which usually removes sets of simple points) are performed on the object at each iteration of the main loop.

3.2.2.1 FULLY PARALLEL ALGORITHMS

Given an object $X \subset \mathbb{Z}^n$, a fully parallel thinning algorithm consists of deciding of a set of simple points to be removed based on a criterion C, and removing all this set from the input object. The global scheme of fully parallel thinning algorithms is presented in Alg. 4.

Algorithm 4: Basic Fully Parallel Thinning(X)

repeat
  $Y = \{x \in X \mid x \text{ is simple for } X \text{ and } C(x) \text{ is true}\}$
  $X = X \backslash Y$
until $Y = \emptyset$;

Rutovitz was the first to propose a (fully) parallel thinning algorithm, in 1966 ([Rut66]). However, it is well known that Rutovitz's algorithm does not always preserve topology, and "patches" exists in order to correct it ([Eck88], [Cou06]).

A fully parallel thinning algorithm - Pavlidis in 1981 In 1981, Pavlidis published a fully parallel thinning algorithm ([Pav81]), in 2d for 8-connected objects, that preserve topology ([Cou06]). In this algorithm, the author defines *multiple pixels*, *corner pixels* and *contour pixels*. Given $X \subset \mathbb{Z}^2$ and $x \in X$, the point x is a *contour point* of X if it has an element of $\overline{X}$ in its 4-neighbourhood.

Definition 3.2.1 *[Pav82b] Let $X \subset \mathbb{Z}^2$, a contour point $x \in X$ is a* multiple point *of X if one of the following condition is satisfied :*

- *One or none of its 8-neighbours belongs to X.*
- *Its neighbourhood conforms to the pattern shown Fig. 17-(a,b) or any pattern obtained from them by a rotation multiple of $\Pi/2$. In this figure, the points marked 0 are elements of $\overline{X}$, the points marked 2 are contour points of X, and each group of pixels marked A or B must contain at least one element of X.*
- *Its neighbourhood conforms to the pattern shown Fig. 17-c or any pattern obtained from them by a rotation multiple of $\Pi/2$. In this figure, the points marked 0 are elements of $\overline{X}$, the points marked 2 are contour points of X, and each group of pixels marked A, B or C must contain at least one element of X. Moreover, if both pixels marked C are elements of X, then the values of pixels A and B can be anything.*

The point x is a corner point *if its neighbourhood conforms to the pattern shown Fig. 17-d or any pattern obtained from them by a rotation multiple of $\Pi/2$. In this figure, the points marked 0 are elements of $\overline{X}$, the points marked 2 are contour points of X, and the points marked 1 are elements of X.*

A	A	A
0	x	0
B	B	B

(a)

A	A	A
A	x	0
A	0	2

(b)

A	A	C
0	x	2
B	B	C

(c)

0	0	0
0	x	2
0	2	1

(d)

Figure 17: Patterns used for thinning in alg. 5

Most fully parallel thinning algorithm are based on the same scheme: some masks are tested on the simple points of an object, and depending on the results of the test (some masks should be matched by the points, and others should not be matched), they are marked for removal; then, all marked simple points are removed simultaneously. Algorithm 5 produces a symmetrical skeleton, and uses four masks of nine pixels each (it can be legitimately argued that the rotations and the various conditions set on points A, B and C give birth to more masks), of size 3x3 each. Figures 35 p.83 and 36 p.83 show the results produced by Alg. 5.

Several other 2d parallel thinning algorithms have been proposed since, using other masks for simple points removal or for detecting interesting features in the skeleton. The algorithm proposed in [Hal89] uses 4 masks (plus their $\Pi/2$ rotations) of size 5x5 to decide if a point should be flagged, and the resulting skeleton is symmetric. In [EM93], the author uses the notion of perfect points (based on 5x5 masks) to produce

Algorithm 5: Pavlidis81(X) [Pav82b]

Data: An 8-connected shape X
Result: A skeleton of X

```
repeat
    C = {x ∈ X|x is a contour point of X}
    M = {x ∈ X|x is a multiple point of X}
    N = {x ∈ X|x is a corner point of X}
    Y = C\{M ∪ N}
    X = X\Y
until Y = ∅;
return X
```

a symmetric skeleton. Moreover, the author proves that using 3x3 masks only in a fully parallel thinning algorithms gives limited freedom of action for points removal. In [BM99], the author gives a method for producing a symmetric skeleton with three masks (and their rotations).

Although fully parallel thinning algorithms are order independent algorithms, the inverse is not always true. Indeed, in some order independent algorithms, such as the one proposed in [KNP09], in order to classify a point, it is necessary to know how its neighbours were classified: these algorithms are not parallel, although removal could be done, at a certain stage of the algorithm, in parallel.

Small masks algorithm - Guo and Hall in 1992 It was proven by [Hal93] that, when designing a fully parallel algorithm with endpoints preservation, at least 11 pixels masks should be used for a correct detection of simple points to remove. Moreover, these masks should contain the 8-neighbourhood of each point. The algorithm presented in [GH92] (see alg. 6) uses such mask, and produces asymmetrical results.

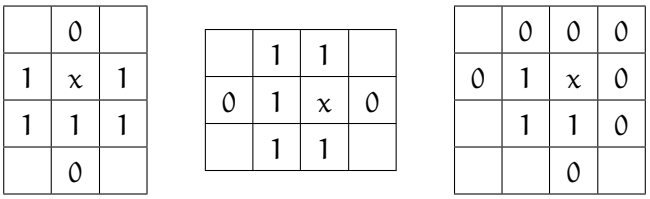


Figure 18: Patterns used for thinning in alg. 6 - empty cells can either contains 1s or 0s

Thinning based on Ronse's minimal simplet sets - Németh and Palágyi in 2009 Recent work based on Ronse's minimal simple sets ([Ron88]) have lead to three bi-dimensional fully parallel thinning algorithms ([NP09]): the basic definitions of simple sets were used in this work for building masks and properties that simple points should fulfil in order to be removed simultaneously. In this work, the authors characterize first curves points which should be kept safe from deletion (called *endpoints*), and then define how to remove simultaneously simple points that are not end points.

Algorithm 6: GuoHall92(X) [GH92]

Data: An 8-connected shape X
Result: A skeleton of X

```
repeat
    A = {x ∈ X| there is one 4-connected component in Γ*_8(x) ∩ X}
    G = {x ∈ X|Γ_4(x) ∩ X̄ ≠ ∅}
    B = {x ∈ X|Γ*_8(x) ∩ X has at least two elements}
    P = {x ∈ X|x does not satisfy any mask presented in Fig. 18}
    Y = A ∩ B ∩ G ∩ P
    X = X\Y
until Y = ∅;
return X
```

Definition 3.2.2 *[NP09] Let $X \subset \mathbb{Z}^2$, a point $x \in X$ is an* endpoint *if $\Gamma^*_8(x)$ contains only one element of X, or contains two elements of X which are 4-neighbours.*

The point x is a self-deletable point for X *if it is not an endpoint and it is simple for X.*

The point x is a double-deletable point for X *if it is self-deletable, and that for any self-deletable point $q \in \Gamma^*_4(x)$, q is simple for $X\backslash\{p\}$ or p is simple for $X\backslash\{q\}$.*

The point x is a square-deletable point for X *if it does not satisfy the mask presented in figure 19.*

The point x is a deletable point *if it is self-deletable, double-deletable and square-deletable (the self-deletable condition is unnecessary because naturally included in the double-deletable condition).*

0	0	0	0
0	x	1	0
0	1	1	0
0	0	0	0

Figure 19: Patterns used for thinning in Alg. 7

Algorithm 7: NemethPalagyi09(X) [NP09]

Data: An 8-connected shape X
Result: A skeleton of X

```
repeat
    X = X\{x ∈ X|x is deletable for X}
until stability;
return X
```

Algorithm 7 produces symmetrical results on large shapes (when the mask presented in figure 19 is never matched).

Thinning based on critical kernels - Bertrand and Couprie in 2008 Critical kernels were used in order to design various thinning algorithms in [BC08], proved valid thanks

to the work presented in [Ber07]. Bertrand defined in 2006 the concept of crucial faces, which can be seen as the set of voxels which should not be removed from an object in order to preserve its topology. The authors define four masks for detecting crucial voxels, as shown in Figure 20. If a pixel p of the object can be placed as a pixel labelled P in the mask, and that the rest of the pixels around p match the mask, then p matches the mask.

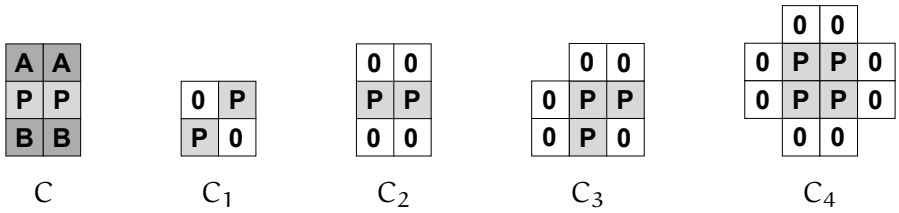


Figure 20: **The four masks used by Bertrand and Couprie's algorithm (MK2) -** In each mask, pixels labelled 0 must belong to the background, pixels labelled P must be simple in the object, and in the mask C, at least one pixels labelled A and one pixel labelled B must belong to the object. The tested pixels can be any of the pixels labelled P. All masks come with their $\Pi/2$ rotations (total: 11 masks).

The authors characterize crucial pixels thanks to the masks shown on Fig. 20.

Proposition 3.2.3 *[BCo8] Let* $X \in \mathbb{Z}^2$*, and let* P *be the set of simple pixels in* X*.*

- *The pixel* p *is* 1-crucial for $< X, P >$ *if and only if it matches the pattern* C *on Fig. 20.*
- *The pixel* p *is* 0-crucial for $< X, P >$ *if and only if it matches any of the pattern* C_1, C_2, C_3, C_4 *on Fig. 20.*

Algorithm 8: $MK^2_\alpha(X)$ [BCo8]

Data: An 8-connected shape X
Result: A skeleton of X

```
repeat
    P = {p ∈ X | p is simple for X}
    R = {p ∈ P | p is 1-crucial or 0-crucial for < X, P >}
    X = (X\P) ∪ R
until (P\R) = ∅;
return X
```

The algorithm produces symmetrical skeletons.

3d fully parallel thinning - Manzanera in 1999 In the mid 90s, important results in three-dimensional fully parallel thinning were established by Ma, who based his work on 3d extensions of Ronse's minimal non-deletable sets ([Ma94]). The main theorem established by Ma states that only verifications on limited configurations of points (unit square or cube) is sufficient in order to prove the topological soundness of an algorithm :

Theorem 3.2.4 *[Ma94] Let $X \subset \mathbb{Z}^3$ and let us consider 26-connectivity. An algorithm that removes points in parallel in X removes 26-simple sets from X if no connected component of X contained in a unit lattice cube is completely removed, and every subset of X that is contained in a unit lattice square and that is removed by the algorithm is 26-simple.*

Thanks to this work, it was then possible to test the validity of 3d parallel thinning algorithms. However, the same author later proposed two fully parallel 3d thinning algorithms in [Ma95] and [MS96], which were later proved to be false in [Loh01], [WB07], [Loh08] and [Loh10]. The first (valid) fully parallel 3d thinning algorithm was proposed by Manzanera et al. in 1999, in [MBPL99], and uses five masks (see, Fig. 21 and hit-or-miss based operations. The authors used the results established in [Ma94] in order to prove validity of their method.

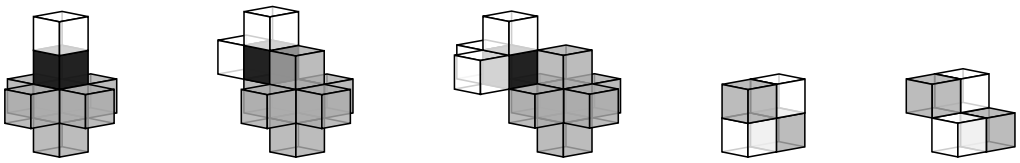


Figure 21: **The five masks used by Manzanera et al.'s algorithm -** In each mask, dark voxel is the considered voxel, grey voxels must belong to the object, and white voxels must belong to the background (unrepresented voxels can be any kind of voxels). All masks come with their $\Pi/2$ rotations around Ox, Oy or Oz.

Given $X \subset \mathbb{Z}^3$ and $p \in X$, we say that the k-neighbourhood of p fits the mask A if $A \subseteq \Gamma_k(p)$.

Algorithm 9: Manzanera99(X)

Data: A 26-connected shape X
Result: A skeleton of X

repeat
  $A = \{x \in X | x$ fits the mask $\alpha 1$, $\alpha 2$ or $\alpha 3$ (see Fig. 21) or any of their $\Pi/2$ rotations around Ox, Oy or $Oz\}$
  $B = \{x \in X |$the 18-neighbourhood of x fits the mask $\beta 1$ (see Fig. 21) or any of its $\Pi/2$ rotations around Ox, Oy or $Oz\}$
  $C = \{x \in X |$the 26-neighbourhood of x fits the mask $\beta 2$ (see Fig. 21) or any of its $\Pi/2$ rotations around Ox, Oy or $Oz\}$
  $Y = A \backslash (B \cup C)$
  $X = X \backslash Y$
until Y *is empty*;
return X

Manzanera's algorithm remains very simple, thanks to the use of a very small number of masks. Moreover, it does not only use masks that should match or not a given point, but it use also masks that should match a point's neighbourhood.

3d fully parallel thinning based on critical kernels - Bertrand and Couprie in 2006
Based on the work proposed by Bertrand in [Ber07], Bertrand and Couprie proposed a

general method for performing 3d fully parallel thinning in [BCo6]. As for the 2d case, this thinning is based on the notion of crucial elements, based on four masks shown Fig. 22.

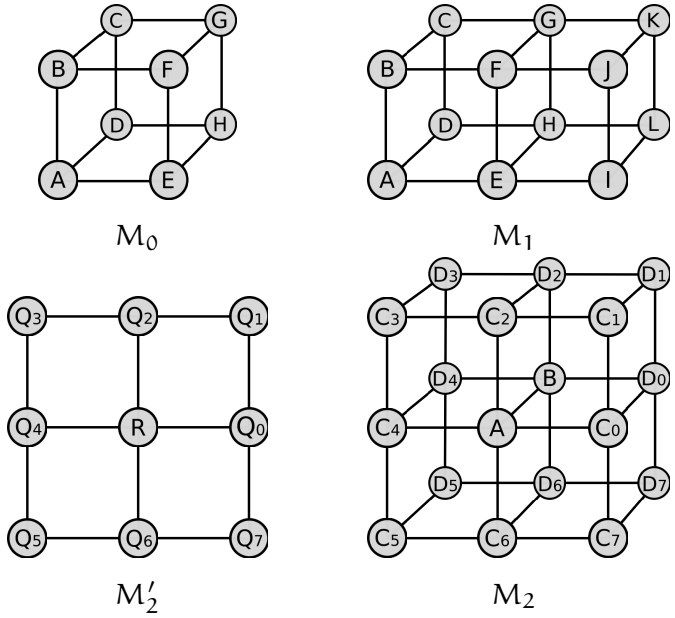


Figure 22: **The four masks used by Bertrand and Couprie's algorithm (SK3) -** Each circle represents a voxel. All masks come with their $\Pi/2$ rotations around Ox, Oy or Oz (total: 7 masks).

Definition 3.2.5 *[BCo6] Let $X \subset \mathbb{Z}^3$, let M be a set of voxels of X, and let us consider the masks shown on Fig. 22.*

- *The set M matches the mask M_2 if:*
  - $M = \{A, B\}$, *and*
  - *all elements of M are simple for X, and*
  - *In the 2d configuration M'_2 obtained by setting $R \in M'_2$ and, for all $i \in \{0, ..., 7\}$, $Q_i \in M'_2$ if $C_i \in X$ or $D_i \in X$, the pixel R is simple in the 2d sense.*
- *The set M matches the mask M_1 if:*
  - $M = \{E, F, G, H\} \cap X$, *and*
  - *all elements of M are simple and do not match mask M_2, and*
  - $\{E, G\} \subseteq M$ *or* $\{F, H\} \subseteq M$ *(inclusive or), and*
  - *we either have* $(\{A, B, C, D\} \cap X \neq \emptyset$ *and* $\{I, J, K, L\} \cap X \neq \emptyset)$ *or* $(\{A, B, C, D\} \cap X = \emptyset$ *and* $\{I, J, K, L\} \cap X = \emptyset)$.
- *The set M matches the mask M_0 if:*
  - $M = \{A, B, C, D, E, F, G, H\} \cap X$, *and*
  - *all elements of M are simple and do neither match mask M_2 nor mask M_1, and*
  - $\{A, G\} \subseteq M$ *or* $\{B, H\} \subseteq M$ *or* $\{C, E\} \subseteq M$ *or* $\{D, F\} \subseteq M$ *(inclusive or).*

Definition 3.2.6 *[BC06] Let* $X \subset \mathbb{Z}^3$, $K \subset \mathbb{Z}^3$, *let* M *be a set of voxels of* $X \backslash K$, *and let us consider the masks shown on Fig. 22.*

- *The set* M *is a* 2-crucial clique for $< X, K >$ *iff* M *matches the mask* M_2.
- *The set* M *is a* 1-crucial clique for $< X, K >$ *iff* M *matches the mask* M_1.
- *The set* M *is a* 0-crucial clique for $< X, K >$ *iff* M *matches the mask* M_0.

Algorithm 10: $SK^3(X)$ [BC06]

Data: A 26-connected shape X, a set of voxels K
Result: A skeleton of X

```
repeat
    P = {p ∈ X|p is not simple for X} ∪ K
    R2 = {p ∈ X|p belongs to a 2-crucial clique included in S\P}
    R1 = {p ∈ X|p belongs to a 1-crucial clique included in S\(P ∪ R2)}
    R0 = {p ∈ X|p belongs to a 0-crucial clique included in S\(P ∪ R2 ∪ R1)}
    X = P ∪ R2 ∪ R1 ∪ R0
until X is stable;
return X
```

The set K allows to define a set of voxels which should not be removed (in order, for example, to preserve some important visual features from the original object). The result of algorithm 10 is symmetrical.

It can be legitimately argued that this algorithm is not fully parallel thinning algorithm, as the classification of a point as, for example, part of a 1-crucial clique depends on its neighbours' classification as part of a 2-crucial clique. It is therefore not possible, with these masks, to pass on one pixel only one time, and decide if it should be removed or not.

However, the decision of classifying each point as part of a k-crucial clique (k being equal to 0, 1 or 2) can be fully made in parallel. For example, in Alg. 10, the classification of each point as part of the set R_1 can be fully done in parallel, but needs to be done after the classification of points as part of R_2. This algorithm cannot be classified as a directional algorithm, as the removal of points is made only once after all the classifications were made.

This particular algorithm could be classified in another category, that could be called *substep parallel thinning algorithm*.

3.2.2.2 DIRECTIONAL ALGORITHMS

Given an object $X \subset \mathbb{Z}^n$, directional thinning algorithms consists of dividing the simple points into several directions: at each step of the algorithm, only simple points of a given direction are removed. Directional thinning algorithms usually iterate through all possible directions in order to obtain a visually "interesting" result. With a fully parallel thinning algorithm, only one criterion C is used for deciding if a point should be added to the set of removable points; with a directional thinning algorithm, various criteria are used (one criterion per direction). The global scheme of directional algorithms is presented in Alg. 11.

Algorithm 11: Basic Directional Thinning(X)

repeat
  for all *direction* d **do**
    $Y_d = \{x \in X \mid x \text{ is simple for } X \text{ and } C_d(x) \text{ is true}\}$
    $X = X \backslash Y_d$
  end
until *all* Y_d *are empty*;

Rosenfeld, a first directional algorithm in 4 sub-iterations In 1975, Rosenfeld proposed a popular method for performing directional thinning in 2d ([Ros75]). The method consists in dividing simple points according to four directions (called north, east, west and south). We give here the 8-connected version of this thinning algorithm.

8	1	5
4	0	2
7	3	6

Figure 23: Numbering the 8-neighbours of a pixel.

Definition 3.2.7 *[Ros75] Let* $X \subset \mathbb{Z}^2$*, a simple point* x *of* X *is a* north *(resp.* east, south, west*)-simple point, also called a simple point of direction* 1 *(resp.* 2, 3, 4*) if its neighbour numbered* 1 *(resp.* 2, 3, 4*) is in* $\overline{X}$ *and if* x *has at least two* 8*-neighbours belonging to* X*.*

Algorithm 12: Rosenfeld75(X)

Data: An 8-connected shape X
Result: A skeleton of X

```
1  remove = true
2  while remove do
3  |  remove = false
4  |  for i : 1 → 4 do
5  |  |  Y = {x ∈ X | x is a simple point of direction i}
6  |  |  if Y ≠ ∅ then
7  |  |  |  remove = true
8  |  |  end
9  |  |  X = X\Y
10 |  end
11 end
12 return X
```

Zhang and Suen, a directional thinning algorithm in 2 sub-iterations In 1984, Zhang and Suen proposed in [ZS84] a 2d directional thinning algorithm in 2 sub-iterations. The algorithm consists of removing south-east corner points and north or

west boundary point, and then removing north-west corner points and south or east boundary points.

Considering the figure 23, we denote by *north* (resp. *south*, *east*, and *west*) neighbour of a pixel the neighbour labelled 1 (resp. 3, 2, 4). For the sake of clarity, we define the following (these definitions were not given "as is" by the original authors) :

Definition 3.2.8 *Given* $X \subset \mathbb{Z}^2$, *a point* $x \in X$ *is a* ZS-point *if* $2 \leqslant (\Gamma_8^*(x) \cap X) \leqslant 6$ *and if* $\Gamma_8^*(x) \cap X$ *consists of exactly one* 4*-connected component.*

Given $X \subset \mathbb{Z}^2$, *a point* $x \in X$ *is a* ZS simple point of direction 1 *if* x *is a ZS-point, if the north, east or the south neighbour of* x *is in* $\overline{X}$*, and if the east, south or the west neighbour of* x *is in* $\overline{X}$*.*

Given $X \subset \mathbb{Z}^2$, *a point* $x \in X$ *is a* ZS simple point of direction 2 *if* x *is a ZS-point, if the north, east or the west neighbour of* x *is in* $\overline{X}$*, and if the north, south or the west neighbour of* x *is in* $\overline{X}$*.*

The thinning methodology proposed in [ZS84] is explained in Alg. 13. In [ZS84], the two authors give additional information about implementation and execution time of their algorithm.

Algorithm 13: ZhangSuen84(X)

```
Data: An 8-connected shape X
Result: A skeleton of X
remove = true
while remove do
    remove = false
    for i : 1 → 2 do
        Y = {x ∈ X|x is a ZS simple point of direction i}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

Tsao and Fu, the first 3d directional thinning algorithm (in 6 sub-iterations) A natural extension of the thinning methodology proposed by Rosenfeld in [Ros75] would consists in a 6 sub-iterations algorithm removing all simple points of a given type (up, bottom, north, south, east, west simple points). However, in 3d, this strategy cannot be straightly applied : consider a rectangle made of voxels, of thickness one and two voxels wide. In such shape, all voxels would be labelled as a same type of simple point (for example, all voxels would be labelled as up simple points) and removed simultaneously.

The first 3d directional thinning algorithm (to our knowledge) was proposed by Tsao and Fu in 1981, in [TF81b] (the two same authors proposed, earlier in the same year, another 3d thinning algorithm in [TF81a], which may be directional, but it was not possible to obtain the original article in order to read it). This algorithm runs with 6 sub-iterations.

The two authors give, in this paper, a definition of simple points using masks, and then give conditions for simultaneous simple point removal. On Fig. 24, we present the labels of the 26 neighbours of a point $x \in \mathbb{Z}^3$: the point $x \in X \subset \mathbb{Z}^3$ is labelled as an *U border point of* X (resp. *D, N, S, E or W border point*) if its neighbour with position 18 (resp. $9, 1, 5, 3, 7$) is in $\overline{X}$.

17	10	11
16	9	12
15	14	13

plane $(k-1)$

8	1	2
7	x	3
6	5	4

plane k

26	19	20
25	18	21
24	23	22

plane $(k+1)$

Figure 24: Numbering the 26-neighbours of a voxel $x = (i, j, k)$.

10	1	19
9	x	18
14	5	23

Window W_i

16	9	12
7	x	3
25	18	21

Window W_j

8	1	2
7	x	3
6	5	4

Window W_k

Figure 25: Three windows W_i, W_j, W_k.

In the following, we give the directional thinning algorithm for 6-connected objects (a 26-connected version is given in [TF81b]). Therefore, the term "simple point" refers to "simple point in 6 connectivity".

Definition 3.2.9 *A simple point* $x \in X \subset \mathbb{Z}^3$ *is a* TF simple point of direction U *(resp.* D*) if*

- *it is an U (resp. D) border point of* X *and,*
- *it is simple in the sense of* 4*-connectedness in* $W_i \cap X$ *and in* $W_j \cap X$ *(see Fig. 25) and,*
- *it has at least two* 8*-neighbours belonging to* X *in one of the two windows* W_i *or* W_j*, and it has at least one* 8*-neighbour belonging to* X *in the other window and,*
- *its neighbour on position* 9 *(resp.* 18*) belongs to* X*.*

The same characteristics should be matched for TF simple point of direction N *(resp.* S*), except that* W_i *and* W_k *should be the two windows to consider in the second and third conditions, and the neighbour on position* 5 *(resp.* 1*) should be considered in the last condition.*

The same characteristics should be matched for TF simple point of direction E *(resp.* W*), except that* W_j *and* W_k *should be the two windows to consider in the second and third conditions, and the neighbour on position* 7 *(resp.* 3*) should be considered in the last condition.*

Based on this, the authors propose a 6 sub-iterations directional thinning algorithm, presented in Alg. 14.

Algorithm 14: TsaoFu81(X)

```
Data: A 6-connected shape X
Result: A skeleton of X
remove = true
while remove do
    remove = false
    for i ∈ {U,D,N,S,E,W} do
        Y = {x ∈ X|x is a TF simple point of direction i}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

Bertrand in 1995, a surface preserving directional thinning algorithm In 1995, Bertrand proposes in [Ber95a] a 6 sub-iterations 3d directional thinning algorithm, which preserves surfaces. The following definitions and algorithms are given for a 6-connected object of $\mathbb{Z}^3$. Moreover, we re-use the definition of *(U,D,N,S,E,W)-border point* given in the previous paragraph.

Definition 3.2.10 *Given* $X \subset \mathbb{Z}^3$ *and* $x \in X$*, we define the two following sets :*

- $A_{26} = \{y \in (X \cap \Gamma^*_{26}(x)) \mid \Gamma^*_{26}(y) \cap \Gamma^*_{18}(x) \cap \overline{X} = \emptyset\}$
- $\overline{A_{26}} = \{y \in (\overline{X} \cap \Gamma^*_{26}(x)) \mid \Gamma^*_{26}(y) \cap \Gamma^*_{18}(x) \cap \overline{X} = \emptyset\}$

A d*-border point (with* $d \in \{U, D, N, S, E, W\}$*) is not an* end-point *iff*

$$|\overline{A_{26}}| = 0 \text{ and } |A_{26}| \neq 0 \text{ and } |\Gamma^*_6(x) \cap X| \leqslant |A_{26}| + 2.$$

Algorithm 15 gives the thinning scheme proposed in [Ber95a].

Palágyi in 2002, a directional thinning algorithm in 3 sub-iterations In 2002, Palágyi proposed in [Pal02] a 3 sub-iterations 3d directional thinning algorithm, which preserves medial surfaces of an object. This algorithm removes, in a same iteration, border points of "opposite directions" (i.e. U and D-border points, N and S-border points, E and W-border points). In order to do so, the author proposes a set of masks: on Fig. 26, we present the masks for the UD directions. By performing a rotation of $\Pi/2$ around the i or j axis, one can obtain the masks for the NS directions or for the EW directions.

Algorithm 16 gives the thinning scheme originally proposed in [Pal02].

Lohou and Bertrand in 2005, a directional thinning algorithm based on P-simple points In 2005, Lohou and Bertrand proposed in [LB05] a 6 sub-iterations 3d directional thinning algorithm based on P-simple points (one year earlier, a 12 sub-iterations directional thinning algorithm was proposed by the same authors in [LB04]).

Algorithm 15: Bertrand95(X)

Data: A 6-connected shape X
Result: A skeleton of X

```
remove = true
while remove do
    remove = false
    for d ∈ {U,D,N,S,E,W} do
        Y = {x ∈ X|x is a d-border point which is simple and not an end point}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

Algorithm 16: Palágyi02(X)

Data: A 26-connected shape X
Result: A skeleton of X

```
remove = true
while remove do
    remove = false
    for d ∈ {UD,NS,EW} do
        Y = {x ∈ X | x matches one of the mask for the d directions}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

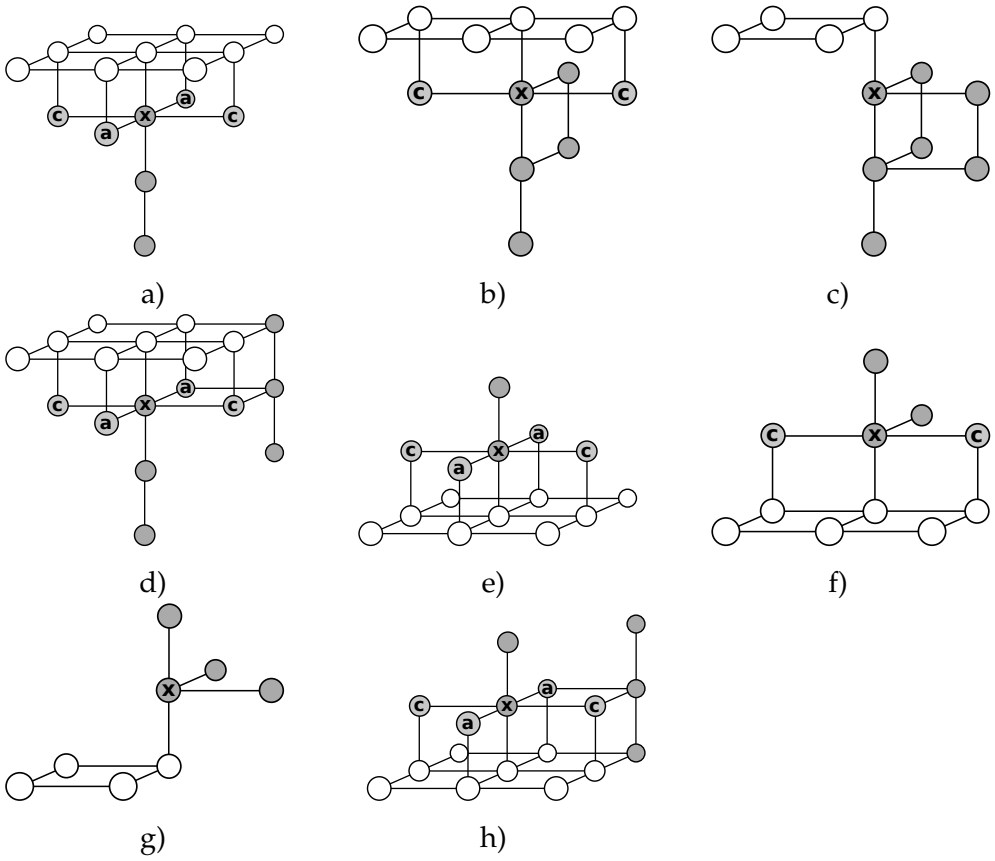


Figure 26: **Masks used by Palágyi for the UD direction:** Given $X \subset \mathbb{Z}^3$, $x \in X$ matches the mask j (with $j \in \{a, b, c, d, e, f, g, h\}$) iff all the grey points of the mask belong to X, all the white points of the mask belong to $\overline{X}$, at least one point marked "a" belongs to X, and at least one point marked "c" belongs to X. Each mask comes with its $\Pi/2$, Π and $3\Pi/2$ rotations around the vertical (or k) axis (the vertical direction being the one pointing to the top of the page). On Fig. 24, the k axis was represented as piercing through the paper; here, the k axis goes to the top of the page.

The following definitions applies for the U-border points. In order to apply it for other directions, the masks of Fig. 27 should be rotated in order to match the good direction.

Definition 3.2.11 *Let $X \subset \mathbb{Z}^3$, we define $P_2(U) = \{x \in X \mid x$ matches one of the masks of Fig. 27$\}$.*

For all $x \in \mathbb{Z}^3$, we set $P_2^x(U) = \{y \in \Gamma_{26}(x) \mid y$ matches one of the masks presented on Fig. 27 restricted to only the points located in $\Gamma_{26}(x)\}$.

In Alg. 17, P-simple points are used to choose points to be deleted: the reader can find the definition of such points in Def. 3.1.5 p.53.

Other periods exist for directional algorithms Many other directional thinning algorithms were proposed in 3d, and making a precise description of all of them would be too exhaustive. However, a quick overview of the thinning algorithms proposed show that algorithms with a period of 6 (the most "natural" way of considering directions in the three-dimensional space) are the most widespread ([GB90], [MDC90],

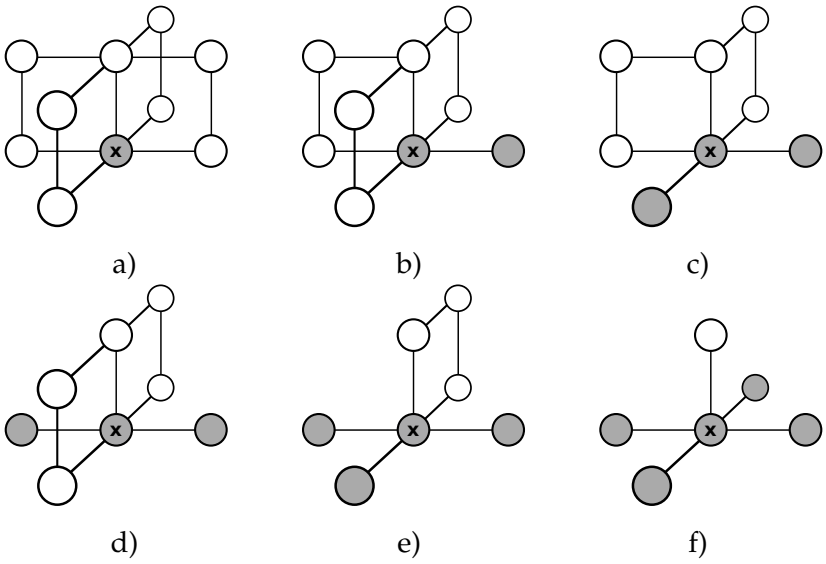


Figure 27: **Masks used by Lohou and Bertrand for the U direction:** Given $X \subset \mathbb{Z}^3$, $x \in X$ matches the mask j (with $j \in \{a, b, c, d, e, f\}$) iff all the grey points of the mask belong to X and all the white points of the mask belong to $\overline{X}$. Each mask comes with its $\Pi/2$, Π and $3\Pi/2$ rotations around the vertical axis (same axis as on Fig. 26).

[LKC94], [PK98], [Maoo] (working for 18-connected object), [XTP03]). Other algorithms were proposed with different periods, such as in [PK99a] or [LB04], where the authors propose algorithms in 12 sub-iterations, [PK99b] where an algorithm with a period of 8 is given, or [Pal02] (previously detailed) and [Pal07b] where algorithms based on 3 sub-iterations are given.

In 2d, although 4 sub-iterations algorithms are the most usual, we can point out the work presented in [CH89] (completed with a note in [Che92]) where the authors give a method for designing 2 sub-iterations directional thinning algorithms.

3.2.2.3 SUBFIELD ALGORITHMS

Given an object $X \subset \mathbb{Z}^n$, thinning algorithms based on subfields consists of partitioning the discrete space $\mathbb{Z}^n$ into several subsets $S_0, ..., S_{k-1}$, called subfields. Such algorithms usually iterate through all possible subfields, and at each step, remove all simple points located in a given subfield. Subfield based algorithms use two kind of criteria : one criterion C is based on the point's neighbourhood and remains constant during the thinning, and one criterion is based on the point's position and changes during the thinning.

Directional thinning algorithms use a collection of criteria (called directions) based on the point's neighbourhood for deciding of deletion, fully parallel thinning algorithms use one criterion based on the point's neighbourhood for deciding of deletion, and subfield based thinning algorithms use one criterion based on the point's neighbourhood and various criteria based one the point's position in the grid for deciding of deletion.

The general scheme of a subfield based thinning algorithm is given in Alg. 18.

A 2d thinning algorithm based on 4 subfields As previously seen in Sec. 3.1.1 p.50, it is only necessary to look at the 8-neighbourhood of a point in order to decide if the point

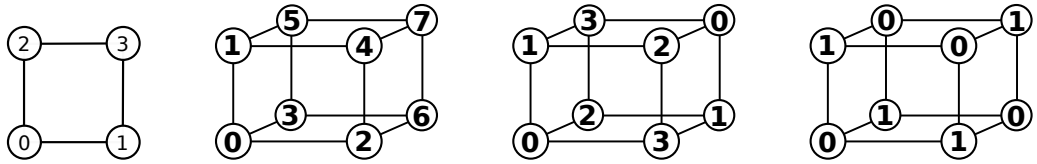
Algorithm 17: Lohou05(X)

Data: A 26-connected shape X
Result: A skeleton of X

```
remove = true
while remove do
    remove = false
    for d ∈ {U,D,N,S,E,W} do
        Y = {x ∈ X|x is a P_2^x(d)-simple point and |Γ*_26(x) ∩ X| > 1}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

Algorithm 18: Basic Subfield Thinning(X)

```
repeat
    for all subfield S_i do
        Y_i = {x ∈ X | x is simple for X, C(x) is true, and x ∈ S_i}
        X = X\Y_d
    end
until all Y_i are empty;
```



a) b) c) d)

Figure 28: **Basic partitions of the space:** An example of a) partition of the discrete space $\mathbb{Z}^2$ into 2 subfields and b) partition of the discrete space $\mathbb{Z}^3$ into 8 subfields. The space should be regularly tiled with the given pattern; two points belong to the same subfield iff they have the same label. Partition into 4 subfielsd (resp. 2 subfields) is shown on c (resp. d).

is simple or not. Therefore, two simple points x and y can be simultaneously removed if $y \notin \Gamma_8(x)$. Based on this, we can partition the space into 4 subfields (S_0, S_1, S_2, S_3), as depicted on Fig. 28a: this partition of the space ensures that two points which are 8-neighbours cannot belong to the same subfield. Algorithm 19 presents a very basic 2d thinning algorithm based on 4 subfields.

A 3d thinning algorithm based on 8 subfields The same way than in the previous paragraph, we can propose a basic 3d thinning scheme based on 8 subfields using the partition $(S_0, ..., S_7)$ depicted on Fig. 28b: this partition of the space ensures that

Algorithm 19: Basic 4 subfields(X)

Data: An 8-connected shape X
Result: A skeleton of X

```
remove = true
while remove do
    remove = false
    for i = 0 → 3 do
        Y = {x ∈ X|x is a simple point and x ∈ S_i (see Fig. 28a}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

two points which are 26-neighbours cannot belong to the same subfield. Algorithm 20 presents a very basic 3d thinning algorithm based on 8 subfields.

Algorithm 20: Basic 8 subfields(X)

Data: A 26-connected shape X
Result: A skeleton of X

```
remove = true
while remove do
    remove = false
    for i = 0 → 7 do
        Y = {x ∈ X|x is a simple point and x ∈ S_i (see Fig. 28b}
        if Y ≠ ∅ then
            remove = true
        end
        X = X\Y
    end
end
return X
```

In this very basic algorithm, little care is given to the visual aspect of the final result (the algorithm will result in a topological kernel of the input). In [SCD97], [BA94] and [NKP10b], the authors give various strategies in order to preserve the visual aspect of the input during thinning: in the last article, the authors give also a different thinning scheme which consists of deleting only simple points which were on the border of the object when the iteration over the subfields started.

Németh, Kardos and Palágyi in 2010, a 3d thinning algorithm based on 4 subfields In [NKP10b], the authors proposed a 3d thinning algorithm based on a decomposition of the grid into 4 subfields: results established in [MWL02], where other algorithms

based on 4 subfields were presented, are used to prove the topological soundness of the algorithms.

Definition 3.2.12 *Given a 26 connected object* $X \subset \mathbb{Z}^3$ *and* $x \in X$*, the point* x *is* SF-4-deletable *if it is simple for* X *and if, for each* $y \in (X \cap (\Gamma_{26}(x) \backslash \Gamma_{18}(x)))$ *such that* y *comes before* x *in the lexicographic order of voxels,* y *is not simple for* X*.*

Algorithm 21 is the 4 subfields algorithm proposed in [NKP10b]: it uses a decomposition of the space into 4 subfields (S_0, S_1, S_2, S_3), presented on Fig. 28c. In this algorithm, the authors separate the removal process from the geometric analysis, by defining points of type ϵ as points which should not be removed during thinning: later, the authors give various definitions of ϵ, leading to various results (topological kernel, curvilinear skeleton, surfacic skeleton).

Algorithm 21: Nemeth 4 subfields(X)

Data: A 26-connected shape X
Result: A skeleton of X

```
repeat
    E = {p | p is a border point of X, but not a point of type ε of X}
    for i = 0 → 3 do
        Y_i = {x ∈ X | x is SF-4-deletable for X, and x ∈ S_i}
        X = X\Y_i
    end
until Y_0 ∪ Y_1 ∪ Y_2 ∪ Y_3 = ∅;
return X
```

Németh, Kardos and Palágyi in 2010, a 3d thinning algorithm based on 2 subfields In [NKP10a], the authors propose to use a decomposition of the grid into two subfields (S_0, S_1), presented on Fig. 28d. As they did in [NKP10b] (see previous paragraph), the authors again define points of type ϵ as points which should not be removed during thinning: various definitions of epsilon are then proposed, giving various visual results.

The following definition is a 3d version of Def. 3.2.2 p.60.

Definition 3.2.13 *Given* $X \subset \mathbb{Z}^3$*,*

the point x *is a* self-ϵ-deletable point for X *if it is not a point of type* ϵ *and it is simple for* X*,*

the point x *is a* square-ϵ-deletable point for X *if it is self-deletable, and that for any self-deletable point* $q \in \Gamma_{18}(x) \backslash \Gamma_6(x)$*,* q *is simple for* $X \backslash \{p\}$ *or* p *is simple for* $X \backslash \{q\}$*,*

the point x *is a* cube-ϵ-deletable point for X *if it does not match the mask presented in figure 29.*

3.2.2.4 BLOCK ALGORITHMS

We conclude this state of the art of skeletonization with another category of parallel algorithms: the "block" algorithms. In a block algorithm, the image is partitioned into several blocks (usually of same size), and each block is thinned independently from other blocks using a classical thinning strategy (sequential, fully parallel, ...).

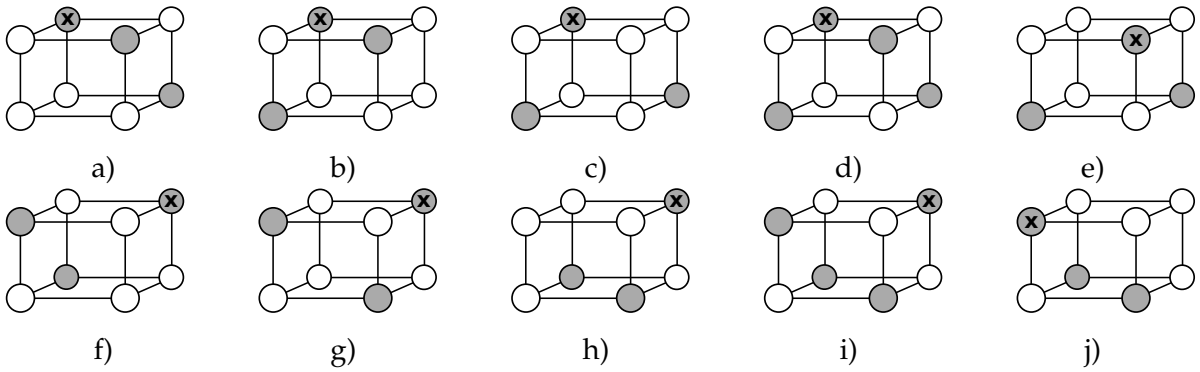


Figure 29: **Masks used by Németh and al:** Given $X \subset \mathbb{Z}^3$, $x \in X$ matches the mask k (with $k \in \{a, b, c, d, e, f, g, h, i, j\}$) iff all the grey points of the mask belong to X and all the white points of the mask belong to $\overline{X}$. Only these masks should be taken into account: no rotation should be considered.

There is three main "challenges" when elaborating such algorithm: the process of dividing the image and performing thinning on each block can be repeated only a fixed amount of time, topology should be preserved (this can be a problem in the zones where the blocks meet), and the visual result should be satisfying ("close" to what would have resulted the use of the classical thinning strategy on the whole image. These algorithms allow to perform thinning of very large images, that cannot be entirely loaded into memory.

Such algorithm is presented in [FMP+04]: the main idea of the authors consists in, given $X \subset \mathbb{Z}^n$, finding the size R of the biggest ball contained in X, and then decomposing the original image into blocks. The thinning is performed, in parallel, in each block: only points whose distance from the block's border is greater than the distance from the object's border will be considered (and eventually removed) by the algorithm. Then, all blocks are translated in order to now take into account points which were previously ignored by the thinning, and the whole process is iterated.

In order to obtain a thin skeleton, the size of each block should strictly superior to $2R$. Moreover, using blocks of size $4R$ allows to iterate the whole process (thinning and block translation) exactly $(n + 1)$ times (when working in an n-dimensional space).

3.3 RECONSTRUCTIBILITY: PRESERVING THE VISUAL ASPECT DURING THINNING

When performing thinning of a shape, only the topological information of the shape is preserved during the process, and visual information (elongated parts, surfacic parts, curvature, ...) are not guaranteed to be preserved. By preserving, in the skeleton, some information about the original visual aspect of the shape, it is possible to perform better and thinner analysis of the initial object. This criteria if often called in literature the *reconstructibility*: it denotes the capacity to regenerate the original shape from its skeleton.

When trying, for example, to recognize automatically the letter "a" from the letter "q", which are topologically equivalent (one connected component, one cavity), only geometric information about the presence of an elongated part can help in distinguishing

the two shapes. If one uses a skeletonization algorithm for shapes simplification, and that the algorithm does not preserve any visual information from the initial shape, it will be impossible to discriminate an "a" from a "q". For this reason, it is necessary to have thinning algorithms "who care" about the preservation of the visual aspect of the shape.

Methods proposed in order to perform this task can be divided in two categories: either the points of the (filtered) *medial axis* of the shape is chosen as undeletable elements during thinning (and then transmitted as an *inhibitor set*, like in Alg. 3), or some points are, during the thinning process, dynamically chosen as undeletable points.

We will give a quick overview of the major methods used in both categories.

3.3.1 MEDIAL AXES

3.3.1.1 THE EUCLIDEAN MEDIAL AXIS

In the 60s, Blum ([Blu62],[Blu67]) introduced the notion of medial axis, which has since been the subject of numerous theoretical studies and has also proved its usefulness in practical applications. Although initially introduced as the outcome of a propagation process, the medial axis can also be defined in simple geometric terms. In the continuous Euclidean space, the two following definitions can be used to formalise this notion: let X be a bounded subset of $\mathbb{R}^n$:

- **Interpretation (a) of the medial axis** of X consists of the centers of the n-dimensional balls that are included in X but that are not included in any other n-dimensional ball included in X.
- **Interpretation (b) of the medial axis** of X consists of the points $x \in X$ that have more than one nearest point on the boundary of X.

These two definitions differ only by a negligible set of points (see [Mat88]), and in general, interpretation (a) of the medial axis is a strict subset of interpretation (b). The following formally defines interpretation (a) based on Euclidean balls (the reader should refer to Sec. 9.1 p.204 for a definition of Euclidean balls):

Definition 3.3.1 *Given* $X \subset \mathbb{Z}^n$ *and* D_X *its Euclidean distance transform (see Sec. 9.1 p.204),* *the* Euclidean medial axis of X *is the set* $EMA(X) = \{x \in X | \text{for all } y \in X \backslash \{x\}, B^<(x, D_X(x)) \not\subset B^<(y, D_X(y))\}$.

It is possible to adapt this definition to various distances (4, 8, 6, 26-distances for example) by changing all references to the Euclidean distances with another distance.

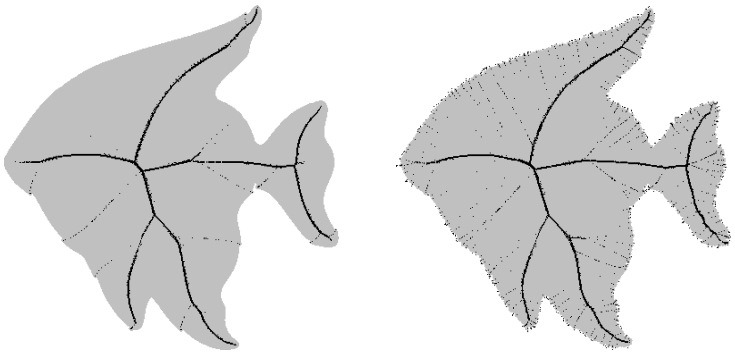
To compute the medial axis approximately or exactly, different methods have been proposed, relying on different frameworks: discrete geometry [BRS91, GF96, MFV98, RT02a, RT05, CM07, HR08], digital topology [DP81, Vin91, TV92, Pud98], mathematical morphology [Ser82, Soi99], computational geometry [AL01, OK95, AM96], partial differential equations [SBTZ99], and level-sets [KSKB95]. We focus here on medial axes in the discrete grid $\mathbb{Z}^2$ or $\mathbb{Z}^3$, which are centered in the shape with respect to the Euclidean distance.

The (discrete) Euclidean medial axis of a shape allows reconstruction of the initial shape and is centered in the shape, but is not homotopic to the initial shape. To cope

with this problem, the Euclidean medial axis can be combined (as previously explained) with an homotopic thinning algorithm in order to obtain a centered skeleton which contains visual features of the initial shape.

In the discrete framework, the Euclidean medial axis is not thin: in some parts, it can be two pixels thick. To cope with this problem, Saúde et al. propose, in [SCd06], an Euclidean medial axis on higher resolution and give some thinness properties of this axis. A study of the various properties of medial axes in the discrete framework is done in [Hul09].

Filtering the Euclidean medial axis A major difficulty when using the medial axis in applications (eg. shape recognition) is its sensitivity to small contour perturbations (see, for example, Fig. 30), in other words, its lack of stability. A recent survey [ABE09] summarizes selected relevant studies dealing with this topic. This difficulty can be expressed mathematically: the transformation which associates a shape to its medial axis is only semi-continuous (see [ABE09]). This fact, among others, explains why it is usually necessary to add a filtering step (or pruning step) to any method that aims at computing the medial axis.



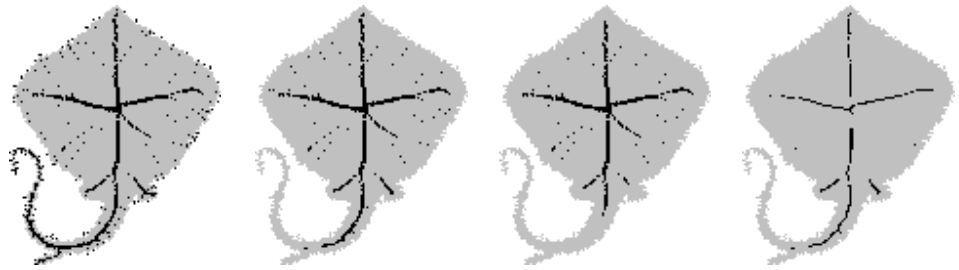
a) b)

Figure 30: **Sensitivity of the Euclidean medial axis -** In **a**, a binary shape (in grey) and its Euclidean medial axis (in black). In **b**, the same shape, but with noise added on the contour: the medial axis of the shape changed a lot compared to the medial axis of the original shape.

The first possibility to filter the medial axis is to keep only points which are centers of maximal balls of at least a given diameter. This method allows to remove spurious branches from a medial axis, but will also remove small branches which might be important for the shape understanding (see Fig. 31a to d). Indeed, some visual features of a shape can have the same size than noise features located elsewhere. The threshold makes no difference between these two features, and it can be difficult to find a satisfactory filtering parameter in some cases.

In 1992, Talbot and Vincent proposed in [TV92] (generalizing a notion proposed in [Mey79]) another method for filtering elements from the Euclidean medial axis, which relies on angles. In the continuous framework, given $X \subset \mathbb{R}^2$, D_X its Euclidean distance transform and $x \in X$, we consider the set $A_x = \{p \in \overline{X} | d(x,p) = D_X(x)\}$. The set A_x is the set of contact points between the maximal ball centered on x and the border of X. The bisector angle associated to x is the maximal angle $\widehat{p_1 x p_2}$, where p_1 and p_2 are elements of A_x.

As pointed out in [Vin91] and [AM96], the bisector angle has good properties: basically, the lower the bisector angle of x is, and the "less interesting" the point x is. In [TV92] and [CCZ07], the authors propose efficient methods for computing the bisector angle in the discrete framework. The bisector angle criteria is usually used with the distance map criteria in order to provide a robust filtering criteria [AM96, AL01, MFV98]: two filtering parameters must therefore be provided by the user. This two criteria allows to achieve more efficient filtering than what could be achieved with the only radius of the balls (see Fig. 31e).



a) Euc. Med. Ax. b) 3.6 c) 6 d) 3.6, 70°

Figure 31: **Filtering the Euclidean medial axis -** In **a**, a (noisy) shape representing a ray (in grey) and its Euclidean medial axis (in black). The medial axis has many "spurious points" due to the noise of the original shape. In **b**, removing all points which are center of an Euclidean ball of radius inferior to 3.6 allows to obtain a less noisy axis, but still noisy points remains. In **c**, higher filtering (remove all points center of an Euclidean ball of radius inferior to 6) remove some "spurious points", but remove also the tail of the shape. In **d**, using the bisector function combined with **c**, and filtering all points with a bisector angle inferior to 70° and radius inferior to 6, removes most "spurious points" and keeps important visual information from the shape.

Efficient computation of the medial axis: Rémy and Thiel in 2003 [RT03] The following naive algorithm computes the Euclidean medial axis of a shape X:

Algorithm 22: Naive Euclidean Medial Axis(X)

```
Data: A shape X
Result: The Euclidean medial axis of X
D =the Euclidean distance transform of X;
EMA = X;
foreach x ∈ X do
    foreach y ∈ (X\{x}) do
        if B^<(x, D_X(x)) ⊆ B^<(y, D_X(y)) then
            EMA = EMA\{x};
        end
    end
end
return EMA
```

For every point **x** of the shape X, we look at all other points **y** $\in X$ (although it would be sufficient to simply look at all the other points in EMA) in order to see if the

maximal ball of X centered on $\mathbf{x}$ is included in the maximal ball of X centered on $\mathbf{y}$. If it is the case, the point does not belong to the Euclidean medial axis. Algorithm 22 is not efficient in terms of computation time: let N be the number of points of the shape, the worst case complexity of this algorithm is N^3.

Fortunately, in 2003, Rémy and Thiel proposed an efficient method for computing the Euclidean medial axis of a discrete shape ([RT03], [RT05]). Their method proposes to compute two look-up tables, which both rely on a parameter R_{max}, which must be a supremum for the values of the Euclidean distance transform of the shape (if it is not, it is necessary to compute larger look-up tables).

The table $\mathcal{M}_{R_{max}}$ is the set of vectors to explore, from a point $\mathbf{x}$ of a shape X, in order to check if $\mathbf{x}$ is center of a maximal ball of X: if, for each $\mathbf{v} \in \mathcal{M}_{R_{max}}, B^<(\mathbf{x}, D_X(\mathbf{x})) \nsubseteq B^<(\mathbf{x}+\mathbf{v}, D_X(\mathbf{x}+\mathbf{v}))\}$, then $\mathbf{x}$ is center of a maximal ball for X. Thanks to this table, it is not necessary to scan all points $\mathbf{y}$ of the shape X in order to check for maximal ball inclusion (on the contrary of what was done in Alg. 22).

The table Lut gives, for any pair $(\mathbf{v}, R)$, the minimal radius R' necessary for having, for any $\mathbf{x} \in X, B^<(\mathbf{x}, R) \subseteq B^<(\mathbf{x}+\mathbf{v}, R')\}$. Thanks to this table, it is not necessary to test each point of each balls in order to test for the ball inclusion (on the contrary of what was done in Alg. 22).

Algorithm 23: Remi Thiel Axis(X)

Data: A shape X
Result: The Euclidean medial axis of X

```
1  D =the Euclidean distance transform of X;
2  EMA = X;
3  foreach x ∈ X do
4  |  foreach v ∈ M_{R_max} do
5  |  |  if D_X(x+v) ⩾ Lut(v, D_X(x)) then
6  |  |  |  EMA = EMA\{x};
7  |  |  end
8  |  end
9  end
10 return EMA
```

In [RT03] and [RT05], the authors explain how to build the look-up tables, and study the size of the set $\mathcal{M}_{R_{max}}$ in order to provide an evaluation of their algorithm's average complexity. Mainly, the set $\mathcal{M}_{R_{max}}$ grows slowly compared to R_{max}: Alg. 23 brings a huge gain when computing the Euclidean medial axis compared to the naive method (it also has an inferior worst case complexity).

3.3.1.2 THE λ-MEDIAL AXIS OF CHAZAL AND LIEUTIER

The λ-medial axis was introduced and studied by Chazal and Lieutier in the continuous framework ([CL05]).

Consider a bounded subset X of $\mathbb{R}^n$, as for example, for $n = 2$, the region enclosed by the solid curve depicted in Fig. 32 (left). Let x be a point in X, we denote by $\Pi_{\overline{X}}(x)$ the set of points of the boundary of X that are closest to x. For example in Fig. 32, we have $\Pi_{\overline{X}}(x) = \{a, b\}$, $\Pi_{\overline{X}}(x') = \{a', b'\}$ and $\Pi_{\overline{X}}(x'') = \{a''\}$.

Given $\lambda \in \mathbb{R}$, the λ-medial axis of X is the set of points $x \in X$ such that the radius of the smallest ball containing $\Pi(x)$ is superior or equal to λ. We give in the following a more formal definition of the λ-medial axis.

Let $X \subset S$, we denote by $R(X)$ the radius of the smallest ball enclosing X, that is, $R(X) = \min\{r \in \mathbb{R}^+ \mid \exists y \in \mathbb{R}^n, X \subseteq B(y, r)\}$.

Definition 3.3.2 ([CL05]) *Let X be an open bounded subset of S, and let $\lambda \in \mathbb{R}^+$. The* λ-medial axis of X *is the set of points x in X such that $R(\Pi_{\overline{X}}(x)) \geqslant \lambda$.*

A point $x \in X$ belongs to the λ-medial axis of X if the contact points between the maximal ball of X centered on x and the border of X do not hold inside a ball of radius strictly less than λ.

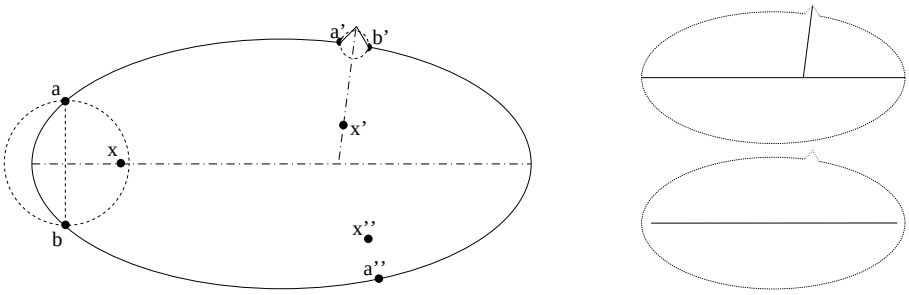


Figure 32: **Illustration of the λ-medial axis -** Left: Points x, x' and x'' and their respective closest boundary points. Top right: λ-medial axis with $\lambda = \epsilon$, a very small positive real number. Bottom right: λ-medial axis with $\lambda = d(a', b') + \epsilon$.

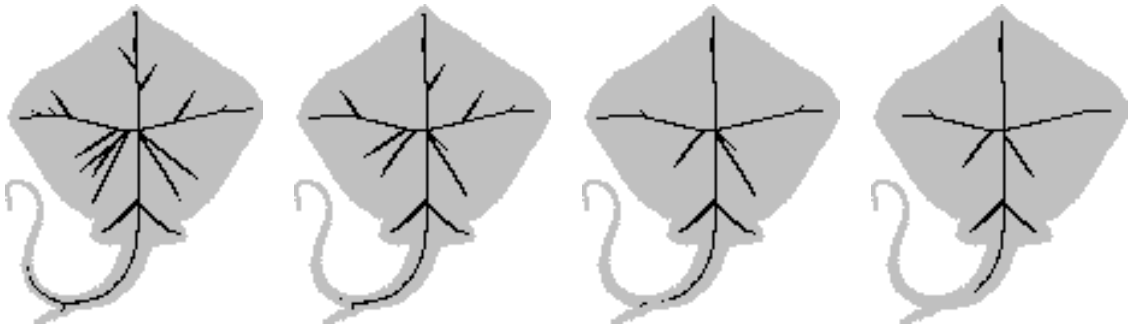
A major outcome of [CL05] is the stability of the λ-medial axis to noise: informally, for "regular" values of λ, the λ-medial axis remains stable under small perturbations of the complement of X (in regards of the Hausdorff distance). Typical non-regular values are radii of locally largest maximal balls. Such property was never proved for any other filtering criteria, this is why we chose to adapt the λ-medial axis to discrete framework in order to see how it would help in filtering skeletons. In the following chapter, we provide a discrete definition of the λ-medial axis, and we compare this axis with the two previous axes studied here. We show, on Fig. 33, the result of the discrete version of the λ-medial axis of a shape, filtered at different values.

In the field of computational geometry, the λ-medial axis has been exploited in particular by [SAAY06] to propose a robust method for reconstructing surfaces from point clouds. Also, notions closely related to the λ-medial axis have led Chazal et al. to propose stable approximations of tangent planes and normal cones from noisy samples ([CCSL09]).

3.3.1.3 THE INTEGER MEDIAL AXIS

The integer medial axis was introduced by Hesselink and Roerdink in 2008 [HR08]. Let X be a subset of $\mathbb{Z}^n$, and let $x \in X$. We denote by $\Pi'_{\overline{X}}(x)$ the element of $\Pi_{\overline{X}}(x)$ that is the smallest with respect to the lexicographic ordering of its coordinates.

Definition 3.3.3 *([HR08]) Let X be a finite subset of $\mathbb{Z}^n$, and let $\gamma \in \mathbb{R}^+$. The* γ-integer medial axis (or GIMA) of X *is the set of points x in X such that at least one $y \in N(x)$ verifies:*



$\lambda = 3$ $\lambda = 3.5$ $\lambda = 4$ $\lambda = 5$

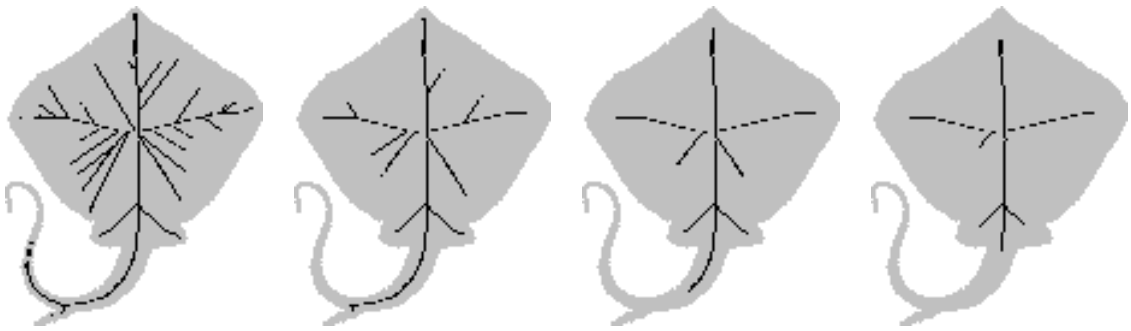
Figure 33: **The discrete λ-medial axis -** Some examples of discrete λ-medial axis of the shape in grey, filtered at different values.

- $d(\Pi'_X(x), \Pi'_X(y)) > \gamma$, *and*
- $d(m, \Pi'_X(y)) \leqslant d(m, \Pi'_X(x))$, *where* $m = \frac{x+y}{2}$.

The integer medial axis of X *is the* γ*-integer medial axis of* X *for* $\gamma = 1$.

The second condition in Def. 3.3.3 allows to get a thinner set of points, compared to what would result of such definition without the second condition.

Parameter γ allows to keep more or less elements in the result: it acts as a filtering parameter for spurious elements. On Fig. 34, we show the impact of the variations of γ on the integer medial axis of a shape. The integer medial axis will be compared with other medial axes in the next chapter.



$\gamma = 5$ $\gamma = 7$ $\gamma = 10$ $\gamma = 13$

Figure 34: **The integer medial axis -** Some examples of γ-integer medial axis of the shape in grey, filtered at different values.

3.3.2 FINDING INTERESTING FEATURES DURING THINNING

The second strategy in order to find important visual features of an object is to detect these features during thinning, as the detection of curves or surfaces becomes more easy with the removal of points from the input object.

The detection of visual features is dependent on how the simple points are detected and removed, and two algorithms based on the same detection of visual features but different simple point removal techniques will produce different results. A large part of the literature devoted to skeletonization algorithms which preserve the visual aspect propose new methods for simple points removal, while the visual aspect detection itself remains globally the same.

In all the algorithms cited hereafter, once a point has been labelled as an important visual feature of the shape, it won't be erased.

3.3.2.1 BIDIMENSIONAL SKELETONS

In the following, a reference with a star means that an example of the algorithm proposed in the cited article can be found on Fig. 35 and Fig. 36.

In 2d, in order to preserve the visual aspect of a shape during thinning, it is necessary to detect the curvilinear elements relevant of the shape and ignore the spurious branches. A common strategy ([Rut66]*, [Hil69], [SR71], [Deu72], [Pav81]*, [Pav82b], [HSCP87]*, [CH88], [GH89], [Hal89]*) in order to preserve these curvilinear elements is to detect end points after each thinning step (a step being the complete processing of all simple points located at the border of the object): a point $x \in X$ is an *end point* if $|N^*(8)(x) \cap X| = 1$ (the point has only one neighbour in X). Although these algorithms rely on the same rule for preserving visual features of the input, they produce different skeletons due to the different methods they use for simple point removal.

Observe that, given a diagonal 4-connected line of pixels, it is possible to remove simple points from the line's extremities, without meeting any end point, until the line is reduced to a single point. In [NS84] and [LW86], the authors propose to extend end points definition in order to include points who have two 4-connected object points in their 8-neighbourhood, thus avoiding the complete reduction of the line example.

Some authors recommend detecting curve points rather than end points, in order to avoid keeping some spurious branches from the result: in [HSCP87]*, the author propose to detect, on vertical lines, points that have only two 8-neighbours, one of those having to be an end point. In [Beu73], it is proposed to detect points whose 8-neighbourhood can be divided in two parts, one part containing only one object point, and the other having up to three object points.

Another common strategy used for aspect preservation is the use of masks in order to detect interesting visual features ([MU74], [WT92]*, [SW94], [BM99]*). This strategy allows to define specific collections of visual features which should be preserved, and generally produces less spurious results than the end point detection method. Masks are also used as restoring templates when performing parallel thinning: they avoid removing interesting visual features in a same loop, and are usually specific to a particular simple point removal method ([CH89], [CWSI87]*).

Some authors are concerned with the thinness of the resulting skeleton, a property hard to obtain when performing parallel thinning. Typically, a skeleton is thin if the only simple points it contains are end points. Some thinning methods ([Hil69], [NS84], [ASM84], [CS86],[Sos89]) allow to obtain directly a "thin" result while others don't ([ZS84], [Pav80], [Rut66]*). In [WHF86], the authors observe that a simple one-pass thinning is sufficient for removing "extra pixels" from a thick skeleton.

For retaining as less spurious branches as possible, some authors recommend detecting end points (or other points of interest) only after the thinning algorithm removed a

given number of layers of simple points ([Izu74] - the author of this thesis could not access any information on this article, and relied on a survey for citing this source), while others propose to smooth the shape after removing a whole layer of simple points ([CS86]).

Estimating the local curvature of the shape can also help in detecting interesting curvilinear parts: in [GN70] and [GS87], Freeman chain codes are used on the contour in order to detect high curvature zones; in [RCC90], curvature is simply estimated by counting, in the 8-neighbourhood of a point, the number of 4-connected points not belonging to the shape; in [AS80] and [AS81], "protusions" (points whose distance from interior points of the shape is superior to a given value) are detected and retained.

In [EM93]*, the authors use morphological tools in order to find, after each step of the thinning, points which would not survive to an opening with a 4-connected ball of radius 1: these points are labelled as visually interesting points. Scan lines [Pav82a, KK88] can also be used in order to detect interesting curves and compute object's skeletons at the same time: it consists in scanning the shape line per line, and retaining only one point in the middle of each interval of object's points. This method gives good results when the object has a particular orientation in space.

Many more information on features preservation during thinning (and also on thinning) in 2d can be found in [LLS92].

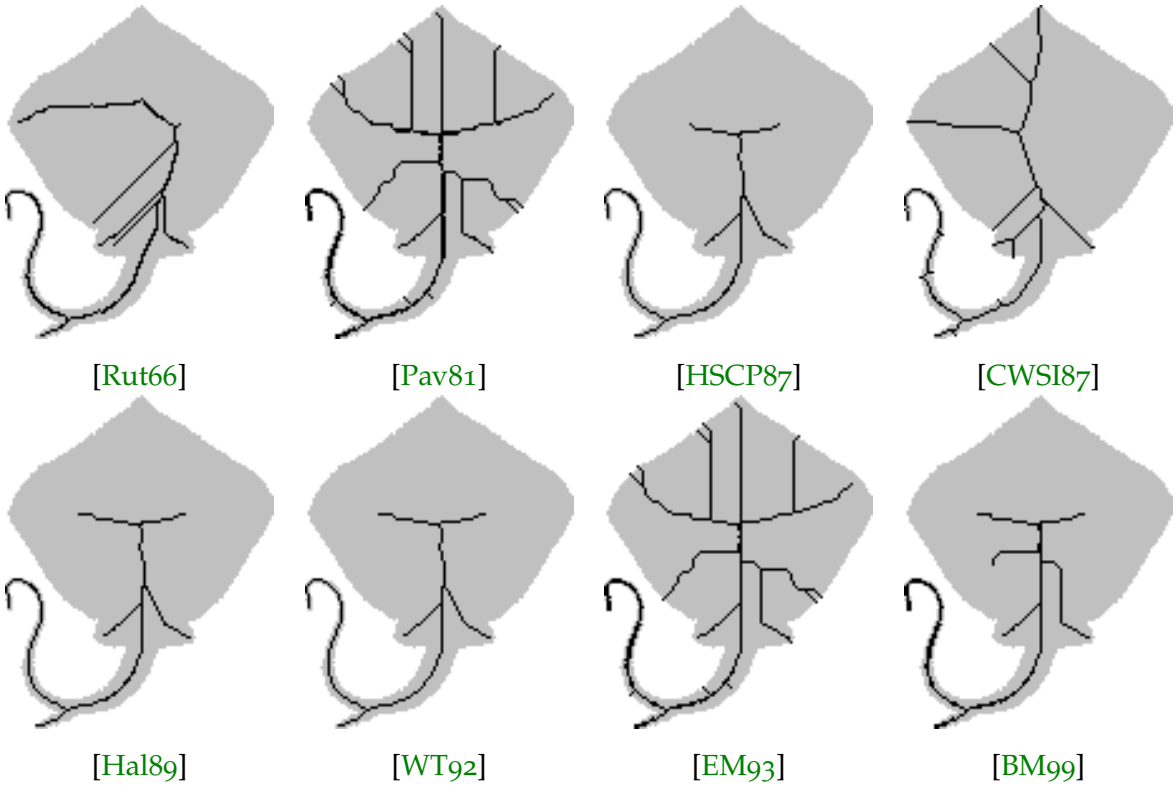


Figure 35: **A few examples of 2d skeletons** obtained with various algorithms. The original shape is shown in grey, the skeleton is in black.

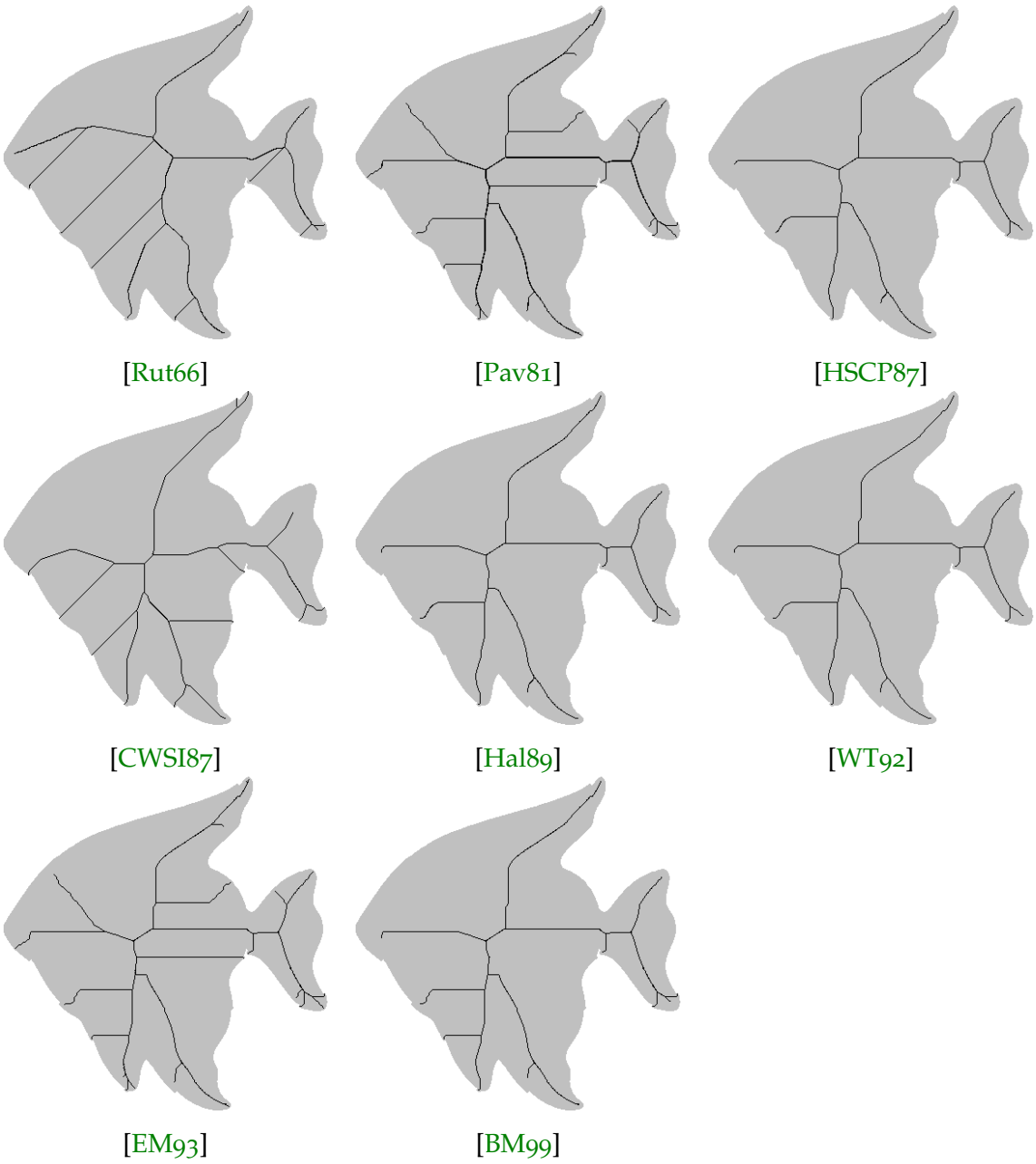


Figure 36: **A few examples of 2d skeletons** obtained with various algorithms. The original shape is shown in grey, the skeleton is in black.

3.3.2.2 THREE DIMENSIONAL SKELETONS

In three dimensions, the detection techniques's literature is globally more "poor" than in the bidimensional framework. Indeed, in most works, high efforts were made in order to produce a good thinning technique, and basic detections of end points (curve points, surface points, ...) is sufficient for obtaining satisfactory skeletons. This part will be therefore shorter than its bidimensional counterpart.

In [Ma95], the author detects and preserves edge points by examining, during the thinning process, each point's neighbourhood and counting the number of object's points in it. For a point $x \in X$, if $\Gamma^*_{26}(x) \cap X$ is a singleton, or if any of the three 3x3

squares centered on x contains only one other point of X, then x is labelled as an edge point. At the end, the author preserves surface and curvilinear parts of an input object (in the edge point definition, the first part is dedicated to curve detection, while the second part is dedicated to surface detection).

In [Ber95a], the author points out the importance of preserving border points in order to preserve medial faces and curves of the object in the skeleton. After characterizing the border points which should be preserved, the author gives a unique and unified characterization of simple border points, making the implementation of his method easy.

The strategy adopted in [SCD97] differs from common strategy in the fact that the author uses two images during his thinning : one image is thinned "layer by layer", and another image, which is regularly updated, is used for detecting which points should be preserved during the current iteration (although, as the author specifies in his article, an implementation using a single image is possible). The result might contain thick elements, so the author performs an extra thinning iteration using no constraint set in order to obtain a thin skeleton.

Most works on parallel thinning are based on the use of masks (see Sec.3.2.2), and many authors propose masks with a double purpose: on one hand, the masks allow to remove simple sets of points from the object, and on the other hand, they allow to preserve interesting features in the final skeleton ([MBPL99], [PK98], [Pal02], [Pal07a]).

Recent works based on the cubical complexes framework have been achieved in order to propose dynamic detection of interesting points during thinning. In [BC06], the authors propose, thanks to a new characterization of simple points based on critical kernels and to the usual definitions of end points and surface points, a new definition of isthmuses which, paired with a new thinning technique, gives very good visual results. Finally, in [JBC07], the authors propose a dynamic end points detection which requires two filtering parameters from the user (one for filtering curvilinear elements, one for filtering surface elements).

On Fig. 94 p.168, we show some examples of 3d skeletons obtained with some of the algorithms previously cited.

4

THE DISCRETE λ-MEDIAL AXIS

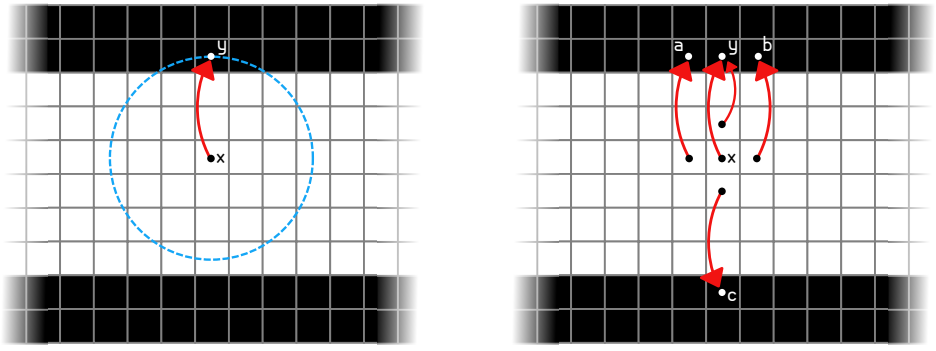
In this chapter, we propose a new discrete filtered medial axis, derived from the λ-medial axis, initially defined by Chazal and Lieutier in the continuous framework ([CL05]). This axis was proved to be stable under small perturbations of the object's border (except for "critical" values of the parameter). After giving a definition of the λ-medial axis in the discrete framework, we define another axis similar to the discrete version of the λ-medial axis, only cheaper and quicker to compute. We then propose an evaluation of our axes' stability under noise and rotation (in 2d and 3d) and we compare our axes with other well-known axes. We also explain how to use this new discrete medial axis in order to perform homotopic thinning (see Sec. 4.3), so as to obtain a skeleton with good reconstruction capacity.

The work exposed in this chapter was presented during the DGCI 2009 conference, held in Montreal (Canada). A conference article was published in [CCT09], and an extended version was accepted by Pattern Recoginition Letters [CCT10].

Contents

4.1 THE DISCRETE λ-MEDIAL AXIS (DLMA)

The λ-medial axis was proposed by Chazal and Lieutier in a continuous framework. Adapting plainly definition 3.3.2 to the discrete framework would consists in declaring, in Def.3.3.2, $S = \mathbb{Z}^n$. Consider a horizontal ribbon in $\mathbb{Z}^2$ with constant even width and infinite length (as depicted on Fig. 37a). Every point of this set has a projection on its complementary that is reduced to a singleton. Hence, if we keep the same definition, any λ-medial axis of this object with $\lambda > 0$ would be empty. Thus, a plain adaptation of the λ-medial axis to the discrete framework does not produce interesting results.



(a) (b)

Figure 37: **The λ-medial axis cannot be plainly transcripted to discrete spaces -** Consider a ribbon (in white), with infinite length and 6 pixels high. **(a)** The projection of the point x is the point y: by symmetry, all points of the ribbon would have a singleton as projection, producing a empty λ-medial axis for any $\lambda > 0$. **(b)** The extended projection of the point x is the set $\{y, a, b, c\}$: by symmetry, the points located near the center of the ribbon would all have an extended projection consisting of four points, leading to a more "interesting" λ-medial axis.

4.1.1 THE EXTENDED PROJECTION OF A POINT

In order to address this problem, we use the *extended projection* (see [CCZ07]) of a point. For each point $x \in \mathbb{Z}^n$, we define the *direct neighborhood of* x as $N(x) = \{y \in \mathbb{Z}^n \mid d(x,y) \leqslant 1\}$ (in 2d, $N(x) = \Gamma_4(x)$, and in 3d, $N(x) = N(6)(x)$). The direct neighbourhood comprises $2n+1$ points.

Definition 4.1.1 *([CCZ07]) Let $X \subseteq \mathbb{Z}^n$, and let $x \in X$. The* extended projection of x on $\overline{X}$, *denoted by $\Pi^e_{\overline{X}}(x)$, is the union of the sets $\Pi_{\overline{X}}(y)$ (defined in Sec 9.1), for all y in $N(x)$ such that*

$$d(y, \overline{X}) \leqslant d(x, \overline{X}) \tag{4.1}$$

Using the extended projection allows us to cope with the problem pointed out previously. Considering the example of an horizontal ribbon in $\mathbb{Z}^2$ with constant even width and infinite length, points close to the "middle" of the ribbon will each have an extended projection consisting of four points (see Fig. 37b). Informally, the extended projection permits to "see" both sides of the object from a point x close to the "middle"

of the object, whereas only one side is taken into account when considering only the projection of x.

4.1.2 DEFINITION OF THE DISCRETE λ-MEDIAL AXIS (DLMA)

We give the definition of the discrete λ-medial axis (DLMA), using the same notations as the ones used in Sec. 3.3.1.3.

Definition 4.1.2 *Let X be a finite subset of $\mathbb{Z}^n$, and let $\lambda \in \mathbb{R}^+$. We define the function $\mathcal{F}_X$ which associates, to each point x of X, the value $\mathcal{F}_X(x) = R(\Pi^e_{\overline{X}}(x))$.*

The discrete λ-medial axis (or DLMA) of X *is the set of points x in X such that $\mathcal{F}_X(x) \geqslant \lambda$.*

As illustrated in Fig. 38, equation 4.1 (see Def. 4.1.1) avoids producing multiple medial axis points when only one is sufficient; in other words, it yields a thinner axis.

Example Consider the object X (white pixels) depicted in Fig. 38a, and the two pixels x, y in X. Fig. 38b shows the squared Euclidean distance map of X, that is, the value in each pixel is the square Euclidean distance between this pixel and the nearest background pixel. We can see that $d(x, \overline{X}) = 4$ and $d(y, \overline{X}) = 5$. The projection of x (resp. y) on $\overline{X}$ is $\{b\}$ (resp. $\{d\}$). We have $\Pi^e_{\overline{X}}(x) = \{a, b, c\}$, and $\Pi^e_{\overline{X}}(y) = \{d, b, c\}$.

The function $\mathcal{F}_X$ is displayed in Fig. 38c and the 4-medial axis of X (threshold of $\mathcal{F}_X$ for $\lambda = 4$) is displayed in Fig. 38d.

In Fig. 38e, we show how the function $\mathcal{F}_X$ would look like if we suppress condition 4.1 from the definition 4.1.2. In this case, the extended projection of x would be $\{a, b, c, d\}$ instead of $\{a, b, c\}$, therefore giving higher value of $\mathcal{F}_X(x)$. Figure 38f shows that, in this case, the 4-medial axis of X would be different (thicker).

On Fig. 39 (right), we show two examples of DLMAs of a shape. More examples are given in Fig. 40 and Fig. 41. Notice that DLMA in general does not exhibit the same topological characteristics as the original shape.

4.1.3 DLMA VS. GIMA

There are indeed some links between the GIMA (see Def. 3.3.3) and the DLMA. In 2D, in the case of a point x that is, conceptually, a "regular medial axis point" (neither a branch extremity nor a branch junction), the first condition of Def. 3.3.3 p.80 is similar to the condition $R(\Pi_{\overline{X}}(x)) \geqslant \lambda$ in Def. 3.3.2 p.80.

Condition 2 of Def. 3.3.3 plays a role analogous to condition $d(y, \overline{X}) \leqslant d(x, \overline{X})$ in the Def. 4.1.1 of the extended projection, that is to get a thinner axis.

However, there are sensible differences in the results of the DLMA and the GIMA transformations (see Fig. 41). For example, in definition 3.3.3, $\Pi'_{\overline{X}}(x)$ is the only element of the projection $\Pi_{\overline{X}}(x)$ that is taken into account into the computation of the integer medial axis of X, while all the elements of $\Pi^e_{\overline{X}}(x)$ are taken into account into the computation of the discrete lambda medial axis. Moreover, in definition 3.3.3, the result depends on the longest distance between two elements of the (reduced) projection, while in definition 4.1.2, the result depends on the size of the smallest ball including the whole projection.

Generally speaking, one can say that the lambda medial axis "takes into account" more elements than the integer medial axis. We analyse quantitatively the consequences

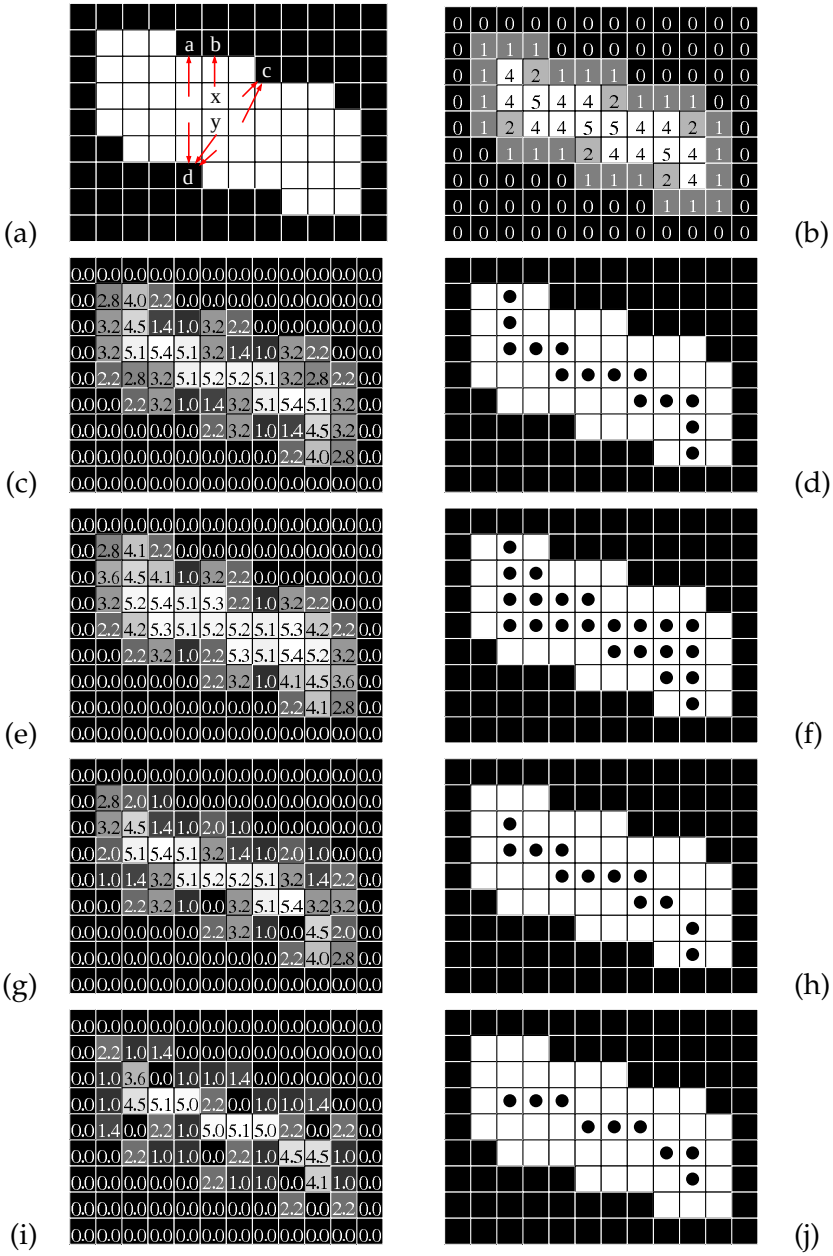


Figure 38: **(a)**: An object X (white pixels). We have $\Pi^e_X(x) = \{a, b, c\}$, and $\Pi^e_X(y) = \{d, b, c\}$. **(b)**: The squared Euclidean distance map of X. **(c)**: The function $\mathcal{F}_X$ (truncated to the first decimal). **(d)**: The λ-medial axis of X, for $\lambda = 4$ (black disks). **(e,f)**: Same as (c,d) assuming that equ. 4.1 is omitted from the definition of the extended projection (Def. 4.1.1). **(g)**: The function $\mathcal{F}'_X$ (truncated to the first decimal). **(h)**: The λ′-medial axis of X, for $\lambda' = 4$ (black disks). **(i,j)**: Same as (g,h) for the integer medial axis.

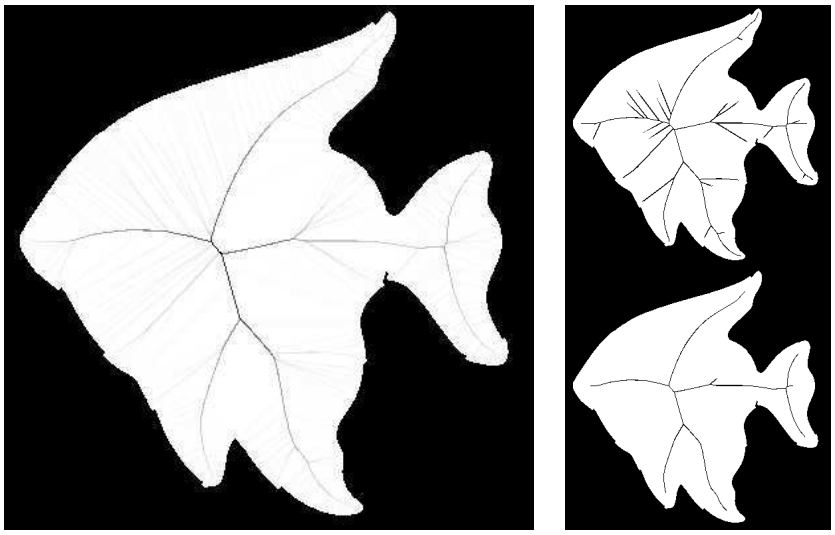


Figure 39: **Example of DLMA in 2d -** Left: The function $\mathcal{F}_X$ superimposed to the shape X. Darkest colours represent highest values of $\mathcal{F}_X(x)$. Any DLMA of X is a threshold of this function at a particular value λ. Top right: discrete 10-medial axis. Bottom right: discrete 30-medial axis of X.

of these differences in Sec. 4.4. These differences lead to different axes, as illustrated on Fig. 38 and on Fig. 46.

4.2 ALGORITHMS - THE DISCRETE λ'-MEDIAL AXIS (DL'MA)

4.2.1 COMPUTING THE DLMA

4.2.1.1 ALGORITHMS

In order to compute, for every $x \in X$, the value of $\mathcal{F}_X(x)$, it is necessary to compute the extended projection of x, and to be able to compute the radius of the smallest enclosing ball of a given set. Thanks to an algorithm proposed by D. Coeurjolly in [CCZ07], the extended projection of a point x can be computed in optimal time and space, that is, in $O(N)$ where $N = \sum_{x \in X} |\Pi^e_X(x)|$.

Thanks to an algorithm proposed by E. Welzl in [Wel91], it is possible to compute the smallest enclosing ball of a set in linear time relatively to the cardinality of this set (see Alg. 24). The reader interested by smallest enclosing ball problem can find a very good overview of the problem in [Men96]. From all the algorithms existing for computing the smallest enclosing ball of a set, we chose the algorithm of Welzl (described in Alg. 24) as it is multi-dimensional, linear and easy to program (thanks to a recursive procedure). Please note that the algorithm takes two parameters as input: in order to call the program, the second parameter should be the empty set while the first parameter should be the considered set of points.

The discrete λ-medial axis of an object can be computed using Alg. 25. Another method for computing the DLMA of a shape X at a value λ_1 consists of computing the map $\mathcal{F}_X$, and then threshold this map at value λ_1. If another DLMA of X needs to be computed at a value λ_2, only a simple threshold of $\mathcal{F}_X$ at value λ_2 would be necessary.

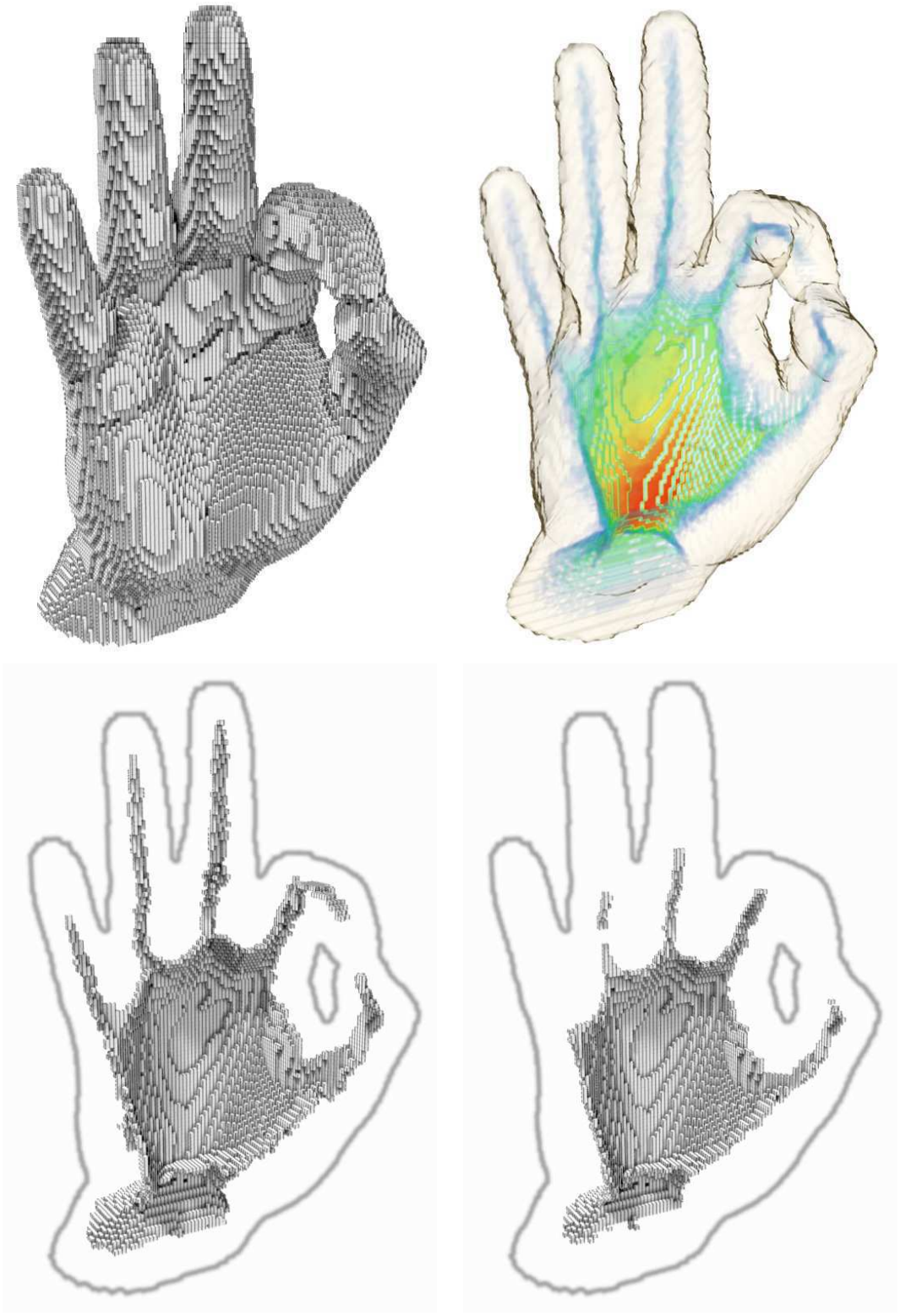


Figure 40: **Example of DLMA in 3d -** Top-right: The function $\mathcal{F}_X$ superimposed to the top-left shape X. Red colour represents highest values of $\mathcal{F}_X(x)$, and blue colour represents lowest. Bottom left: discrete 11-medial axis. Bottom right: discrete 15-medial axis of X.

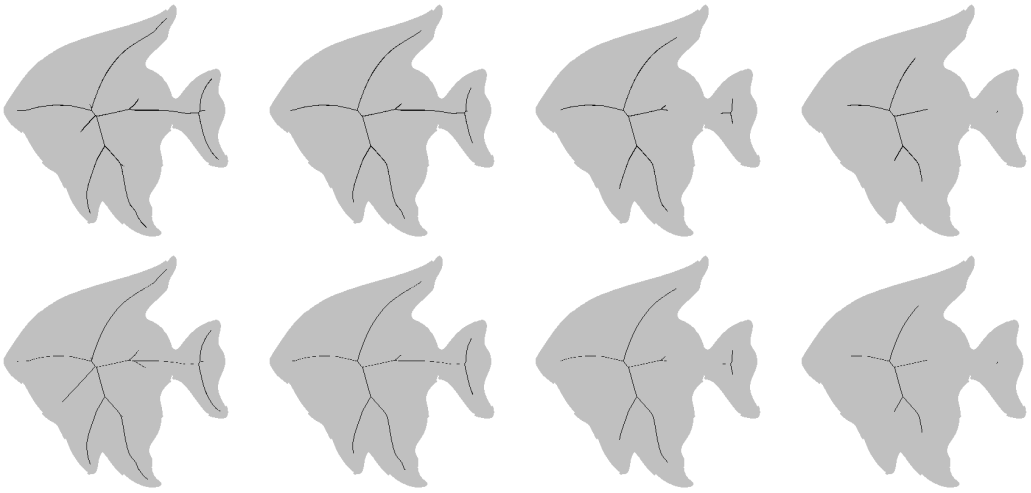


Figure 41: **Examples of DLMA and GIMA -** Results of DLMA (first row) and GIMA (second row) for parameter values yielding similar reconstruction ratios (see Sec. 4.4). From left to right, the normalised residual is 1%, 4%, 8% and 16%.

Algorithm 24: SmallestEnclosingBall(P, R)

Data: Two subsets of points of $\mathbb{R}^d$.
Result: The smallest enclosing d-dimensional ball B containing the finite set $P \cup R$, with constraint that the elements of R should be on the border of B.

```
1  if P == ∅ or |R| == d + 1 then
2  |  compute B directly;
3  end
4  else
5  |  choose an element p ∈ P;
6  |  B = SmallestEnclosingBall(P \ {p}, R);
7  |  if B is defined and p ∉ B then
8  |  |  B = SmallestEnclosingBall(P \ {p}, R ∪ {p});
9  |  end
10 end
11 return B;
```

4.2.1.2 COMPLEXITY

The worst case complexity of Alg. 25 is more than linear when the dimension of the space is equal to 2 or more. Indeed, the complexity of the operation performed on line 3 is proportional to $|S|$. Therefore, for each point x of the input shape, we perform a calculation which takes, to complete, an amount of time proportional to the number of points on the extended projection of x. For this reason, we introduce, in the following, a variant of the DLMA which can be computed in linear time.

Algorithm 25: DLMA(X, λ)

Data: An object $X \subset \mathbb{Z}^d$, and a real value λ.
Result: The discrete λ-medial axis of X.

```
R = ∅;
foreach x ∈ X do
    S = Π^e_X̄(x);
    B =SmallestEnclosingBall(S, ∅);
    if Radius(B) ⩾ λ then
        R = R ∪ x;
    end
end
return R;
```

4.2.2 THE DISCRETE λ'-MEDIAL AXIS (DL'MA)

The discrete λ'-medial axis (DL'MA) is introduced in order to propose a medial axis close to the concept of the DLMA, only with linear complexity. In order to enhance the worst-case complexity of Alg. 25, the computation of the extended projection of a point x on $\overline{X}$ has been modified in order to provide, as result, a subset of $\Pi^e_{\overline{X}}(x)$. The result is called the *sub extended projection*: basically, it consists of replacing, in Def. 4.1.1, the occurrences of $\Pi_{\cdot}(.)$ by $\Pi'_{\cdot}(.)$.

Definition 4.2.1 *Let* $X \subseteq \mathbb{Z}^n$, *and let* $x \in X$. *The* sub extended projection of x on $\overline{X}$, *denoted by* $\Pi^{e'}_{\overline{X}}(x)$, *is the union of the sets* $\Pi'_{\overline{X}}(y)$, *for all* y *in* $N(x)$ *such that* $d(y, \overline{X}) \leqslant d(x, \overline{X})$.

We now define the DL'MA:

Definition 4.2.2 *Let* X *be a finite subset of* $\mathbb{Z}^n$, *and let* $\lambda \in \mathbb{R}^+$. *We define the function* $\mathcal{F}'_X$ *which associates, to each point* x *of* X, *the value* $\mathcal{F}'_X(x) = R(\Pi^{e'}_{\overline{X}}(x))$.
The discrete λ'-medial axis (or DL'MA) of X *is the set of points* x *in* X *such that* $\mathcal{F}'_X(x) \geqslant \lambda$.

Algorithm 26 allows to compute a DL'MA of a shape. In this new algorithm, given $x \in X$, we consider for each element of $N(x)$, only one projection point on $\overline{X}$ (the sub extended projection of x): this point can be found in constant time [Coe03, MQR03, HR08]. The computation of the DL'MA of a shape has therefore a complexity in $o(n.d)$.
In Fig 38g and h, an example of the DL'MA is given.

4.3 TOPOLOGY PRESERVATION

4.3.1 SKELETONISATION ALGORITHM WITH THE λ-MEDIAL AXIS AS CONSTRAINT SET

It is easy to see that the DLMA or the DL'MA of a given shape X does not exhibit, in general, the same homotopy type as X (see for example Fig. 41).
In order to guarantee topology preservation, a popular method [DP81, Vin91, TV92, Pud98] consists of performing an homotopic thinning of X with the constraint of

Algorithm 26: DL'MA(X, λ)

Data: An object $X \subset \mathbb{Z}^d$, and a real value λ.
Result: The discrete λ'-medial axis of X.

```
R = ∅;
foreach x ∈ X do
    S = Π^{e'}_X(x);
    B = SmallestEnclosingBall(S, ∅);
    if Radius(B) ⩾ λ then
        R = R ∪ x;
    end
end
return R;
```

retaining the points of its filtered medial axis M, that is to say, of iteratively removing simple points [KR89, Ber94, CB09] from X that do not belong to M. The thinning algorithm used can be, for example, the one presented on Alg. 3, p.56.

Removal order has an important effect on the final result, and a defined order must be precisely set during the thinning in order to obtain a satisfying (centered) result. In Alg. 3, the removal order is defined by a map $\mathcal{P} : X \to \mathbb{R}$, associating to each point of the input object a value in $\mathbb{R}$. Simple points are then removed based on the map, points with lowest map value being removed first. In the general case, the choice of this priority function is not obvious (see [TV92, CCZ07]).

An example is shown on Fig. 42, where the filtered DLMA of an object X is used as a constraint set during skeletonization. Choosing the Euclidean distance transform of X as a priority function for removing simple points from the object may lead to geometric distortions (see [TV92]). In some cases, "extra branches", not corresponding to any geometric feature of the original object, may even appear, as shown on Fig. 42c. In our example, the skeletonization algorithm "reached too quickly" the elements of the constraint set (see Fig. 43a), which could not be removed, and continued removing elements around.

Choosing the map $\mathcal{F}_X$ as priority function yields more satisfying results on Fig. 42d, as it guides the thinning process towards elements that belong to the discrete λ-medial axes (see Fig 43b). Although this solution works in the 2d case, unfortunately, as we will see in the following, using such map in 3d is generally not a good solution, as it does not, on the contrary of the Euclidean distance map, result in an isotropic removal of simple points.

4.3.2 THE DLMA MAP AND TOPOLOGICAL LUMPS

An object can be generally divided into layers of points: the outer layer is made of points on the border of the object object, the second layer is made of points neighbouring the outer layer, etc. Distance maps are generally used as priority functions for simple points removal because they allow to "simulate" a thinning process "layer by layer": simple points deep inside the object have a higher value than simple points on the border of the object, and the thinning process generally removes all simple points of a layer before removing points from a deeper layer. Removing simple points layer by layer

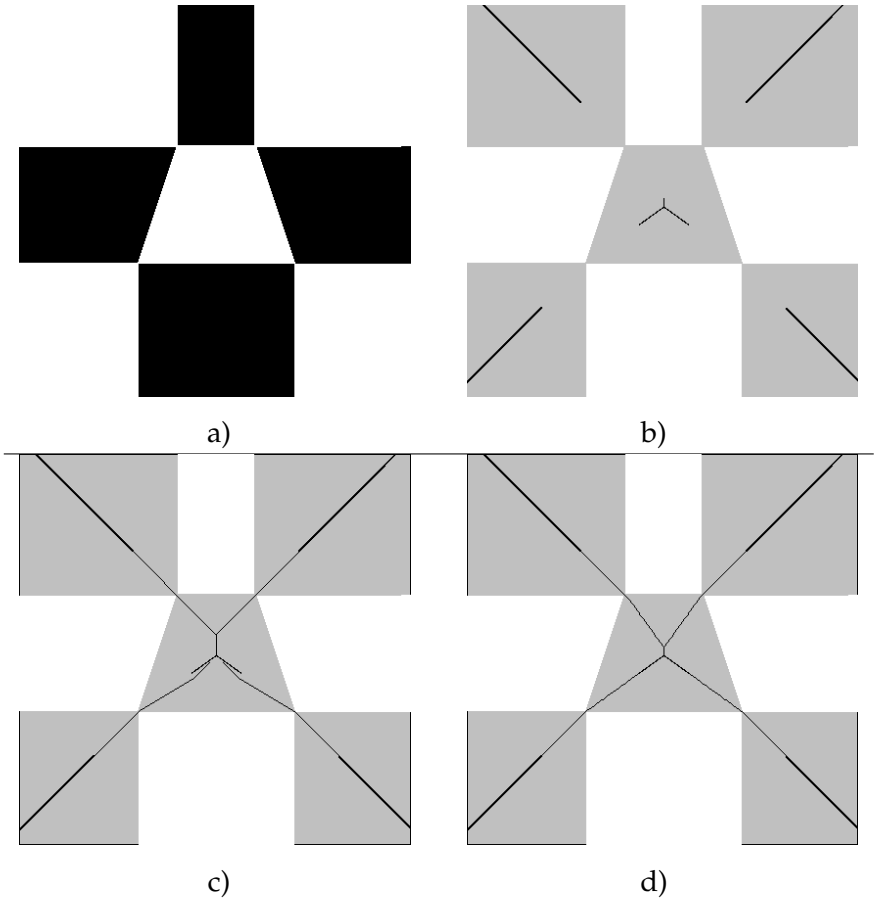


Figure 42: **Problems related to using the DLMA map as a priority function during thinning - (a)** The original object (in white). **(b)** The 50-medial axis of the shape (the medial axis is in black, the original shape appears in grey, the background is in white). **(c)** Result of the skeletonization using the Euclidean distance transform as a priority function, and the 50-medial axis as inhibitor set. **(d)** Result of the skeletonization using the DLMA map as a priority function, and the 50-medial axis as inhibitor set.

yields generally good results (when no constraint set is used) as it "imitates" parallel removal of simple points.

The map $\mathcal{F}_X$ does not offer in general such configuration, and points deep inside the object can have a lower value than points on the border. The result is that the simple points removal will not imitate a "layer by layer" removal, and the process will start digging very deeply a given spot of the object before digging another spot: this can be clearly seen on Fig 43b. In 3d (or more), homotopic thinning using the map $\mathcal{F}_X$ as a priority function may result in a skeleton containing *lumps* (see [PCB08]). A lump is defined as a set of voxels P, homotopic to a subset $Q \subset P$, and containing no simple point. Indeed, in order to obtain Q from P by an homotopic operation, it is necessary to have a step performing homotopic enlargement of P (add simple point(s) to P); this step can also be seen as an homotopic thinning of $\overline{P}$. The Bing's house [Bin64] is a famous example of lump: it contains no simple points, and it is homotopy equivalent to a single voxel.

An example is given on Fig 44, where the object X (Fig 44a) is skeletonized using the map $\mathcal{F}_X$ as a priority function, and the filtered DLMA as constraint set. The resulting

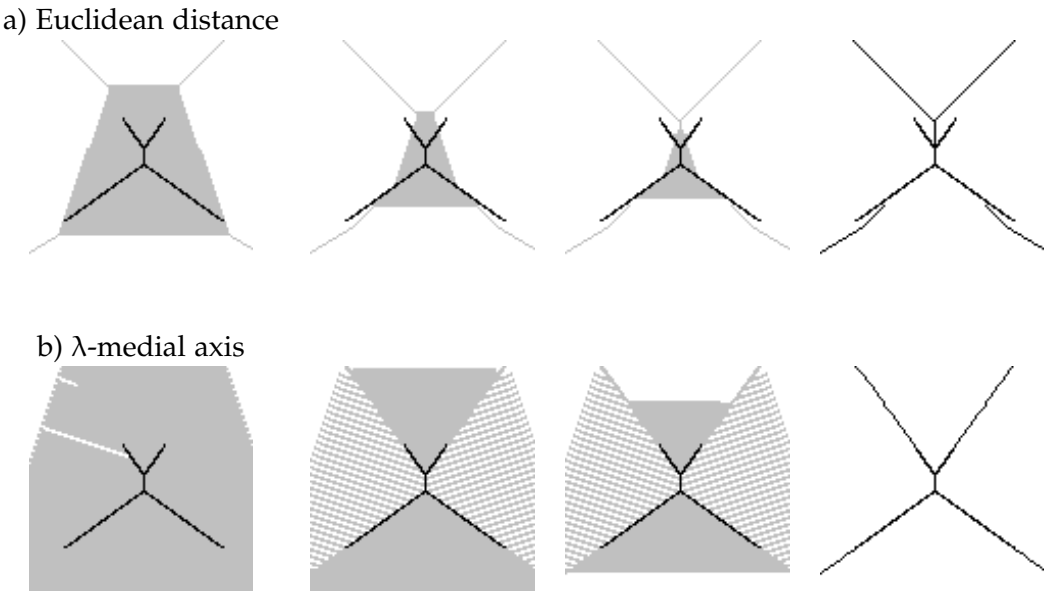


Figure 43: **Problems related to using the DLMA map as a priority function during thinning (close-up) - a)** : A detailed view of the center of the shape presented in Fig 42, while performing thinning using the Euclidean distance transform as a priority function, and the 50-medial axis as inhibitor set (the shape is in grey, the inhibitor set is in black). Images from left to right show the progress of the thinning. **b)** : Same as sub figure a, but for the thinning process using the DLMA map as a priority function, and the 50-medial axis as inhibitor set.

skeleton (Fig 44d) contains many lumps, looking like closed surface patches. These lumps are actually subsets of voxels, homotopic to a single voxel and containing no simple voxel. Performing the skeletonization with the exact Euclidean distance map as priority function results in a skeleton containing no lumps (see Fig 44c): removing simple points "layer by layer" reduces the chances of obtaining lumps in the skeleton. A further analysis using methods proposed in [PCB08] reveals that some of these lumps can be reduced by removing simple pairs of points. We may point out that $\mathcal{F}_X$ can be used for thinning in 2d, as "topological lumps" do not appear in this dimension.

4.3.3 ANOTHER PRIORITY FUNCTION FOR THINNING WITH THE λ-MEDIAL AXIS IN 3D

At the time this thesis is written, finding a way to use the map $\mathcal{F}_X$ for homotopic 3d thinning is still under investigation. A solution was recently tested: it consists in choosing a threshold value t for the map $\mathcal{F}_X$, such that any value under t in $\mathcal{F}_X$ is considered as irrelevant and corresponding voxels will be removed "layer by layer", using the Euclidean distance transform. Points x such that $\mathcal{F}_X(x)$ is superior to t are removed based on their value. Typically, this task can be achieved with only one map as priority function: let D_X be the Euclidean distance transform of X, let $m = \max_{x \in X} D_X(x)$, we set

$$E_X(x) = \begin{cases} \mathcal{F}_X(x) + m & \text{if } \mathcal{F}_X(x) > t \\ D_X(x) & \text{else} \end{cases}$$

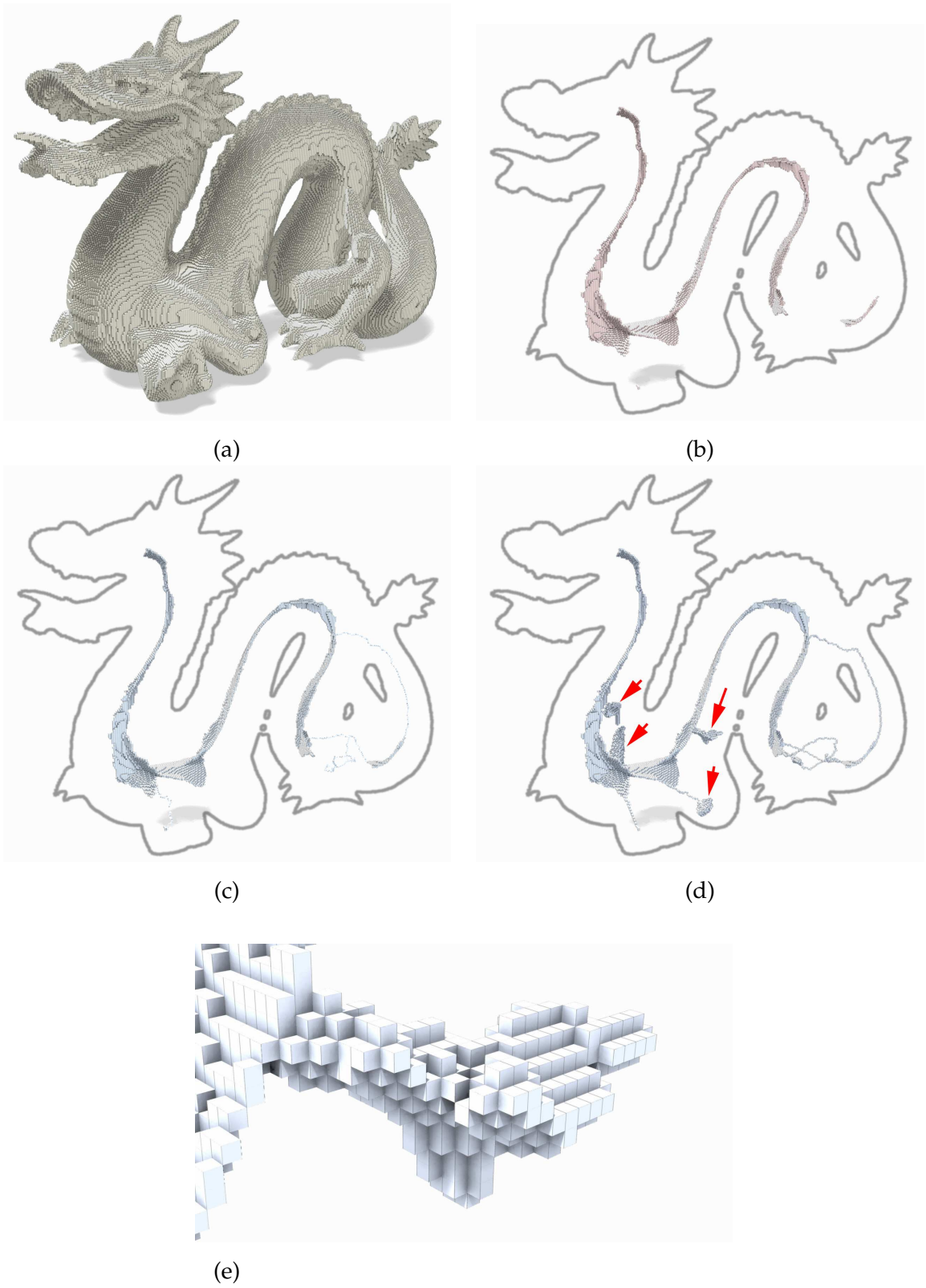
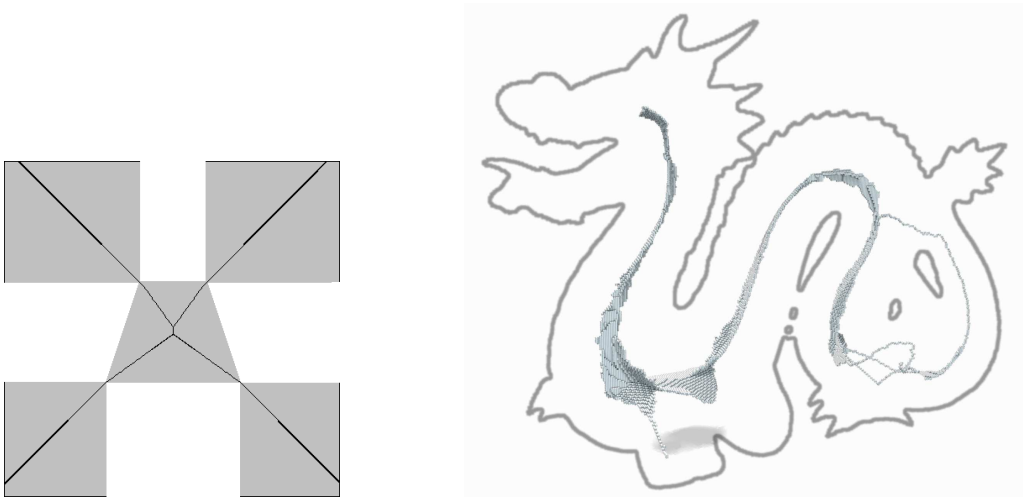


Figure 44: **Homotopic thinning of the Stanford dragon with various priority functions and a DLMA as constraint set - (a)** : The original image, *"Stanford dragon"*. **(b)** : The λ-medial axis of the shape a), for $\lambda = 35$. **(c)** : Result of the homotopic thinning of the shape (a) using (b) as an inhibitor set and the Euclidean distance transform of (a) as a priority function. **(d)** : Same as (c), but using the DLMA map as a priority function. Some "lumps" appear on the resulting skeleton, outlined by the arrows. **(e)** : Close view of the top-right lump of (d). This group of voxels does not contain any simple voxel, but is homotopic to a single voxel.

When the parameter t is well chosen, the homotopic thinning using E_X as a priority function, and a λ-medial axis as a constraint set gives good results: no extra branches appear on the skeleton of the tested shapes, and no lumps neither. Thanks to the map E_X, some points of the shape are removed with a "layer by layer" strategy, while others are removed based on the values of $\mathcal{F}_X$: this strategy seems to be sufficient for avoiding lumps in the resulting skeleton, and for guiding the thinning towards elements of the constraint set.



(a) (b)

Figure 45: **Homotopic thinning with our new priority function and a DLMA as constraint set - (a)** : The result of the homotopic thinning of Fig. 42a, using the 50-medial axis as inhibitor and the map E_X as priority function. **(b)** : The result of the homotopic thinning of Fig. 44a, using the 35-medial axis as inhibitor and the map E_X as priority function.

4.4 RESULTS AND COMPARISONS

In this section, we compare the stability of DLMA, DL'MA, GIMA, and the well-known *filtered Euclidean Medial Axis (EMA)*, defined as the set of the centers of maximal balls that have a radius greater than a parameter ρ (see Sec. 3.3.1, p.76). We will see how these axes behave to noise or rotation, in 2d and 3d.

Let us give first some definitions that are useful for describing our experiments and analysing their results. Let X be a finite subset of $\mathbb{Z}^n$, and Y be a subset of X; we set $\mathrm{REDT}_X(Y) = \bigcup_{y \in Y} B^<(y, d(y, \overline{X}))$. The set $\mathrm{REDT}_X(Y)$, sometimes called *reverse Euclidean distance transform* [Coe03], tells us how much information of X can be retrieved from Y.

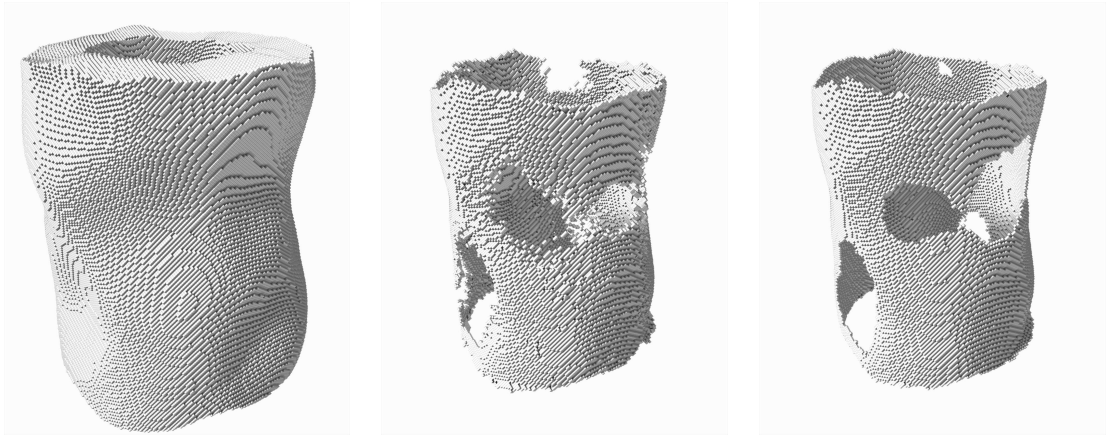
It is well known that any object can be fully reconstructed from its medial axis, more precisely, we have $X = \mathrm{REDT}_X(M)$ whenever M is the (exact and non-filtered) medial axis of X. This property holds if M is the set of the centers of all maximal balls of X (EMA of X with $\rho = 0$), the DLMA of X with $\lambda = 1$, the DL'MA of X with $\lambda' = 0$ (of course, as it is equal to X), or the integer medial axis of X with $\gamma = 1$. However, it is no longer true if we consider filtered medial axes, eg. EMA, DLMA, DL'MA or GIMA with arbitrary ρ, λ, λ' or γ.

It is interesting to measure how much information about the original object is lost when we raise the filtering parameter of an axis. We set

$$\mathcal{L}_X(\lambda) = \frac{|X \setminus \mathrm{REDT}_X(\mathrm{LM}_X(\lambda))|}{|X|}$$

where $\mathrm{LM}_X(\lambda)$ is the DLMA of X. In words, $\mathcal{L}_X(\lambda)$ is the normalized area of the difference between X and the set reconstructed from its DLMA. We call $\mathcal{L}_X(\lambda)$ the *(normalised) residual* of the DLMA of X filtered at value λ. The same way, we define $\mathcal{L}'_X(\lambda')$ (resp. $\mathcal{I}_X(\gamma)$, $\mathcal{M}_X(\rho)$) as the (normalized) residuals of the DL'MA (resp. GIMA, EMA) of X filtered at value λ' (resp. γ, ρ). Residuals bring a numerical evaluation of how much an object can be reconstructed from its filtered medial axis: the lower the value, the more can be reconstructed.

Figure 47 shows the evolution of the residuals for the EMA, GIMA, DLMA and DL'MA versus the filtering value. Since differences are not negligible, to ensure a fair evaluation we compare the results of methods for approximately equal values of their residuals, rather than for equal values of their parameters. Therefore, in the following, we compare DLMA, DL'MA, GIMA and EMA for filtering parameter λ, λ', γ and ρ yielding (approximately) a same residual ($\mathcal{L}_X(\lambda) = \mathcal{L}'_X(\lambda') = \mathcal{I}_X(\gamma) = \mathcal{M}_X(\rho)$). An example of a filtered DLMA and a filtered GIMA giving the same residual is presented on Fig. 46.



Object GIMA DLMA

Figure 46: **Comparison of DLMA and GIMA in 3d -** On this example, the GIMA and the DLMA of the leftward shape were filtered in order to yield a residual of 5%. Here, the best aspect preservation is achieved by the DLMA.

Globally, stability of the medial axes will be tested by measuring the "difference" between the filtered medial axis of a shape X, and the filtered medial axis (same filtering parameter) of $T(X)$, where T is a transformation on X (such as noise addition). The "difference" will be measured using the Hausdorff distance (see below) or a dissimilarity measure proposed by [DJJ94]. The drawback of Hausdorff distance for measuring shape dissimilarity is its extreme sensibility to outliers, the latter measure performs better in this respect.

Let X, Y be two subsets of $\mathbb{R}^n$. We set

$$H(X|Y) = \max_{x \in X}\{\min_{y \in Y}\{d(x,y)\}\}$$

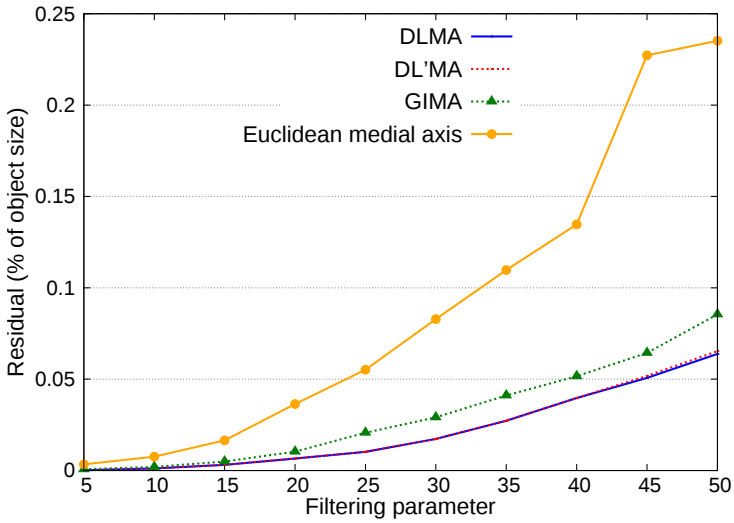


Figure 47: **Residuals vs. filtering parameter -** Residuals $\mathcal{L}_X(\lambda)$ (2 variants) and $\mathcal{I}_X(\gamma)$, for the set X depicted in Fig. 39. Horizontal axis: the value of the parameter (λ or γ). Vertical axis: the value of the residual. Notice that curves corresponding to DLMA and DL'MA are superimposed.

and $d_H(X,Y) = \max\{H(X|Y), H(Y|X)\}$ is the *Hausdorff distance between* X *and* Y. We set

$$D(X|Y) = \frac{1}{|X|}\sum_{x\in X} \min_{y\in Y}\{d(x,y)\}$$

and $d_D(X,Y) = \max\{D(X|Y), D(Y|X)\}$ is the *Dubuisson and Jain's dissimilarity measure between* X *and* Y (called *dissimilarity* in the sequel for the sake of brevity). We point out the fact that the Dubuisson and Jain's dissimilarity does not satisfy the properties of a distance function.

We conducted our experiments on a database of 216 two-dimensional shapes, divided into 18 classes, provided by B. Kimia [SCTK98], and 20 three-dimensional objects gathered on the Internet or created ourselves. Figure 48 shows one (reduced) image of each class of the bidimensional shapes, and a sample of the 20 three-dimensional objects.

4.4.1 STABILITY TO NOISE

Medial axes are notoriously sensitive to border noise. Since the λ-medial axis is supposed to cope reasonably well with shape deformation, it is useful to test how it fares in practice.

To introduce noise to the boundary of an object we propose deforming it using a process derived from the Eden accretion process [Ede61]. The Eden process, in its simplest form, deforms a shape by randomly choosing points on the shape's border and adding them to the shape (each border point have the same probability to be chosen). In spite of its simplicity, the Eden process exhibits good asymptotic isotropy [LC94]. In our process, we require accretion steps to concern only simple points. This way, at each step, the object's homotopy type remains unchanged. Moreover, only point addition is performed: points removal is avoided as it is a deformation which will affect regular values of the medial axis.

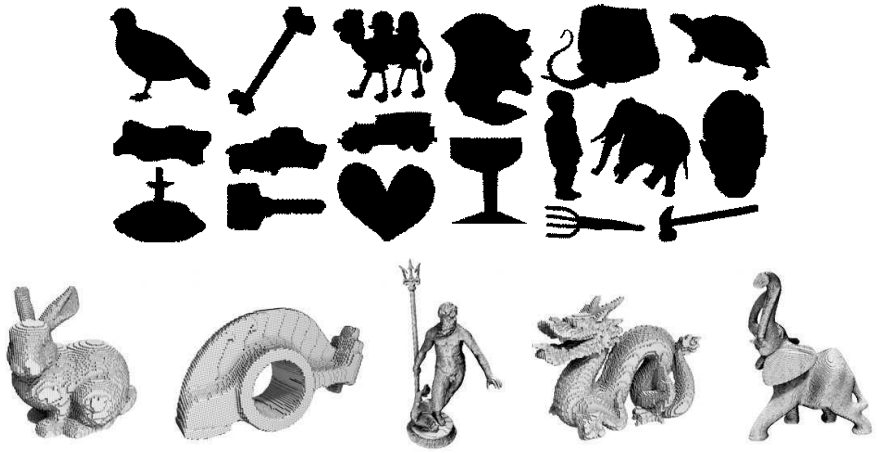


Figure 48: A sample of the 216 shapes of Kimia's database, and of the 3d shapes from our database.

We denote by $E(X, n)$ the result of applying n steps of this process to the shape X. In this experiment, we compare the (filtered) medial axis of an original shape with the one of a deformed shape.

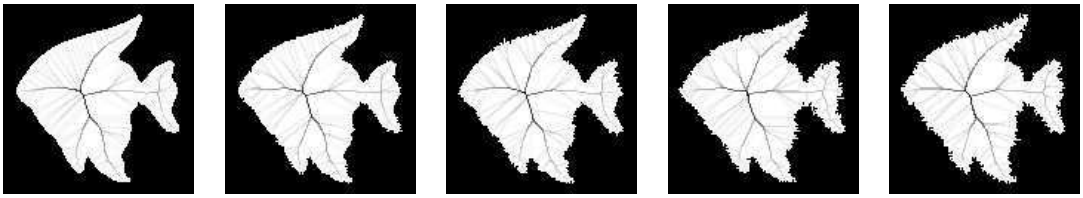


Figure 49: **Illustration of the Eden process -** Illustration of the border deformation process, after 0 (original image), 50, 100, 150 and 200 steps. The function $\mathcal{F}_X$ is superimposed to each shape X.

In Table 1 (resp. Table 2) we give the average Hausdorff distance (resp. dissimilarity) between $M(X)$ and $M(E(X, n))$ on the 216 shapes of Kimia's database and the 20 shapes of our 3d database, for different definitions of M (filtered EMA, GIMA, DLMA or DL'MA of X). Filtering parameters ρ, γ, λ and λ' were chosen in order to give a normalised residual varying from 5 to 30 in 2d, and a normalised residual varying from 3 to 10 in 3d. Results are given for values of n equal to 3%, 6% and 9% of object's surface/volume (values shown are averages of results obtained for all tested residuals), and the last column indicates all results averaged together.

Note that the tested residual values are not the same in the 2d and 3d case. Indeed, in 3d, choosing a residual superior to 10 usually gives a medial axis which contains very little information about the original object (surfacic parts are mostly deleted). We constrained our study to medial axes holding enough information on the original objects, and ignored normalised residual superior to 10 in the 3d case.

On Fig. 50 and 51, we show the evolution of the Hausdorff distance and the dissimilarity between $M(X)$ and $M(E(X, n))$ against the quantity of noise added to the shape,

	2D				3D			
Noise	3%	6%	9%	all	3%	6%	9%	all
EMA	6.97	8.78	9.88	7.58	7.07	7.42	7.72	6.92
GIMA	6.30	8.05	8.97	6.77	7.75	8.46	8.77	7.56
DLMA	5.61	7.57	8.32	6.23	5.17	6.01	6.78	5.36
DL'MA	5.89	8.05	9.00	6.66	4.92	5.97	7.72	5.61

Table 1: Average Hausdorff distance between $M(X)$ and $M(E(X, n))$. Noise level (parameter n) is expressed as a percentage of object area, results shown are averages of all tested residuals. Lowest values are highlighted in grey.

	2D				3D			
Noise	3%	6%	9%	all	3%	6%	9%	all
EMA	1.01	1.42	1.75	1.21	0.39	0.52	0.60	0.44
GIMA	1.32	1.83	2.20	1.53	0.80	0.99	1.11	0.85
DLMA	1.02	1.51	1.75	1.23	0.28	0.37	0.43	0.31
DL'MA	1.12	1.66	1.94	1.36	0.28	0.36	0.44	0.31

Table 2: Average dissimilarity between $M(X)$ and $M(E(X, n))$. Noise level (parameter n) is expressed as a percentage of object area, results shown are averages of all tested residuals. Lowest values are highlighted in grey.

for DLMA, DL'MA, GIMA and EMA algorithms. The results shown are average of the results obtained on the 216 shapes of Kimia's 2D image database. Figure 52 shows the same experimental results on the 20 shapes of our 3d database.

As we expected, DLMA fares better on the average than other axes, with regard to Hausdorff distance criterion. Its faster variant DL'MA ranks second. According to the dissimilarity criterion, DLMA and DL'MA yield the best results in 3d, and in 2d, DLMA performs nearly as well as EMA.

4.4.2 ROTATION INVARIANCE

Rotation invariance is an important property of the Euclidean medial axis that holds in the continuous framework. If R_θ denotes the rotation of angle θ about the origin, and M denotes the Euclidean medial axis transformation, the rotation invariance property states that $M(R_\theta(X)) = R_\theta(M(X))$, irrespective of X and θ.

In a discrete framework, this property holds only in particular cases (eg. when θ is a multiple of 90 degrees). Nevertheless, we can experimentally measure the dissimilarity between $M(R_\theta(X))$ and $R_\theta(M(X))$ for different instances, and different definitions of the medial axis. The lower this dissimilarity, the more stable under rotation the method is.

For each shape of Kimia's database and of our database, we computed the Hausdorff distance and the dissimilarity between $M(R_\theta(X))$ and $R_\theta(M(X))$ for values of the

parameters yielding a normalized residual of 5, 10, 15, 20, 25 and 30, and for rotation angles varying from 0 to 89 degrees by 3 degrees steps. In 3d, rotations were successively performed around the Y and Z axis. Moreover, in 3d, only parameters yielding a normalized residual of 3, 5 and 10 were tested: higher residual values yielded skeletons that were not keeping enough information from the original object (see discussion in Sec. 4.4.1).

Figures 53 and 54 show detailed results of such experiment in 2d, while Fig. 55 and Fig. 56 show results in 3d. For both cases, an inverse truncated real rotation algorithm was used to perform images rotation.

The results are summarised in Tables 3 and 4. In Table 3 (resp. Table 4) we give the average Hausdorff distance (resp. dissimilarity) between $M(R_\theta(X))$ and $R_\theta(M(X))$ on the 216 shapes of Kimia's database and on the 20 3d shapes of our database, for θ varying from 0 to 89 degrees (in 3d shapes, rotation was performed independently around Y and Z axes), and for normalized residual values equal to 5, 10, 15, 20, 25 and 30 (for 3d shapes, only 3, 5 and 10 were tested). We show, in the first three columns, the average value obtained for all tested rotation angles for a given residual, while the fourth column indicates the average value obtained for all residuals and all rotation angles.

In 3d, λ′- and λ-medial axis yield, in average, the best results for both measures. In 2d, they are better than other axes with regards to Hausdorff distance, and are only outperformed by EMA with regards to dissimilarity.

	2D				3D			
Residual	10%	20%	30%	all	3%	5%	10%	all
EMA	8,23	8,13	10,71	8.55	4,86	5,71	6,89	5.82
GIMA	6,94	9,00	13,17	8.65	5,24	5,48	6,64	5.79
DLMA	7,60	9,07	10,97	8.28	3,86	4,65	6,59	5.03
DL'MA	7,41	8,67	10,62	8.02	4,14	4,80	7,26	5.40

Table 3: Average Hausdorff distance between $M(R_\theta(X))$ and $R_\theta(M(X))$ (in 3d, rotations shown were performed around the Y and around the Z axis). Results shown are averages of all tested angles. Lowest values are highlighted in grey.

	2D				3D			
Residual	10%	20%	30%	all	3%	5%	10%	all
EMA	1,21	1,34	3,88	1,71	0,72	0,72	0,73	0,72
GIMA	1,50	2,41	6,09	2,27	1,40	1,43	1,52	1,45
DLMA	1,46	1,89	3,72	1,94	0,66	0,66	0,76	0,69
DL'MA	1,37	1,78	3,60	1,85	0,67	0,68	0,78	0,71

Table 4: Average dissimilarity between $M(R_\theta(X))$ and $R_\theta(M(X))$ (in 3d, rotations shown were performed around the Y and around the Z axis). Results shown are averages of all tested angles. Lowest values are highlighted in grey.

4.4.3 COMPUTING TIME

In Fig. 57, we show the results of computing time measurements that we performed on an Intel Core 2 Duo processor at 1.83 GHz. Computing times for the GIMA are the lowest, but DL'MA is only slightly slower (and also linear).

4.5 CONCLUSION

We introduced in this chapter the definition of a discrete λ-medial axis (DLMA), and compared it with the integer medial axis and the Euclidean medial axis. The results of the comparison show that both the DLMA and its linear variant DL'MA provide good robustness to boundary noise and rotation. We also studied how DLMA could be used with an homotopic thinning algorithm in order to provide skeletons with good reconstruction capacities (in other words, containing important visual features from the original shape). We present, on Fig. 58, 59, 60, 61, 62 and 63 some results of homotopic thinning constrained by a filtered DL'MA.

For our experimental study of rotational invariance and border noise robustness, both in 2D and 3D, we introduced an original methodology that ensures a fair comparison between different methods, under the mere assumption that their result decreases in size under the control of a single parameter. This methodology could be of interest for comparing other medial axis filtering approaches.

In the next chapter, we will see that performing homotopic thinning in the digital topology framework does not give, in general, good results for the skeleton decomposition. We will therefore introduce a new framework which will allow to perform easily skeleton decomposition, and we will see how to use the DLMA (or any other medial axis) in this new framework. Finally, we will compare homotopic thinning algorithms, based on the same criteria as the ones used for comparing medial axis in this chapter.

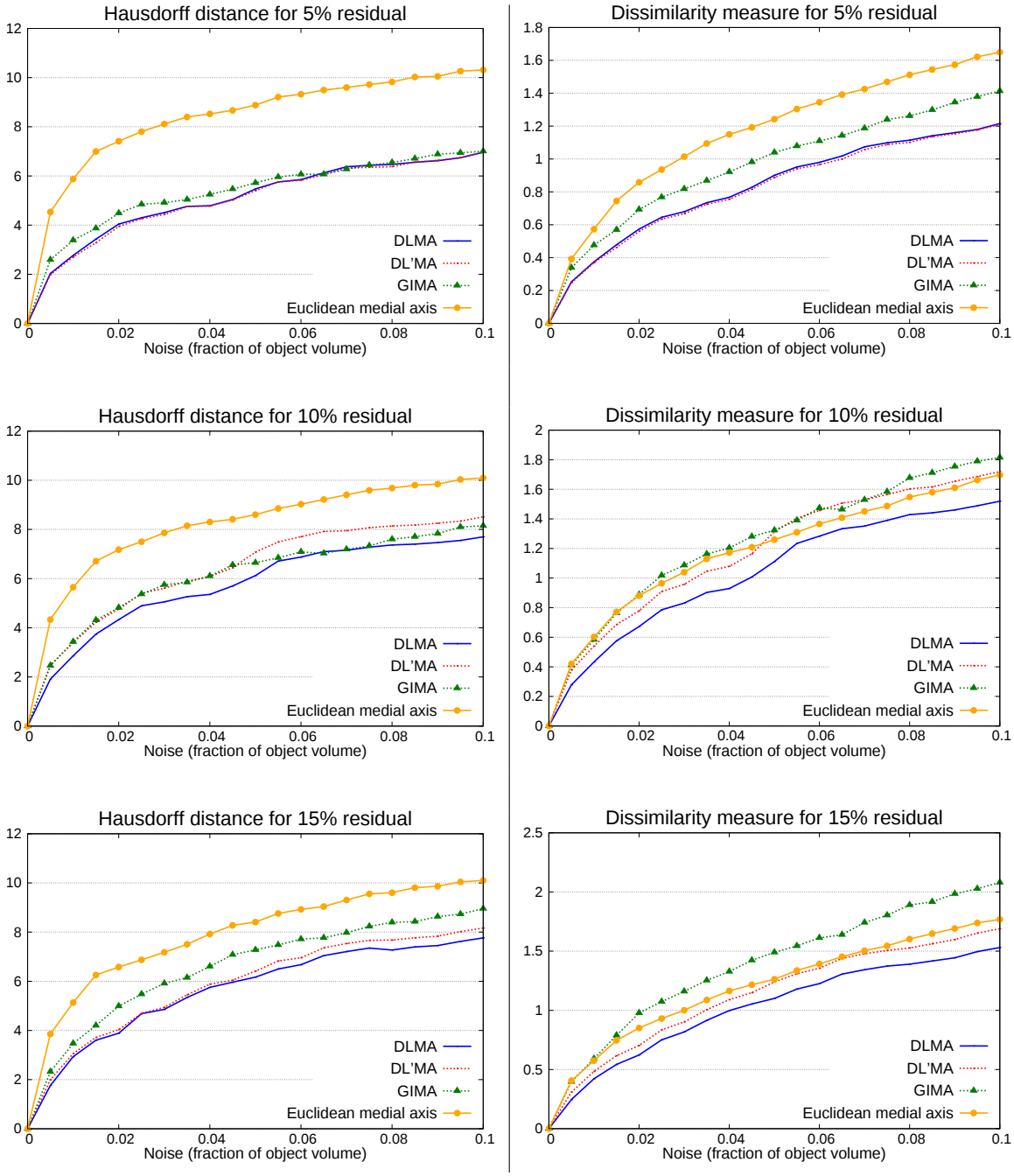


Figure 50: Hausdorff distance (**on the left**) and dissimilarity measure (**on the right**) between $M(X)$ and $M(E(X, n))$ on the 216 shapes of Kimia's 2D image database for normalized residual values equal to 5, 10 and 15. Noise level (parameter n) is expressed as a percentage of object area.

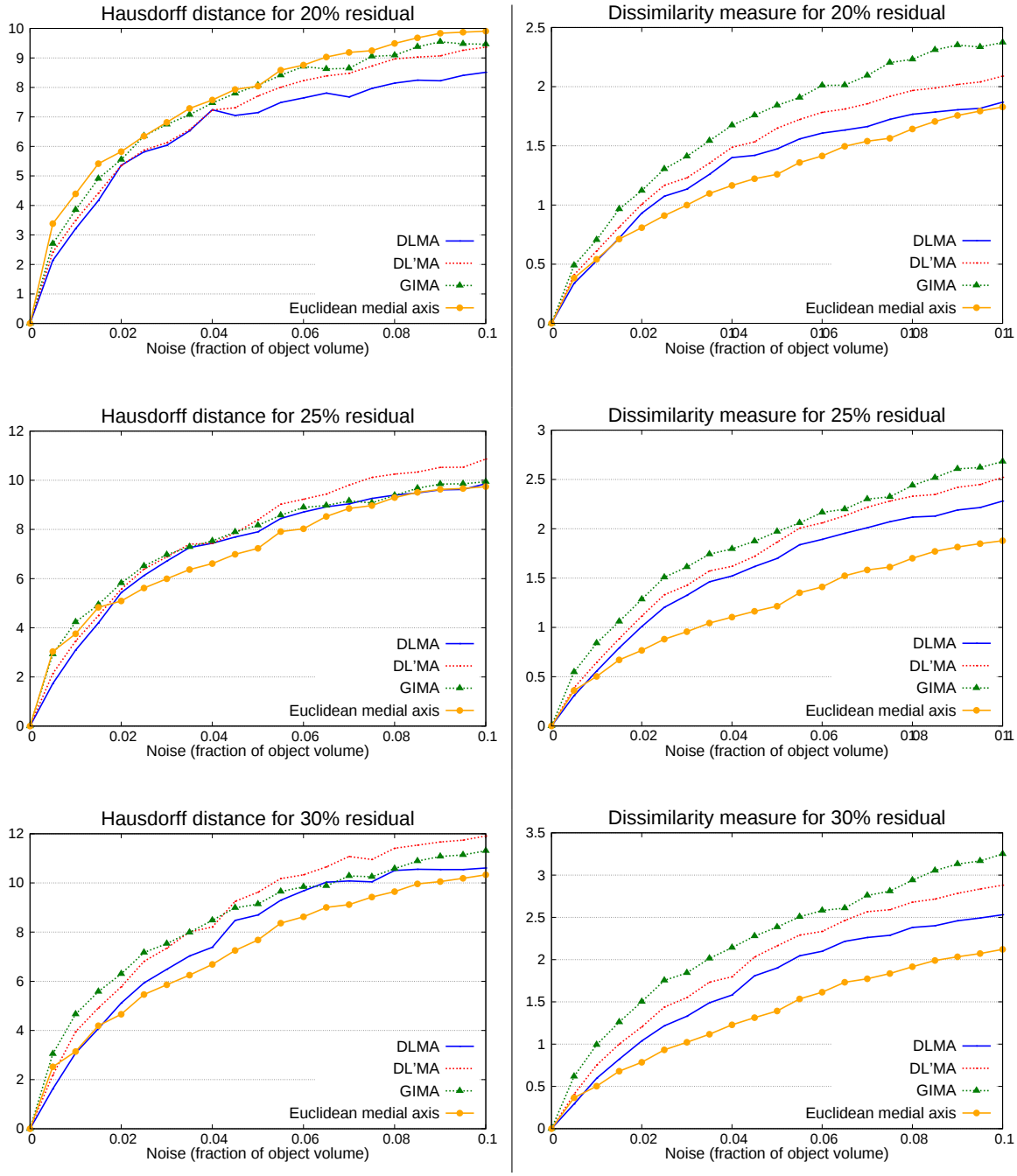


Figure 51: Hausdorff distance (**on the left**) and dissimilarity measure (**on the right**) between $M(X)$ and $M(E(X, n))$ on the 216 shapes of Kimia's 2D image database for normalized residual values equal to 20, 25 and 30. Noise level (parameter n) is expressed as a percentage of object area.

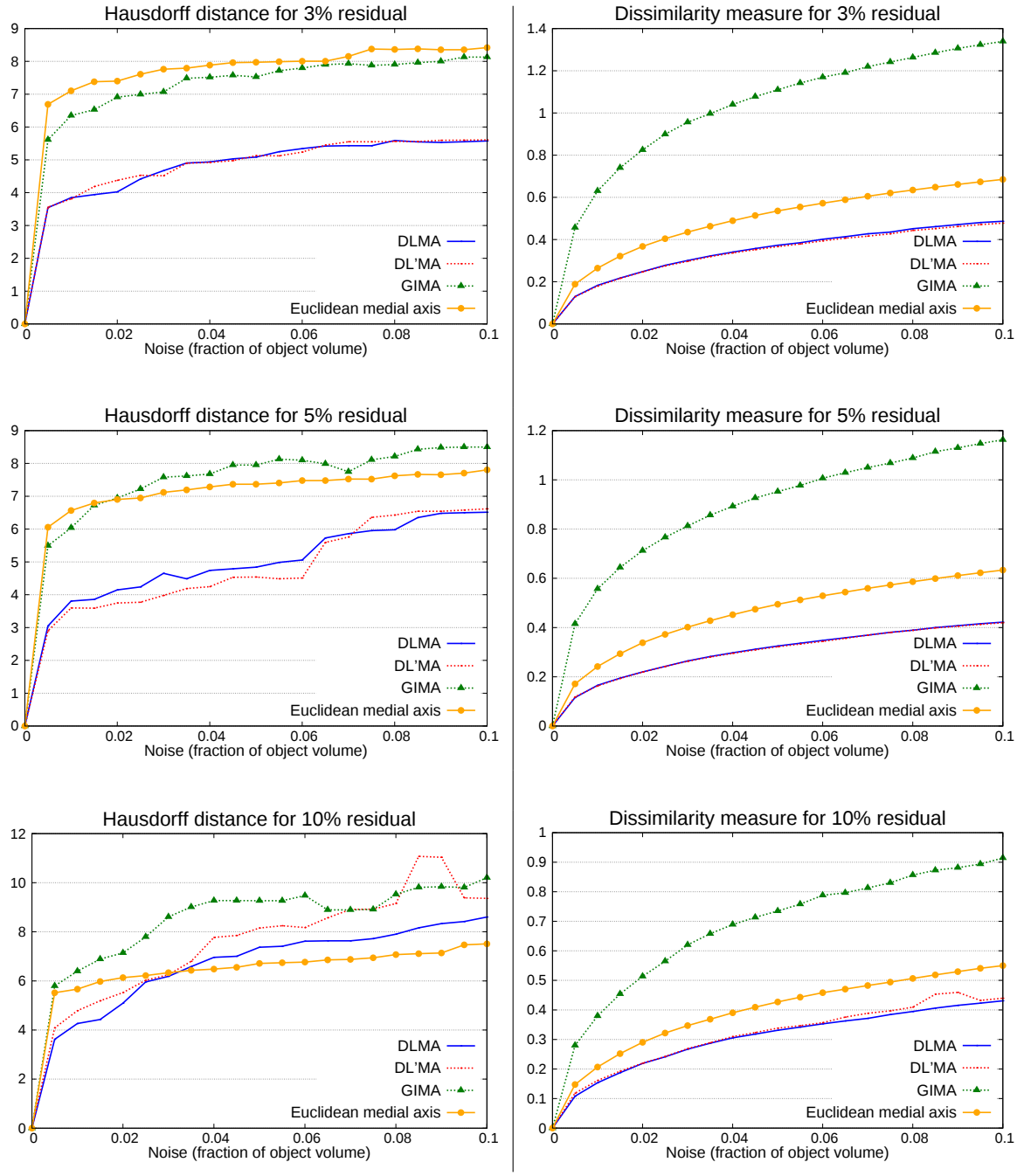


Figure 52: Hausdorff distance (**on the left**) and dissimilarity measure (**on the right**) between $M(X)$ and $M(E(X,n))$ on the 20 shapes of our 3D image database for normalised residual values equal to 3, 5 and 10. Noise level (parameter n) is expressed as a percentage of object area. When the DLMA curve is not visible, it is superimposed with DL'MA curve.

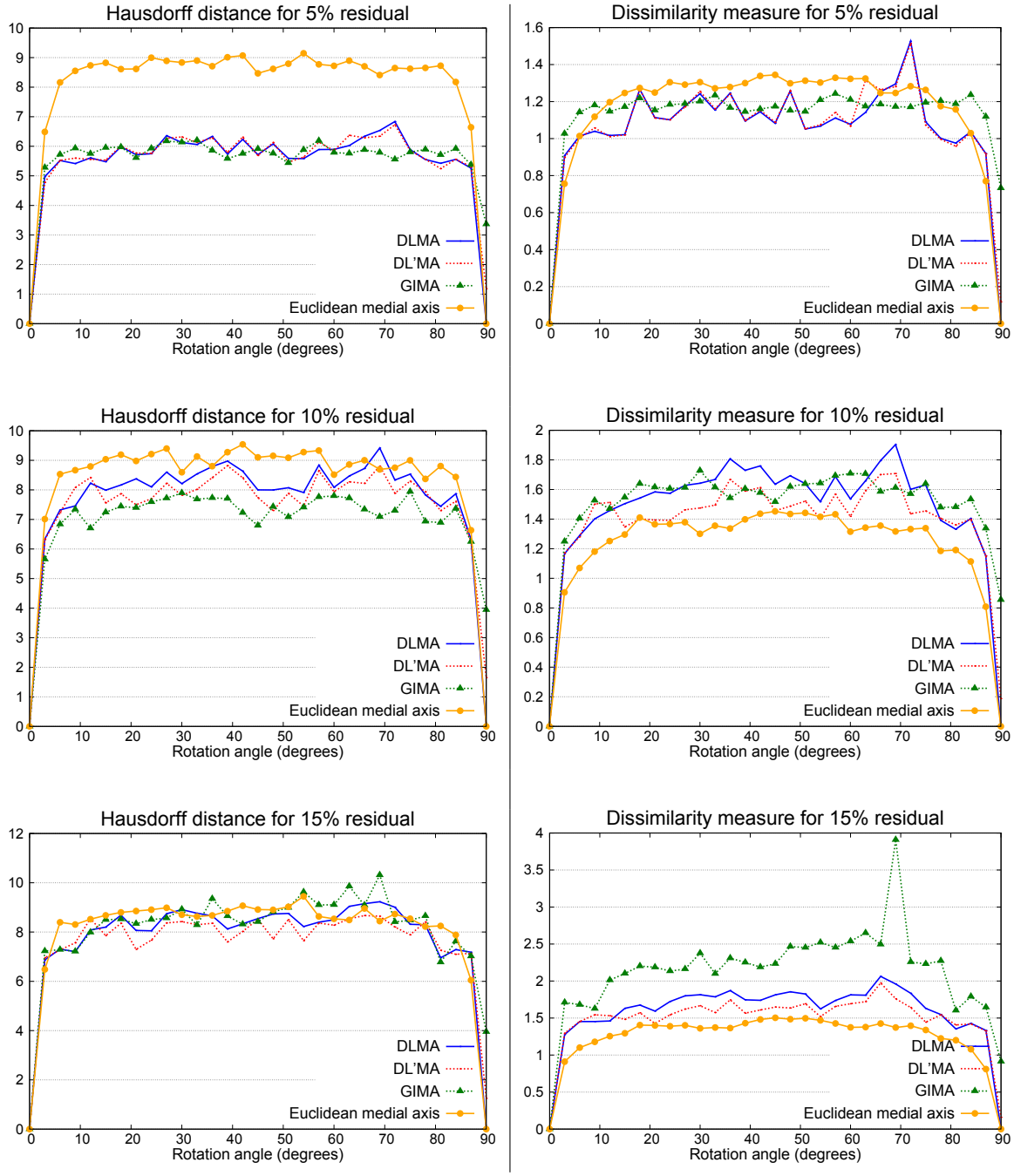


Figure 53: Hausdorff distance (**on the left**) and dissimilarity measure (**on the right**) between $M(R_\theta(X))$ and $R_\theta(M(X))$ on the 216 shapes of Kimia's 2D image database for normalised residual values equal to 5, 10 and 15. Noise level (parameter n) is expressed as a percentage of object area.

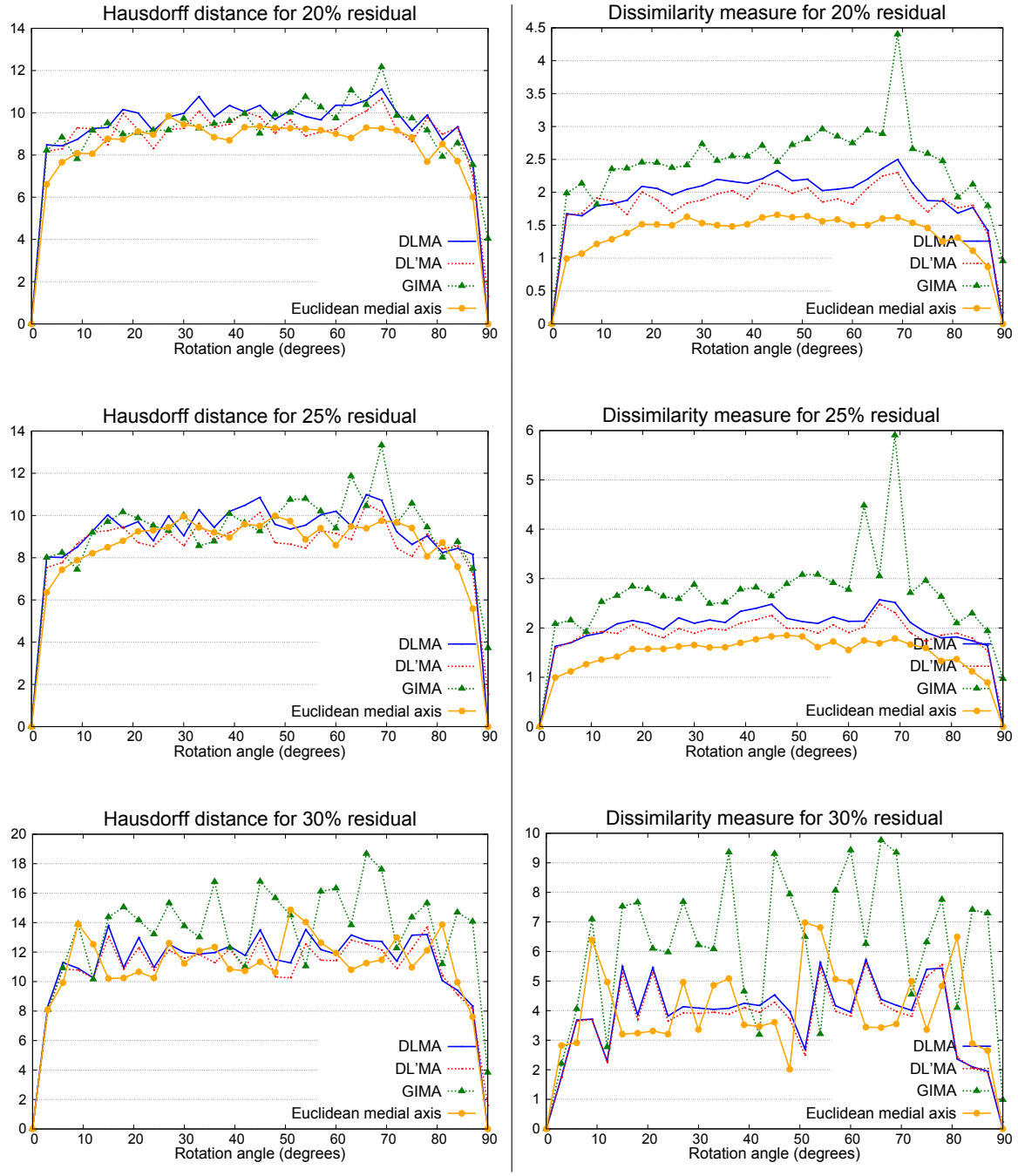


Figure 54: Hausdorff distance (**on the left**) and dissimilarity measure (**on the right**) between $M(R_\theta(X))$ and $R_\theta(M(X))$ on the 216 shapes of Kimia's 2D image database for normalised residual values equal to 20, 25 and 30. The rotation angle is represented on the horizontal axis.

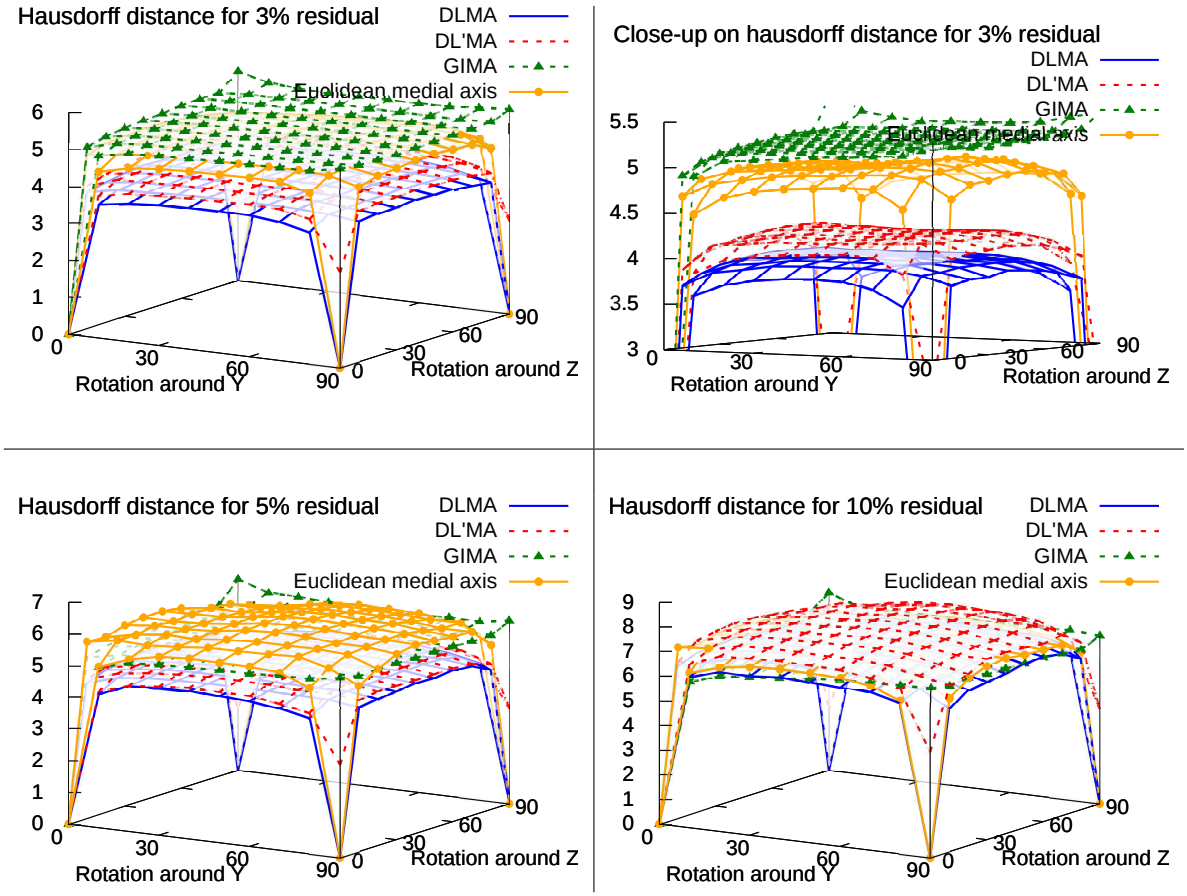


Figure 55: Hausdorff distance between $M(R_\theta(X))$ and $R_\theta(M(X))$ on the 20 shapes of our 3D image database, for normalised residual values equal to 3 (top images), 5 (bottom left image) and 10 (bottom right image) (the horizontal axes represent the rotation angle around the Y and Z axis). The top right image represent a close-up of the top left image.

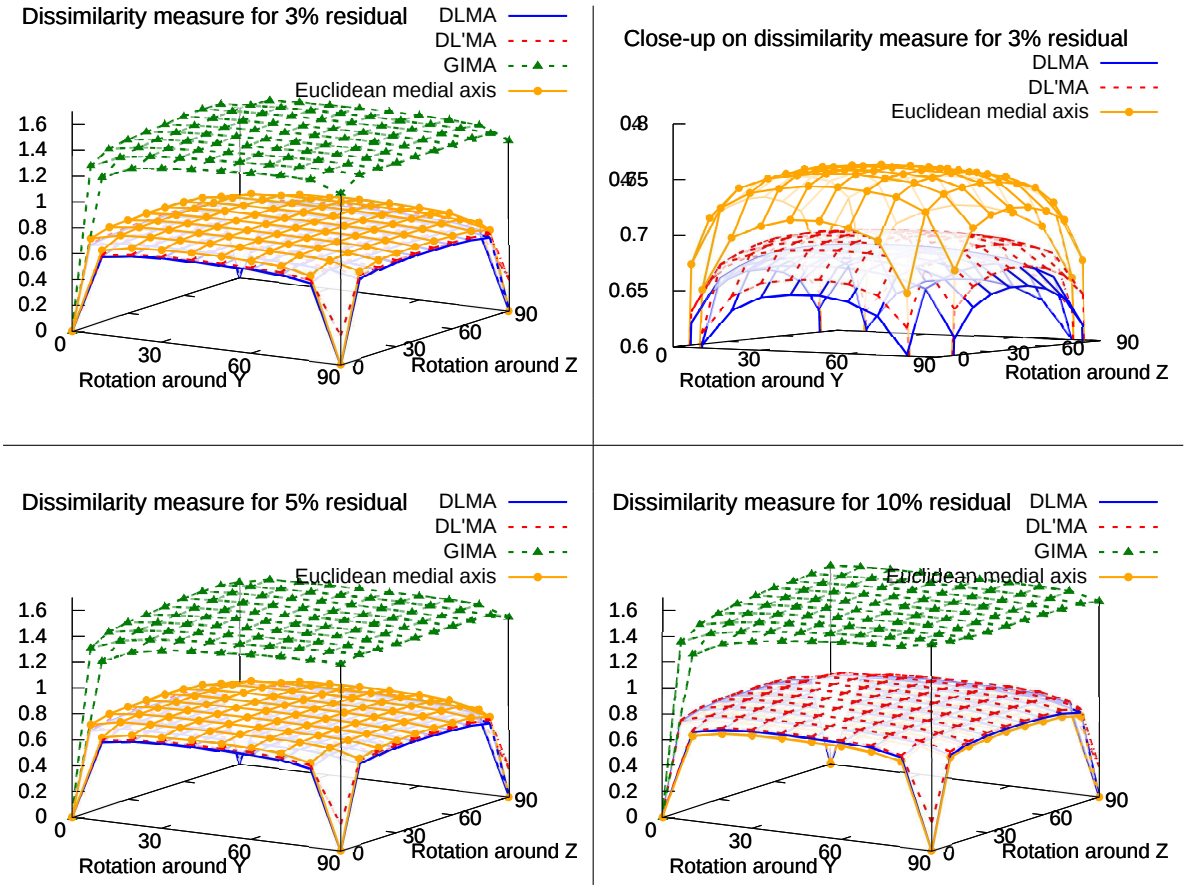


Figure 56: Dissimilarity measure between $M(R_\theta(X))$ and $R_\theta(M(X))$ on the 20 shapes of our 3D image database, for normalised residual values equal to 3 (top images), 5 (bottom left image) and 10 (bottom right image) (the horizontal axes represent the rotation angle around the Y and Z axis). The top right image represent a close-up of the top left image.

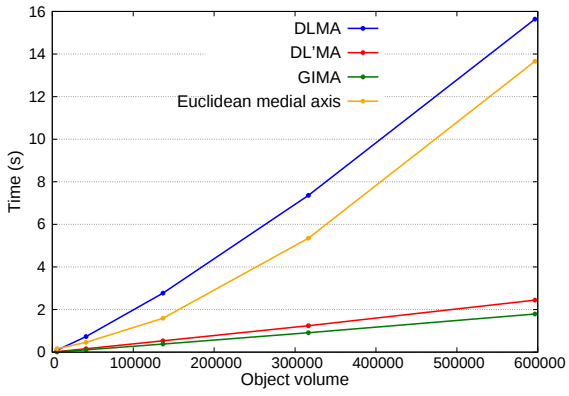
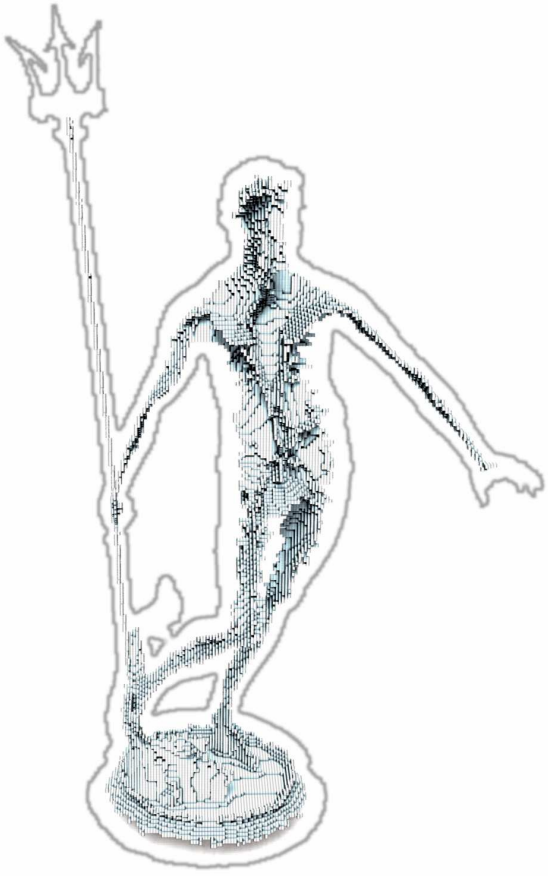
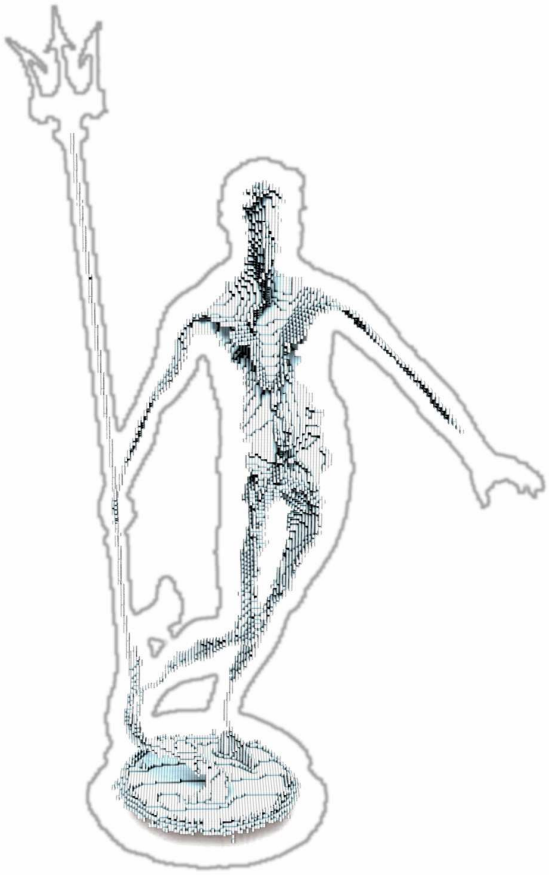


Figure 57: Computing times (in seconds, vert. axis) versus image sizes (horiz. axis).

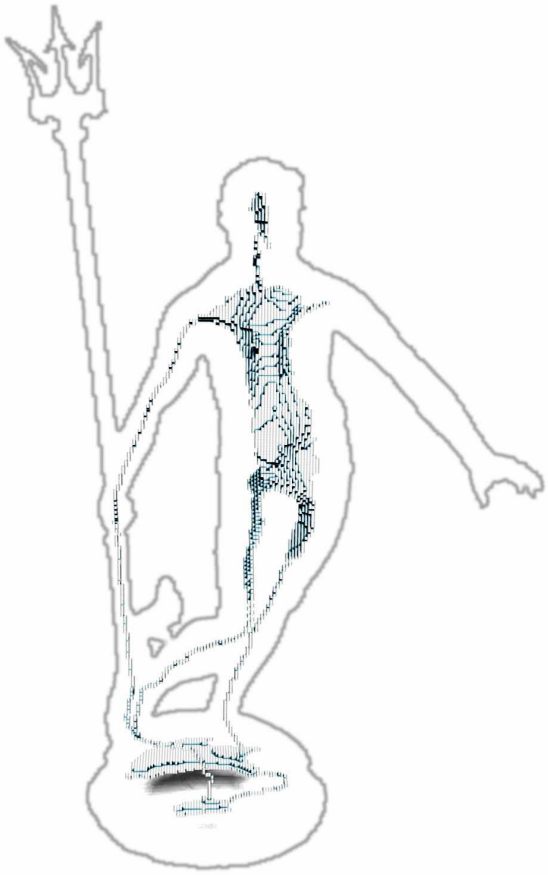
(a) Neptune



(b) $\lambda' = 8$



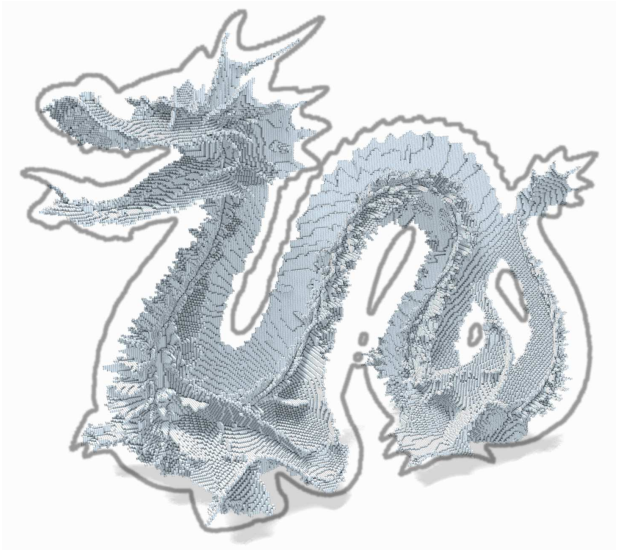
(c) $\lambda' = 10.2$



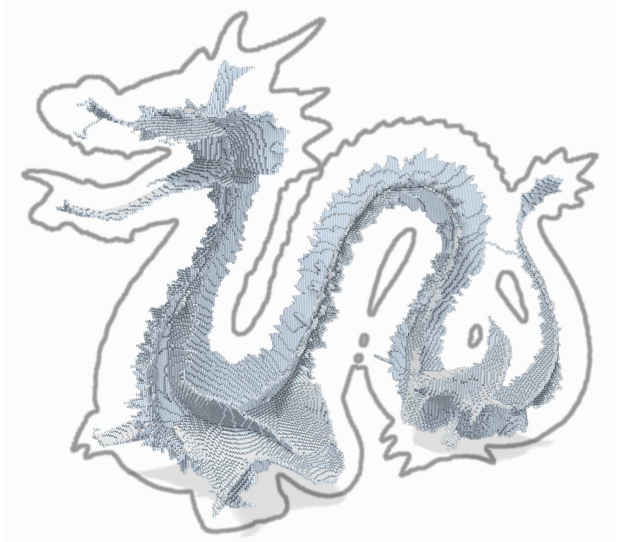
(d) $\lambda' = 17.5$

Figure 58: **Neptune shape and DL'MA -** The *Neptune* shape (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

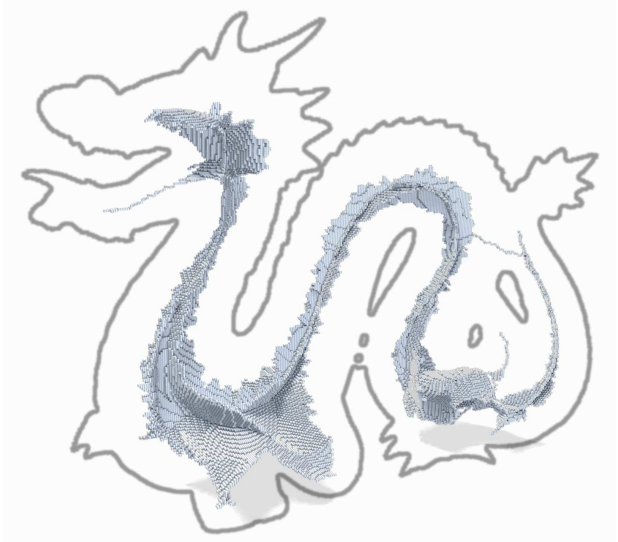
(a) Stanford dragon



(b) $\lambda' = 8$



(c) $\lambda' = 13.2$



(d) $\lambda' = 19$

Figure 59: **Stanford dragon shape and DL'MA -** The *Stanford dragon* shape [CL96] (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

(a) Chair 04

(b) $\lambda' = 7.5$

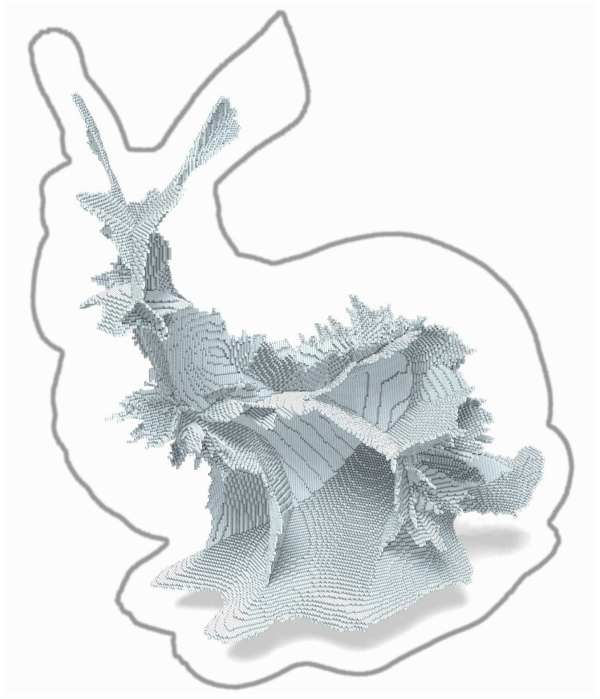
(c) $\lambda' = 12.8$

(d) $\lambda' = 18$

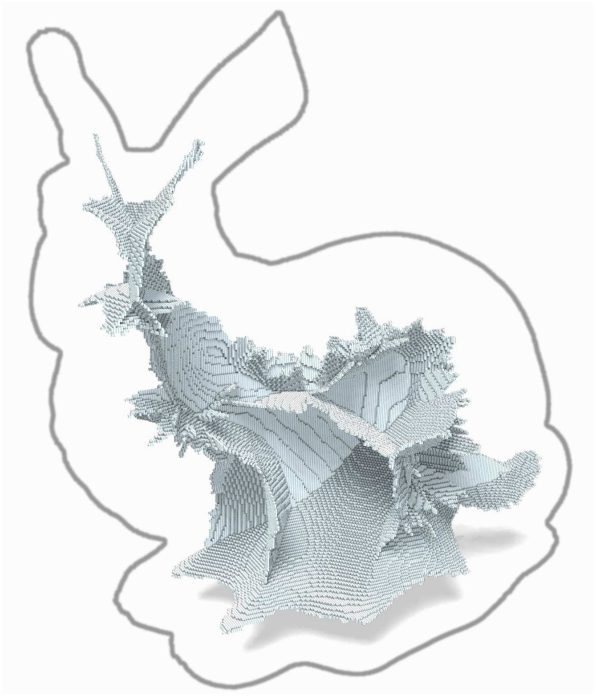
Figure 60: **A chair shape and DL'MA -** The *Chair 04* shape (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

(a) Stanford bunny

(b) $\lambda' = 12$



(c) $\lambda' = 19.7$



(d) $\lambda' = 24$

Figure 61: **Stanford bunny shape and DL'MA -** The *Stanford bunny* shape (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

(a) Pegasus

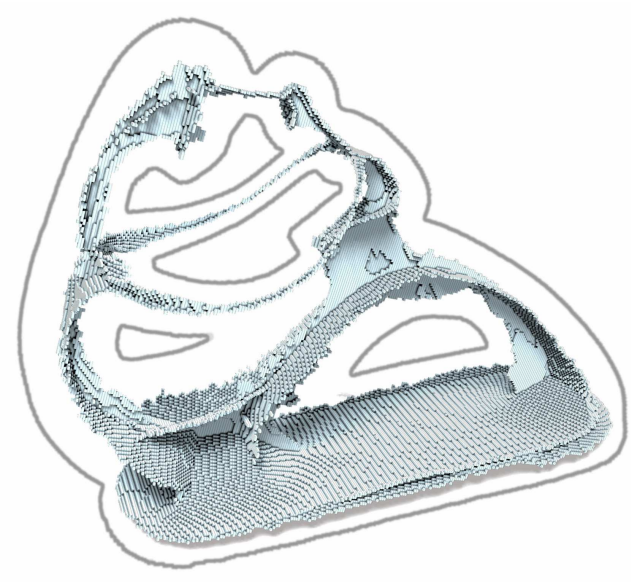
(b) $\lambda' = 6$

(c) $\lambda' = 9.3$

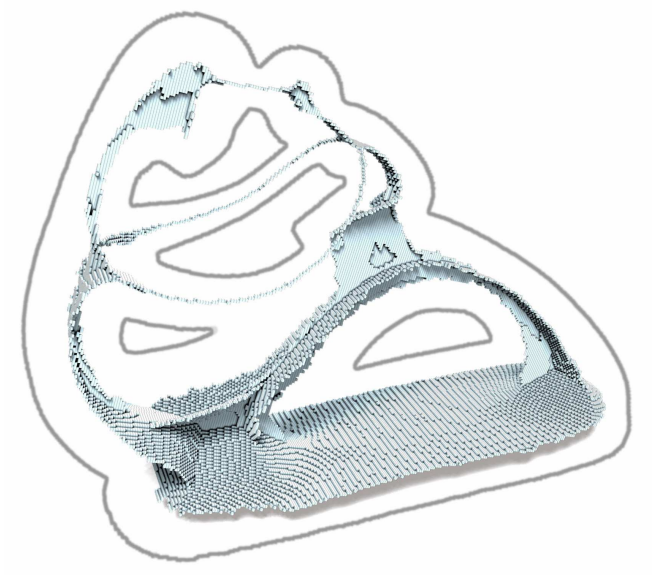
(d) $\lambda' = 14.5$

Figure 62: **Pegasus shape and DL'MA -** The *Pegasus* shape (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

(a) Fertility



(b) $\lambda' = 12$



(c) $\lambda' = 18.3$

(d) $\lambda' = 25$

Figure 63: **Fertility shape and DL'MA -** The *Fertility* shape (in **a**) and various homotopic thinning using a λ'-medial axis as constraint set (in **b**, **c** and **d**). The λ' parameter used for filtering the DL'MA map is specified under each image.

Part III

OBTAINING AND ANALYSING A SKELETON IN THE CUBICAL COMPLEX FRAMEWORK

5

INTRODUCTION TO THE CUBICAL COMPLEX FRAMEWORK

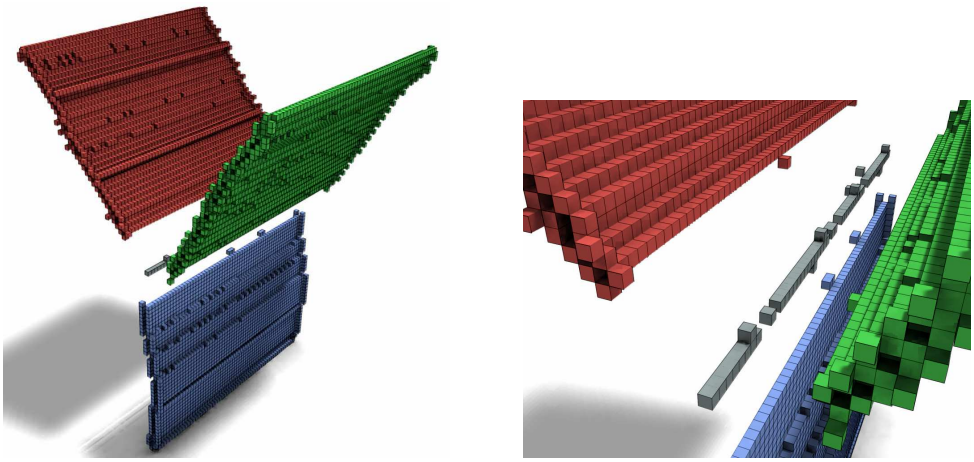
In this chapter, we introduce the cubical complex framework. In Sec. 5.1, we explain why the digital topology (DT) framework does not suit skeleton decomposition. In Sec. 5.2, we introduce the various concepts of the cubical complex framework that will be useful in the rest of this work.

Contents

5.1 UNSATISFYING RESULTS IN THE DT FRAMEWORK

In the previous chapter, we saw that various tools exist in the digital topology framework in order to provide skeletons with good reconstruction properties (containing important visual features from the original shape). Shape analysis based on the skeleton usually involves decomposing the skeleton into basic "parts" in order to better understand the structure of the initial object. Typically, in the continuous framework, a skeleton is a set of curves in 2d, and a set of curves and surfaces in 3d. Moreover, the skeleton is thin in any dimension, and the classical properties of the continuous framework can be applied: the intersection of two curves or more is a point, the intersection of two surfaces or more is a curve, ...

When working with voxels, the various parts of a skeleton are usually classified in two steps: first, each point is labelled according to the configuration of a more or less wide range of neighbour points. Then, the skeleton is divided into basic parts, each part being connected sets of points having "compatible" labels ([MBA93], [MFV98], [BPA01], [GK04], [Str05], [Kle06], [RT08], [JAB+10]). The results of the decomposition of a skeleton does not, in the digital topology framework, have the same properties than in the continuous framework: most thinning methods fail to always guarantee a thin result (for example, in 3d, some points of the skeleton can be classified as volume points), the intersection of two curves is not necessarily a single voxel, the intersection of two surfaces is not necessarily a set of voxels forming a curve (see, for example, Fig. 64). Thus, studying some specific parts of the skeleton, such as surfaces intersections, can be difficult in the DT framework. To our knowledge, there exists no decomposition nor thinning method in the DT framework which holds the properties listed in the first paragraph.



a) b)

Figure 64: **Decomposition of a skeleton made of voxels -** In **a)**, a skeleton made of voxels is decomposed into three basic parts, thanks to the method proposed in [MBA93]. In **b)**, a close-up to the center of the decomposition: voxels in grey were labelled as "surface intersection", however, put together, they do not form a simple curve.

The cubical complex framework is different from the DT framework, in the way that objects are made of basic bricks of various dimensions (not only voxels). As we show in Chap. 7, it is possible to obtain a sound decomposition of objects in the cubical complex

framework, with the guarantee that the intersection between two curves is a point, the intersection between two surfaces is a curve, ... We propose, in Chap. 6, various thinning algorithms, requiring no filtering parameter from the user, and allowing to obtain, in the cubical complex framework, a skeleton with good reconstruction properties. Moreover, we propose a methodology (based on thinning) in order to embed skeletons of the DT framework into the cubical complex framework, and obtain a thin skeleton. Finally, we give a method for embedding our skeletons in the DT framework, and we compare various thinning algorithms with our thinning method. We also prove that our skeleton, in the cubical complex framework, are thin (see Prop. 6.3.7 p. 145).

In the following chapter, we introduce the basis of cubical complex framework.

5.2 A FORMAL INTRODUCTION TO THE CUBICAL COMPLEX FRAMEWORK

In the following, we formally define some important concepts of the cubical complex framework.

5.2.1 BASIC DEFINITIONS

In the cubical complex framework, objects are made of various kind of basic bricks: a three-dimensional cubical complex is made of cubes, squares, lines and points. These basic bricks are called *faces*, defined hereafter. Let $\mathbb{Z}$ be the set of integers, we consider the family of sets $\mathbb{F}_0^1$ and $\mathbb{F}_1^1$, such that $\mathbb{F}_0^1 = \{\{a\} \mid a \in \mathbb{Z}\}$ and $\mathbb{F}_1^1 = \{\{a, a+1\} \mid a \in \mathbb{Z}\}$. Any subset f of $\mathbb{Z}^n$ such that f is the cartesian product of m elements of $\mathbb{F}_1^1$ and $(n-m)$ elements of $\mathbb{F}_0^1$ is called a face or an m-*face* of $\mathbb{Z}^n$, m is the dimension of f, we write $\dim(f) = m$. A 0-face is called a *vertex*, a 1-face is an *edge*, a 2-face is a *square*, and a 3-face is a *cube*.

a) Vertex b) Edge c) Square d) Cube e) A cell

Figure 65: **Example of faces in the cubical complex framework**, from **a** to **d**. In **e**, example of a cell (here, a bidimensional cell).

We denote by $\mathbb{F}^n$ the set composed of all faces in $\mathbb{Z}^n$. Given $m \in \{0, \ldots, n\}$, we denote by $\mathbb{F}_m^n$ the set composed of all m-faces in $\mathbb{Z}^n$.

Let $f \in \mathbb{F}^n$. We set $\hat{f} = \{g \in \mathbb{F}^n | g \subseteq f\}$, and $\hat{f}^* = \hat{f} \setminus \{f\}$. Any element of $\hat{f}$ is *a face of* f, and any element of $\hat{f}^*$ is *a proper face of* f. We call *star of* f the set $\check{f} = \{g \in \mathbb{F}^n | f \subseteq g\}$, and we write $\check{f}^* = \check{f} \setminus \{f\}$: any element of $\check{f}$ is a *coface of* f. It is plain that $g \in \hat{f}$ iff $f \in \check{g}$.

A set X of faces in $\mathbb{F}^n$ is a *cell*, or m-*cell*, if there exists an m-face $f \in X$ such that $X = \hat{f}$. The *closure* of a set of faces X is the set $X^- = \cup\{\hat{f} | f \in X\}$. The set $\overline{X}$ is $\mathbb{F}^n \setminus X$.

Definition 5.2.1 *A finite set* X *of faces in* $\mathbb{F}^n$ *is a* cubical complex *if* $X = X^-$*, and we write* $X \preceq \mathbb{F}^n$.

Any subset Y *of* X *which is also a complex is a* subcomplex of X*, and we write* $Y \preceq X$.

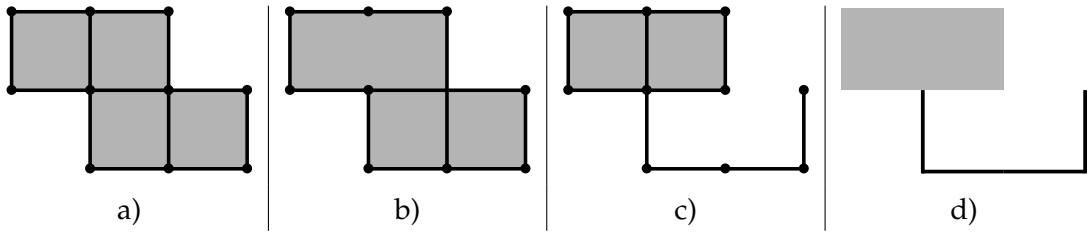


Figure 66: **Some examples of sets of faces -** Let A (resp. B, C, D) be the set of faces represented on **a** (resp. **b**, **c**, **d**). The set B is not a complex, as two squares miss an edge, and four edges miss a vertex. The set D is neither a complex, but A and C are complexes. Moreover, $A = B^{-}$. The complex A is pure as all its facets are squares, the complex C is not pure as some of its facets are squares while some others are edges. We have $D = C^{+}$. The complexes A and C have a dimension equal to 2.

Informally, in 2d, a set of face is a cubical complex if, for each square of the complex, the four edges (sides) of the complex also belong to the complex, and for each edge of the complex, the two vertices (extremities) of the edge belong to the complex (see Fig. 66). In 3d, a set of face is a cubical complex if, for each cube of the complex, the six squares (sides) of the cube are also in the complex, for each square of the complex, the four edges (sides) of the square are also in the complex, and for each edge of the complex, the two vertices (extremities) of the edge are also in the complex.

A face $f \in X$ is *a facet of* X if f is not a proper face of any face of X (see Fig. 67). We denote by X^{+} the set composed of all facets of X . A complex X is *pure* if all its facets have the same dimension (see Fig. 66). The *dimension of* X is $\dim(X) = \max\{\dim(f) \mid f \in X\}$. If $\dim(X) = d$, then we say that X is a d-complex. The notions of purity and dimensions can be trivially extended to sets of faces.

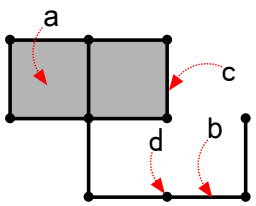


Figure 67: **Facets in complexes -** The square a is a facet, as well as the edge b. The edge c is not a facet, as it is included in a square. The vertex d is not a facet, as it is included in two edges.

5.2.2 THINNING: THE COLLAPSE OPERATION

The collapse operation is the basic operation for performing homotopic thinning of a complex. It consists of removing two distinct elements (f, g) from a complex X under the condition that g is contained in f and is not contained in any other element of X. This operation may be repeated several times. A more precise definition follows.

Definition 5.2.2 *Let* $X \preceq \mathbb{F}^n$*, and let* f, g *be two faces of* X*. The face* g *is* free for X*, and the pair* (f, g) *is* a free pair for X *if* f *is the only face of* X *such that* g *is a proper face of* f*.*

In other terms, (f, g) is a free pair for X whenever $\check{g}^* \cap X = \{f\}$ (g is included only in f). It can be easily seen that if (f, g) is a free pair for X and $\dim(f) = n$, then f is a facet and $\dim(g) = n - 1$.

Definition 5.2.3 *Let* $X \preceq \mathbb{F}^n$*, and let* (f, g) *be a free pair for* X*. The complex* $X \setminus \{f, g\}$ *is an* elementary collapse of X.

Let $Y \preceq \mathbb{F}^n$*, the complex* X collapses onto Y *if there exists a sequence of complexes* $(X_0, ..., X_\ell)$ *of* $\mathbb{F}^n$ *such that* $X = X_0$*,* $Y = X_\ell$ *and for all* $i \in \{1, \dots, \ell\}$*,* X_i *is an elementary collapse of* X_{i-1}*. We also say, in this case, that* Y is a collapse of X.

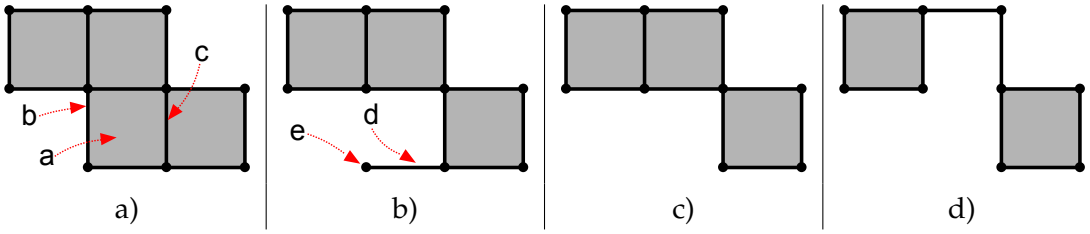


Figure 68: **Collapse of complexes -** Let A (resp. B, C, D) be the set of faces represented on **a** (resp. **b**, **c**, **d**). In the complex A, the edge b is only included in the square a: the pair (a, b) is free for A. The edge c is included in two squares of A, it is therefore not free for A. The complex $B = A \setminus \{a, b\}$ is an elementary collapse of A. In B, the vertex e is included in only one edge: the pair (d, e) is free for B, and $C = B \setminus \{d, e\}$ is an elementary collapse of B. The complex D is an elementary collapse of C, and therefore, D is a collapse of A.

Let f_0, f_ℓ be two n-faces of $\mathbb{F}^n$ (with ℓ being even). An $(n-1)$*-path from* f_0 *to* f_ℓ is a sequence $\pi = (f_0, ..., f_\ell)$ of faces of $\mathbb{F}^n$ such that for all $i \in \{0, \dots, \ell\}$, either i is even and f_i is an n-face, or i is odd and f_i is an $(n-1)$-face with $\check{f}_i^* = \{f_{i-1}, f_{i+1}\}$ (such path always exists).

The following proposition will serve us to prove the thinness of our skeletons.

Proposition 5.2.4 *Let* $X \preceq \mathbb{F}^n$ *be an* n*-complex, with* $n > 0$*. Then* X *has at least one free* $(n-1)$*-face.*

Proof Since X is an n-complex (hence X is finite) there exists an n-face a in X and an n-face b in $\overline{X}$. Obviously, there exists an $(n-1)$-path from a to b. Let h be the first n-face of π that is not in X, let k be the last n-face of π before h (thus k is in X), and let $e = k \cap h$ be the $(n-1)$-face of π between k and h. Since k and h are the only two n-faces of $\mathbb{F}^n$ that contain e, we see that the pair (k, e) is free for X. □

In conclusion, in $\mathbb{F}^3$, as long as a complex still contains 3-faces, it has a free 2-face and more collapse operations can be performed. Therefore, it is possible to perform collapse on a complex until no more 3-faces (volumes) can be found. When an n-complex has no n-faces, then it is said to be *thin*.

As in the DT framework, it is sometimes necessary to perform collapse in the cubical complex framework while preserving some faces safe from deletion: these faces are the so-called inhibitor set. When using an inhibitor set during collapse, the guarantee of having a thin result does no more hold. If the inhibitor set is not thin, the result of the thinning constrained by the inhibitor set cannot be thin. However, as illustrated on

figure 69, the inhibitor set can be thin and the result of the constrained thinning can still not be thin.

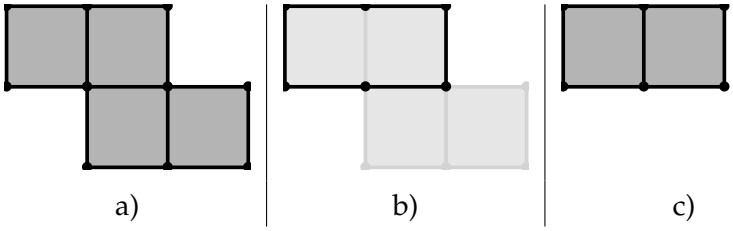


Figure 69: **The collapse of a complex may not be thin -** We consider the complex X in **a**, and the inhibitor set W highlighted in **b**. The set W is thin, but the result of the thinning of X constrained to keep W, presented in **c**, is not thin.

Under some conditions, it is possible to use an inhibitor set while collapsing, and still have the guarantee of getting a thin result. In order to prove this, let us state the following:

Lemma 5.2.5 *Let* $X \preceq \mathbb{F}^n$*, let* (f_1, g_1) *be a free pair of* X*, and* (f_2, g_2) *be a free pair of* $(X \backslash \{f_1, g_1\})$*. If* $\dim(f_2) > \dim(f_1)$*, then* (f_2, g_2) *is free for* X*.*

Proof If (f_2, g_2) is a free pair of $(X \backslash \{f_1, g_1\})$, then g_2 is included in only one face of $(X \backslash \{f_1, g_1\})$, which is f_2. As $(\dim(g_2) = \dim(f_2) - 1)$ and $(\dim(f_1) \leqslant (\dim(f_2) - 1))$, then $g_2 \nsubseteq f_1$ and $g_2 \nsubseteq g_1$. Therefore, g_2 is included in only one face of X, which is f_2: therefore, (f_2, g_2) is free for X.□

This lemma implies that, when one has a sequence of removal of free pairs of faces from a complex, one can only keep the free pairs of highest dimension and still have a sequence of removal of free pairs from the complex.

The following proposition states that, under some conditions on the inhibitor set (denoted W), it is possible to guarantee the thinness of the result of the constrained thinning.

Proposition 5.2.6 *Let* X *be an* n*-complex with* $n > 0$*, let* S *be a collapse of* X *such that* $\dim(S) \leqslant (n-1)$ *and let* $W \preceq S$*. Let* Y *be a collapse of* X *such that* $W \subseteq Y$ *and such that there are no free pairs in* Y *included in* $Y \backslash W$*. Then, the dimension of* Y *is inferior or equal to* $(n-1)$*.*

Proof In the following, we show that if there exists an n-face in Y, then it belongs to a free pair for Y that is included in $Y \backslash W$, a contradiction with the hypothesis of the proposition.

Let $C = ((a_1, b_1), ..., (a_k, b_k))$ be a sequence of removal of free faces which allows to obtain S from X: for all $i \in [1; k]$, (a_i, b_i) is free for $X \backslash \{a_1, b_1, ..., a_{i-1}, b_{i-1}\}$ and $S = X \backslash \{a_1, b_1, ..., a_k, b_k\}$. Let C' be the sequence C restrained only to free pairs containing an n-face: $C' = ((f_1, g_1), ..., (f_h, g_h))$. A consequence of lemma 5.2.5 is that, for all $i \in [1; h]$, (f_i, g_i) is free for $X \backslash \{f_1, g_1, ..., f_{i-1}, g_{i-1}\}$ and $(X \backslash \{f_1, g_1, ..., f_k, g_k\})$ is a collapse of X.

Any n-face $c \in Y$ is such that $c \in X$ and $c \notin S$, and therefore there exists $j \in [1; h]$ such that $c = f_j$. Without loss of generality, let j be the smallest integer such that $f_j \in Y$: for all $k \in [1; j-1]$, $f_k \notin Y$ and $g_k \notin Y$. As previously said, (f_j, g_j) is free for

$(X\backslash\{f_1, g_1, ..., f_{j-1}, g_{j-1}\})$: g_j is included in only one face of $(X\backslash\{f_1, g_1, ..., f_{j-1}, g_{j-1}\})$, and this face is f_j.

As $\{f_1, g_1, ..., f_{j-1}, g_{j-1}\} \cap Y = \emptyset$ and that $f_j \in Y$, then g_j is included in only one face of Y, and that face is f_j. Consequently, the pair (f_j, g_j) is free for Y. Moreover, the pair (f_j, g_j) belongs to the sequence C, therefore $f_j \notin S$ and $g_j \notin S$. As $W \subseteq S$, the pair (f_j, g_j) is included in $Y\backslash W$.□

5.2.3 FROM BINARY IMAGES TO CUBICAL COMPLEX

Traditionally, a binary image is defined as a finite subset of $\mathbb{Z}^n$ (with $n = 2$ or $n = 3$). Given $S \subseteq \mathbb{Z}^n$, the object voxels are the elements of S. This kind of image is the most common one in the field of image processing so, in order to work in cubical complex framework, we need to find a way to transpose a binary image to cubical complex framework.

Informally, to do so, we associate to each element of $S \subseteq \mathbb{Z}^n$ an n-face of $\mathbb{F}^n$ (to a pixel we associate a square, to a voxel we associate a cube). More precisely, let $x = (x_1, ..., x_n) \in S$, we define the n-face $\Phi(x) = \{x_1, x_1 + 1\} \times \ldots \times \{x_n, x_n + 1\}$. We can extend the map Φ to sets: $\Phi(S) = \{\Phi(x) | x \in S\}$. Given a binary image S, we associate to it the cubical complex $\Phi(S)^-$ (see Fig. 70).

In the following, most of the objects we consider were indeed binary images which were then transposed into cubical complex framework: this is why most of the two-dimensional complexes we show are pure 2-complexes, and most of the three-dimensional complexes we show are pure 3-complexes.

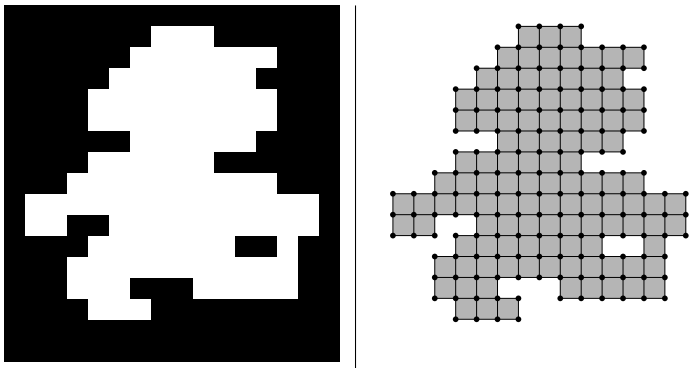


Figure 70: **From voxels to complex -** In **a**, a discrete object made of voxels. In **b**, its transposition to the cubical complex framework through the application Φ.

6

HOMOTOPIC THINNING IN THE CUBICAL COMPLEX FRAMEWORK

We propose in this chapter new parallel thinning algorithms in the cubical complex framework. One of the algorithm (see Sec. 6.2) allows to use digital topology framework tools (like the λ'-medial axis) in order to obtain a skeleton with good reconstruction properties (a skeleton containing visual information from the original object), in the cubical complex framework. In Sec. 6.3, we propose various thinning methods for obtaining skeletons with good reconstruction properties, and requiring no filtering parameter (no input from the user). We show that all these thinning methods produce a thin skeleton.

Finally, in Sec. 6.4, we propose an algorithm for embedding skeletons of the cubical complex framework inside the digital topology (DT) framework, and we compare a wide range of thinning algorithms with our own thinning methods.

The work exposed in this chapter was partially presented during the IWCIA 2009 conference, held in Cancun (Mexico). A conference article, partially covering the work presented in this chapter, was published in [CC09].

Contents

6.1 PARALLEL DIRECTIONAL THINNING BASED ON CUBICAL COMPLEX

Generally, the most "natural" method to thin an object consists of removing its border elements in parallel, in a symmetrical way. However, in general, removing at the same time two free pairs from a complex does not guarantee topology preservation (see Fig. 71a and b).

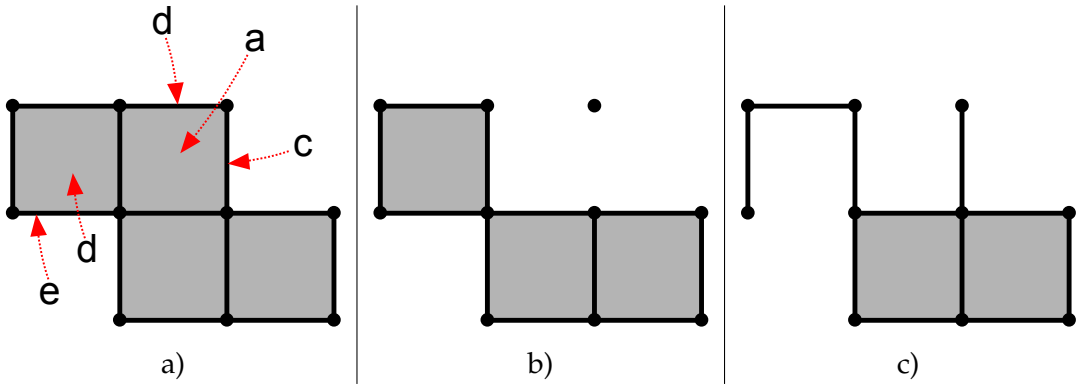


Figure 71: **Parallel removal of free pairs -** Let us consider, in the complex A depicted in **a)**, the two squares a and d and the three edges b, c and e. The pairs (a, b), (a, c) and (d, e) are free for A. Parallel removal of free pairs does not guarantee topology preservation: as shown on **b)**, $A\backslash\{a, b, c\}$ is not a collapse of A. However, as shown on **c)**, as $a \neq d$, $A\backslash\{a, b, d, e\}$ is a collapse of A.

In the 2d digital topology framework (pixels framework), A. Rosenfeld [Ros75] proposed a method for removing sets of simple points from an object without modifying its topology: this method consists of removing only simple points that have no neighbour in a given direction, and then change direction in order to scan all possible directions.

Knowing if this method can have an extension in 3D was pointed out by Kong, Litherland and Rosenfeld as an important problem in [KLR90] (question 547) (direct application of such method in 3D fails in preserving topology). In [Ber07], Bertrand developed a new framework, the critical kernel framework, relying on the cubical complexes, to give a method on how to remove, in the DT framework, multiple simple points at the same time (this method extends to all dimensions, although computation problems appear after the 4th dimension). Using this framework, question 547 of [KLR90] receives a first answer in [CB08].

In the following, we show that the directional strategy can also be extended to 3d cubical complexes, and we propose new thinning algorithms allowing to obtain thin skeletons. The thinness of the skeletons we obtain will allow to easily analyse and decompose them (see Sec. 7.1).

6.1.1 REMOVING FREE PAIRS IN PARALLEL

In the cubical complex framework, parallel removal of simple pairs can be easily achieved when following simple rules that we will give now. First, we need to define the *direction* and the *orientation* of a free face.

Let $f \in \mathbb{F}^n$, the *center of* f is the center of mass of the points in f, that is, $c_f = \frac{1}{|f|} \sum_{a \in f} a$. The center of f is an element of $[\frac{\mathbb{Z}}{2}]^n$, where $\frac{\mathbb{Z}}{2}$ denotes the set of half integers. Let

$X \preceq \mathbb{F}^n$, let (f, g) be a free pair for X, and let c_f and c_g be the respective centers of the faces f and g. We denote by $V(f, g)$ the vector $(c_f - c_g)$ of $[\frac{\mathbb{Z}}{2}]^n$.

We define a surjective function $Dir() : \mathbb{F}^n \times \mathbb{F}^n \to \{0, \dots, n-1\}$ such that, for all free pairs (f, g) and (i, j) for X, $Dir(f, g) = Dir(i, j)$ if and only if $V(f, g)$ and $V(i, j)$ are collinear (we don't bother defining $Dir()$ for non free pairs as it won't be useful in this case). The number $Dir(f, g)$ is called the *direction* of the free pair (f, g). Let (f, g) be a free pair, the vector $V(f, g)$ has only one non-null coordinate: the pair (f, g) has a *positive orientation*, and we write $Orient(f, g) = 1$, if the non-null coordinate of $V(f, g)$ is positive; otherwise (f, g) has a *negative orientation*, and we write $Orient(f, g) = 0$. On Fig. 71, the free pair (a, c) and the free pair (d, e) have different directions; the free pairs (a, b) and (d, e) have the same direction, but opposite orientations.

Now, we give a property of collapse which brings a necessary and sufficient condition for removing two free pairs of faces in parallel from a complex, while preserving topology (see Fig. 71c).

Proposition 6.1.1 *Let $X \preceq \mathbb{F}^n$, and let (f, g) and (k, ℓ) be two distinct free pairs for X. The complex X collapses onto $X \setminus \{f, g, k, \ell\}$ if and only if $f \neq k$.*

Proof If $f = k$, then it is plain that (k, ℓ) is not a free pair for $Y = X \setminus \{f, g\}$ as $k = f \notin Y$. Also, (f, g) is not free for $X \setminus \{k, \ell\}$. If $f \neq k$, then we have $g \neq \ell$, $\check{g}^* \cap X = \{f\}$ (g is free for X) and $\check{\ell}^* \cap X = \{k\}$ (ℓ is free for X). Thus, we have $\check{\ell}^* \cap Y = \{k\}$ as $\ell \neq g$ and $k \neq f$. Therefore, (k, ℓ) is a free pair for Y. □

From Prop. 6.1.1, the following corollary is immediate.

Corollary 6.1.2 *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X such that, for all $a, b \in \{1, \dots, m\}$ (with $a \neq b$), $f_a \neq f_b$. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

Considering two distinct free pairs (f, g) and (i, j) for $X \preceq \mathbb{F}^n$ such that $Dir(f, g) = Dir(i, j)$ and $Orient(f, g) = Orient(i, j)$, we have $f \neq i$. From this observation and Cor. 6.1.2, we deduce the following property.

Corollary 6.1.3 *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X having all the same direction and the same orientation. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

6.1.2 A DIRECTIONAL PARALLEL THINNING ALGORITHM

We say that a d-face of X is a *border face* if it contains a free $(d-1)$-face. Define $CBorder(X)$ as the set of all border faces of X. We are now ready to introduce a directional parallel thinning algorithm (Alg. 27).

Intuitively, we want the algorithm to remove free faces "layer by layer": we don't want, after a single execution ($\ell = 1$), to have unequal thinning of the input complex. Therefore, we want each execution of the algorithm to remove free faces located on the border of the input complex: this is why we introduce, on line 4 the set L, and that we remove only faces located in L on line 17. The sets E (line 12) and G (line 15) allows to remove whole sets of free faces in parallel from X, thanks to the direction and

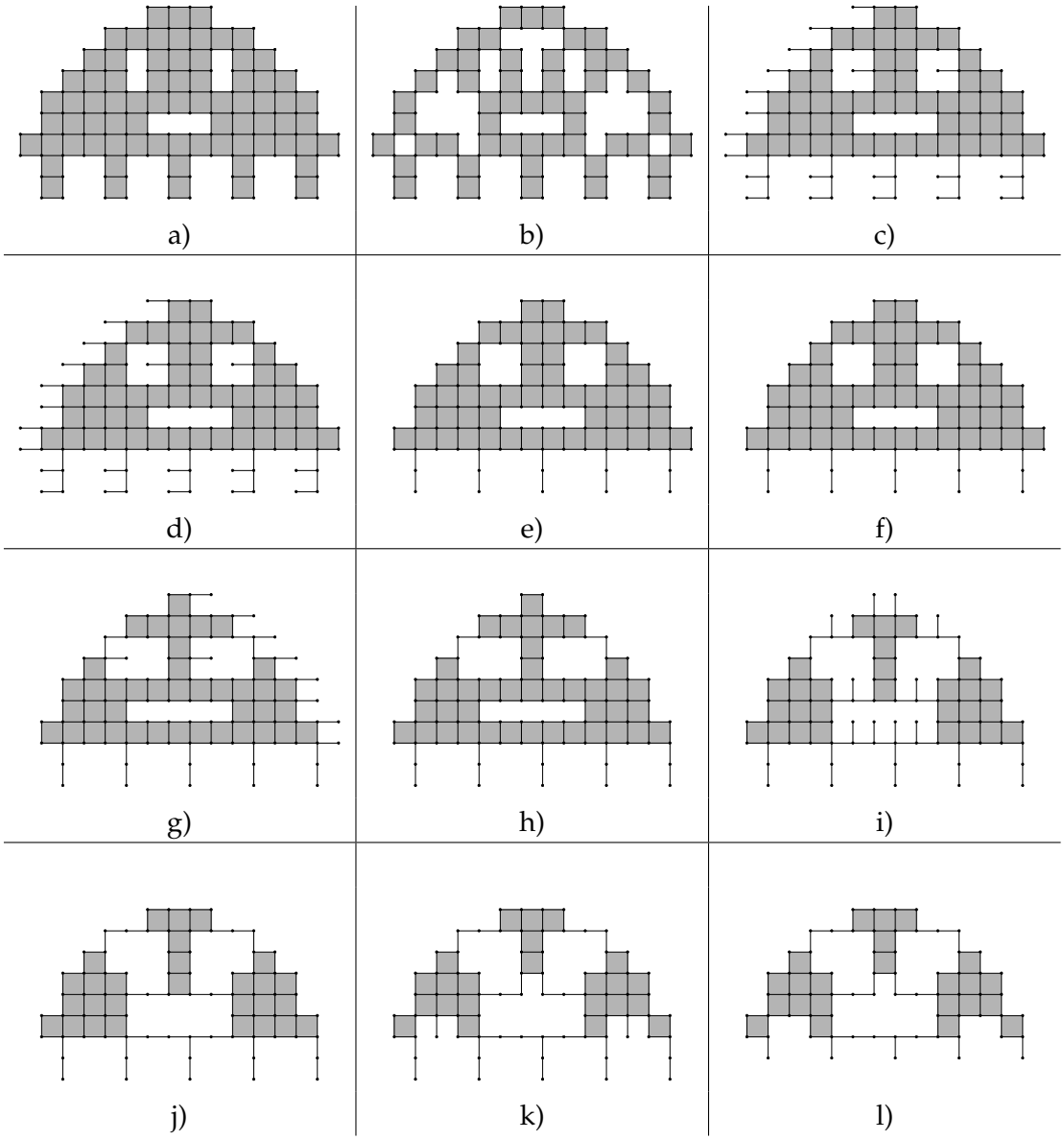


Figure 72: **One iteration of the ParDirCollapse algorithm -** On **a**, the original complex, and on **b**, the border of the complex (the clear faces do not belong to the border): no face outside the border will be removed during the iteration of the ParDirCollapse algorithm. The face will be removed in the following direction/orientation: rightward, leftward, downward, upward. On **c**, all free pairs consisting of a square and an edge, and having a rightward direction/orientation, are represented with a clear colour, and removed on **d**. On **e**, all free pairs consisting of an edge and a vertex, and having a rightward direction/orientation, are represented with a clear colour, and removed on **f**. The rest of the steps of the algorithm are represented from **g** to **l**.

Algorithm 27: $\mathtt{ParDirCollapse}(X, W, \ell)$

Data: A cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed, and $\ell \in \mathbb{N}$, the number of layers of free faces which should be removed from X

Result: A cubical complex

```
while there exists free faces in X \ W and ℓ > 0 do
    L = CBorder(X)⁻;
    for t = 0 → n − 1 do
        for s = 0 → 1 do
            for d = n → 1 do
                E = {(f, g) free for X | g ∉ W,
                     Dir(f, g) = t, Orient(f, g) = s, dim(f) = d};
                G = {(f, g) ∈ E | f ∈ L and g ∈ L};
                X = X \ G;
            end
        end
    end
    l = l − 1;
end
return X;
```

orientation of faces previously defined. A detailed view of each step of the algorithm is shown on Fig. 72.

Different definitions of the orientation and direction can be given, corresponding to different order of removal of free faces in the complex. These changes lead to different results, but arbitrary choices on the order of removal of free pairs must be made in order to obtain, at the end, a thin skeleton (no more n-faces when working in the n-dimension). Once orientation and direction have been defined, the results of the algorithm are uniquely defined.

Algorithm 27 may be easily implemented to run in linear time complexity (proportionally to the number of faces of the complex). Indeed, checking if a face is free or not may be easily done in constant time. Moreover, when a free pair (f, g) is removed from the input complex, it is sufficient to scan the faces of f and the cofaces of g in order to find new free faces, as other faces' status won't change (the implemented algorithm contains these optimizations).

As shown in Fig.73, when the input complex has a nearly constant thickness, it is possible to obtain a surface skeleton of X by choosing a convenient value of ℓ as the last parameter of Alg. 27. However, in most cases, it is not possible to find a value of ℓ which is satisfying for the whole complex. In the next section, we explain how to use the directional thinning strategy to obtain a surface skeleton from a complex without having to tune any "thickness" parameter.

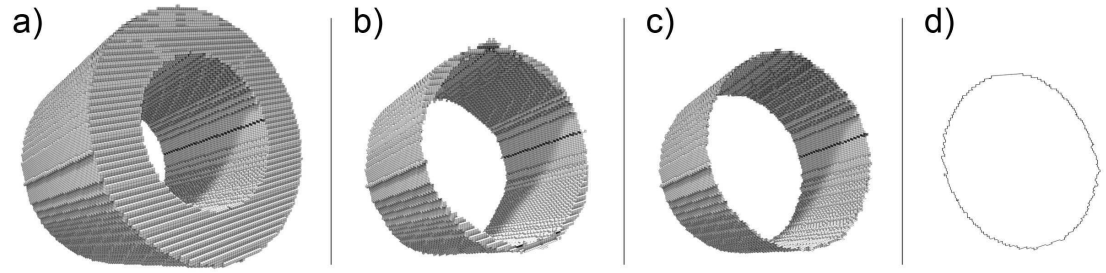


Figure 73: **Example of thinning with ParDirCollapse algorithm - (a)**: A 3-complex X having the shape of a thick tube with a skewed axis. **(b)**: The result of $\mathtt{ParDirCollapse}(X, \emptyset, 10)$ is not thin, some 3-faces remain in the object. **(c)**: The result of $\mathtt{ParDirCollapse}(X, \emptyset, 15)$ is a 2-complex. **(d)**: The result of $\mathtt{ParDirCollapse}(X, \emptyset, \infty)$ does not contain any cube nor any square, it is composed of edges and points.

6.2 ASPECT PRESERVATION DURING THINNING: A METHOD BASED ON MEDIAL AXES

As said previouly, alg. 27 can perform satisfactory thinning in some cases. However, setting the wrong value as third parameter of the algorithm may result either in loss of information, or in a complex which still contains volumic parts. If that parameter is set to infinity, the algorithm performs an ultimate thinning of the input: free faces are removed until no more can be found. If one wants to obtain a skeleton which keeps some "visual features" from the input, other strategies must be used. In order to obtain a satisfactory skeleton, which contains interesting visual information from the original object without containing "spurious" parts, it is usually necessary to use a filtering step, during or after the thinning, in order to remove unwanted "extra parts" from the skeleton.

This step is usually difficult, as the skeleton is very sensitive to small contour perturbation. A recent survey [ABE09] summarizes recent studies dedicated to this matter in the digital topology (DT) framework. In the following, we propose various methods for obtaining a filtered skeleton in the cubical complex.

6.2.1 CRITERION FOR DYNAMIC ANCHOR DETECTION

We will, in the following, propose two strategies for obtaining a skeleton which retains visual information from the original object, each involving the use of Alg. 27. These two methods have drawbacks, but will allow to point out the difficulty of the skeleton filtering task.

Both algorithms proposed hereafter are based on the same principle: a single layer of faces is removed using Alg. 27, then all facets of a given dimension are detected and kept safe from any future removal. This process is repeated until stability.

On Fig. 74, we present various steps of Alg. 28 running on Fig. 72a. Figure 76a shows the result of $\mathtt{CollapseFacet}(X, \emptyset)$, where X is the tube shown in Fig. 73a. The resulting complex is a 2-complex containing some "branches" and "surface patches" (as shown on the detailed view) which do not represent significant "surfacic features" of the original

Algorithm 28: CollapseFacet(X, W)

Data: An n-dimensional cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed
Result: A cubical complex

```
while there exists free faces for X in X\W do
    X = ParDirCollapse(X, W, 1);
    W = W ∪ {f̂ ∈ X | f ∈ X⁺ and dim(f) ⩽ (n − 1)};
end
return X;
```

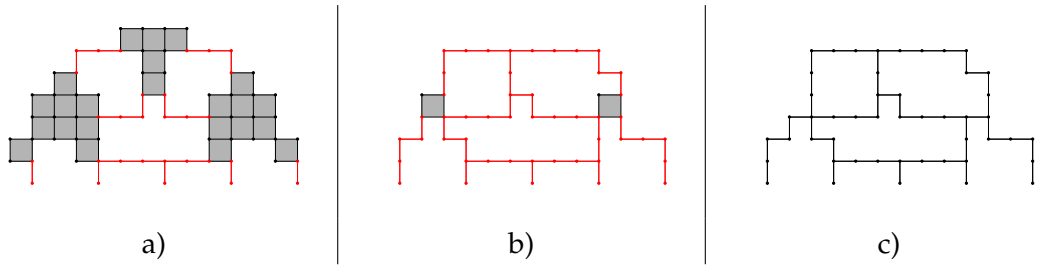


Figure 74: **The CollapseFacet algorithm - a)** After one iteration of Alg. 28 applied on shape of Fig. 72a, all the edges which are facets (not contained in any square) are marked in red and won't be deleted by the thinning. **b)** After a second iteration, new facets appear and are marked in red. **c)** The result of the thinning, after a third iteration.

object. This motivates the introduction of a variant of this method, which imposes a more restrictive condition to preserve 2-facets from removal.

We say that a facet $f \in X^+$ is an *isthmus of* X if it contains no free face, in other words, if each face in f is included in a facet of X distinct from f.

In Alg. 29, the facets that are detected and kept safe from further removal are the isthmuses of X.

Algorithm 29: CollapseIsthmus(X, W)

Data: An n-dimensional cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed
Result: A cubical complex

```
while there exists free faces for X in X\W do
    X = ParDirCollapse(X, W, 1);
    W = W ∪ {f̂ ∈ X | f is an isthmus of X and dim(f) ⩽ (n − 1)};
end
return X;
```

On Fig. 75, we present various steps of Alg. 29 running on Fig. 72a. Fig. 76b shows the result of CollapseIsthmus$(X, \emptyset)$, where X is the tube shown in Fig. 73a. The resulting complex contains less branches and spurious surfaces (as shown on the detailed view) than the complex obtained with CollapseSurface$(X, \emptyset)$.

The results obtained show that both algorithms presented above allow one to obtain a 2-dimensional skeleton from a 3-complex, or a 1-dimensional skeleton from a 2-

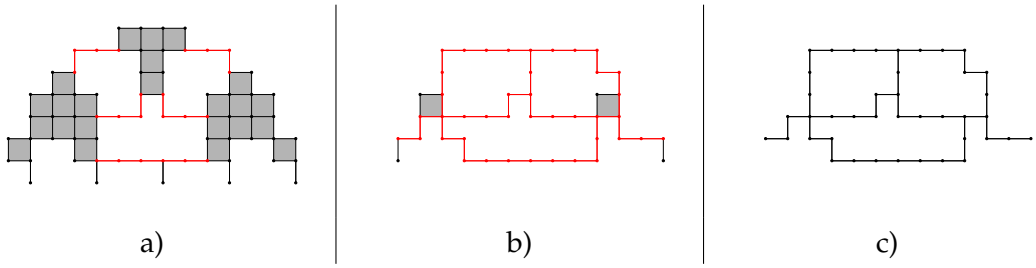


Figure 75: **The CollapseIsthmus algorithm - a)** After one iteration of Alg. 29 applied on shape of Fig. 72a, all the edges which are isthmus (not contained in any square and do not contain any free vertex) are marked in red and won't be deleted by the thinning. **b)** After a second iteration, new isthmus appear and are marked in red. **c)** The result of the thinning, after a third iteration.

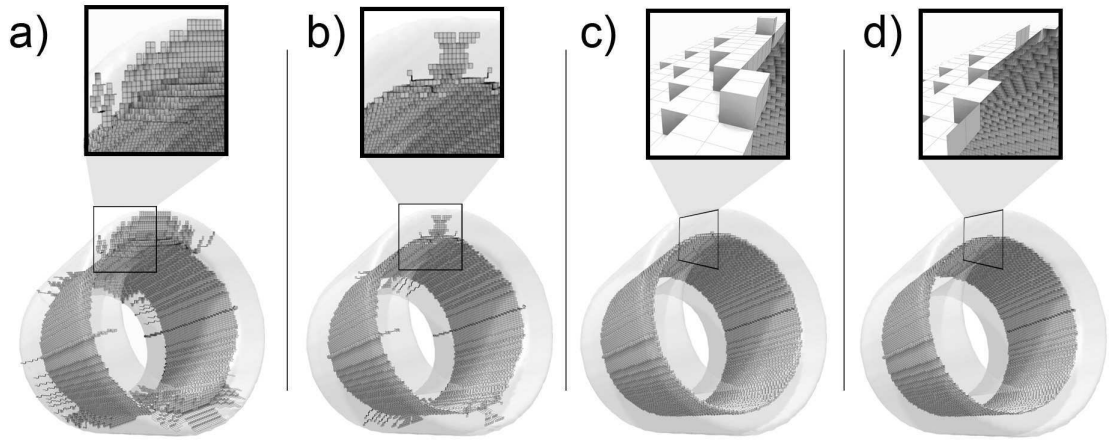


Figure 76: **Results of different thinning algorithms** performed on the complex X presented in Fig 73a. (a) : $\mathtt{CollapseFacet}(X, \emptyset)$. (b) : $\mathtt{CollapseIsthmus}(X, \emptyset)$. (c) : The discrete λ-medial axis (Sec. 4.1) of X, with $\lambda = 18$. (d) : Result of the method described in Sec. 6.2.2 performed on X, with $\lambda = 18$.

complex, containing important "shape information" from the initial object. However, even if Alg. 29 produces better results than Alg. 28, it fails in obtaining a completely satisfactory skeleton. In the next section, we will see how to perform better filtering with two different methods.

Proposition 6.2.1 *Let X be an n-complex, the results of* $\mathtt{CollapseIsthmus}(X, \emptyset)$ *and* $\mathtt{CollapseFacet}(X, \emptyset)$ *do not contain any n-face (the results are thin).*

Following Prop. 5.2.4, proposition 6.2.1 is straightforward: any n-face would become free after some iteration of the loop of the algorithms, and the only elements added to the inhibitor set are faces not included in any n-face. The results of Alg. 28 and Alg. 29 are thin in 3d (they do not contain any 3-face), provided that the input parameter W is set to $\emptyset$. However, thanks to parameter W, it is possible to constrain these algorithms to preserve selected parts of the original object, in addition to the surface parts that are automatically detected.

6.2.2 USING MEDIAL AXES: AN APPLICATION WITH DL'MA

In the following, we will see how filtering tools from the DT framework can be directly used in the cubical complex framework in order to obtain a filtered skeleton.

Let S denote our original binary set, a finite subset of $\mathbb{Z}^n$, and let T be a subset of S. Typically, the set T would have been computed from S, and should contain interesting visual information of S (for example, T could be the discrete λ'-medial axis of S thresholded at a given value). In this part, the function Φ, introduced in Sec.5.2.3, which allows to embed a subset of $\mathbb{Z}^n$ into $\mathbb{F}^n$.

Here is a description of the methodology to follow in order to obtain, from a binary set $S \subset \mathbb{Z}^n$, a "thin" skeleton in the cubical complex framework. First, a subset T of S representing the important visual features of S which should be kept, must be computed. For example, T could be the discrete λ'-medial axis of S for a chosen value of λ. From here, we will consider the complexes $X = \Phi(S)$ and $W = \Phi(DL'MA(S, \lambda))$.

The step consists of computing, using Alg. 27, the complex $Y = ParDirCollapse(X, W, +\infty)$ that contains W and that is topologically equivalent to X.

Now, if X is an n-complex, then W is also an n-complex (see, in 3d, the close-up of Fig. 76c). Therefore, Y is also an n-complex; in order to obtain a "thin" result (an $(n-1)$-complex), we set $Z = CollapseIsthmus(Y, \emptyset)$ which is both thin (following Prop. 5.2.4 there is no n-face in this complex) and topologically equivalent to X. See the close-up of Fig. 76d for an illustration.

The centering of Z in X is achieved thanks to the use of the DL'MA, based on the Euclidean distance. The parameter λ can be tuned in order to adjust the filtering of the characteristics (size, smoothness of contours ...) of the input shape, and to the requirements of the user.

As previously said, the user may use any subset of S rather than the DL'MA in order to filter the interesting visual features in the first step. Algorithm 30 sumarizes the previous description.

Algorithm 30: $CollapseVox(S, T)$

Data: A finite set $S \subset \mathbb{Z}^n$, and a set $T \subseteq S$ representing important visual features of S

Result: A cubical complex of dimension $(n-1)$

```
1 X = ParDirCollapse(Φ(S), Φ(W), +∞);
2 X = CollapseIsthmus(X, ∅);
3 return X;
```

Figures 76d and 77 show various results using algorithm 30 and the DL'MA. It can be seen that the resulting 2-complexes indeed capture the main surfacic features of the original objects, without spurious branches or surface patches. A significant advantage of the 2D nature of the obtained skeletons, is to enable an easy analysis of important shape features such as intersections of surface parts.

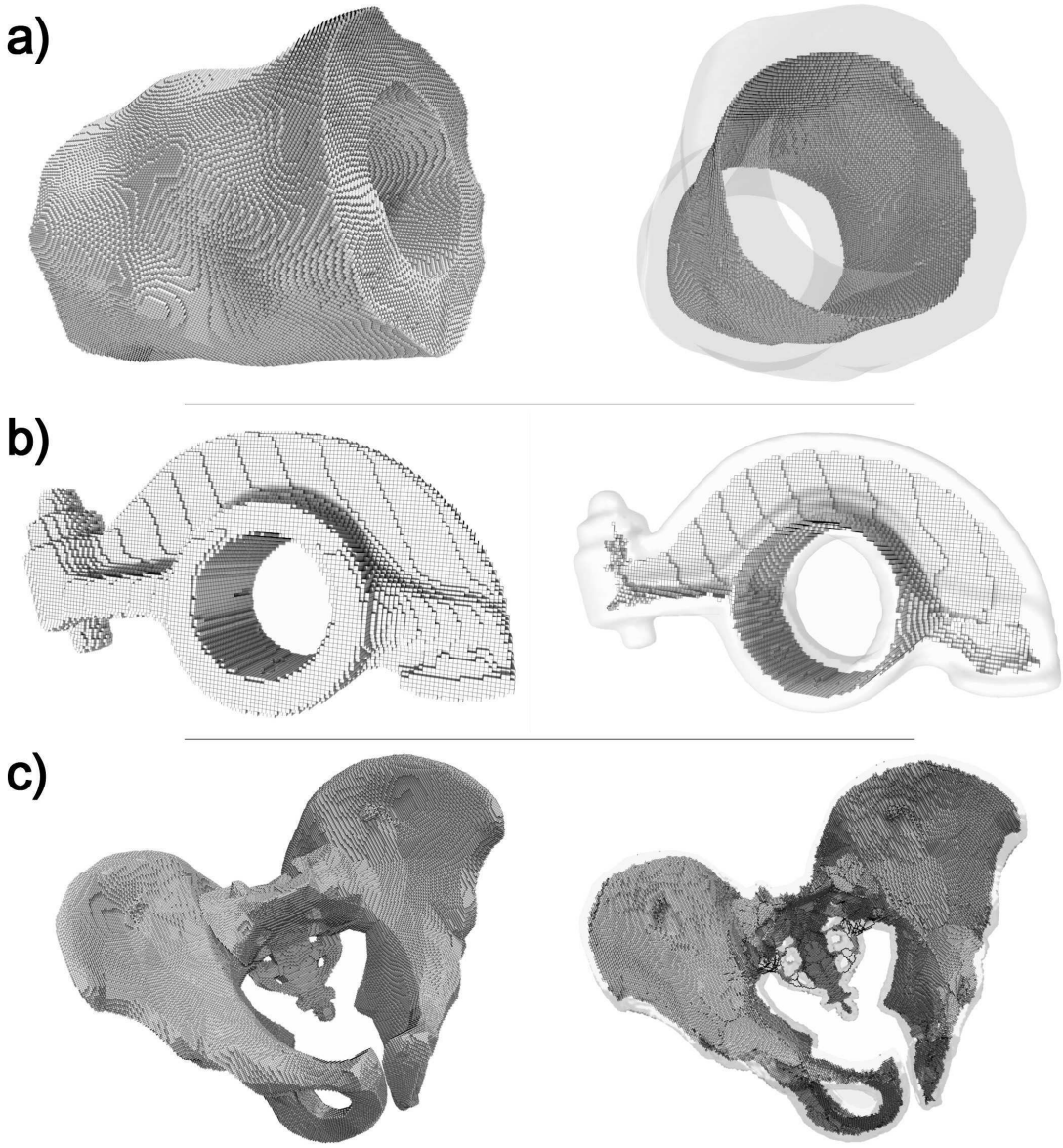


Figure 77: **Results of CollapseVox with DL'MA -** Results of the methodology described in Sec. 6.2.2, using the DL'MA. On the left, the original object; on the right, the result obtained by our method. (a): A bumped and skewed tube, $\lambda = 20$. (b): A rocker arm, $\lambda = 8$. (c): A pelvis bone, $\lambda = 5$.

6.3 ASPECT PRESERVATION DURING THINNING: A PARAMETER-FREE METHOD

Even though methods seen in section 6.2.2 can produce satisfactory results in most case, they usually require some user input. For example, if one wants to use the DL'MA as the second parameter of alg. 30, he or she will need to choose a λ value for thresholding the DL'MA map.

In the following, we will see a new method in the cubical complex, requiring no user input, for obtaining a skeleton from a cubical complex, yielding satisfactory visual properties. Moreover, the algorithm can be modified in order to accept user input, in case results need to be filtered more or less in some cases.

6.3.1 THE LIFESPAN OF A FACE

In the following, we define new notions in the cubical complex. The first one we present is the *death date* of a face.

Definition 6.3.1 *Let* $f \in X \preceq \mathbb{F}^n$, *the* death date of f in X, *denoted by* $\mathtt{Death}_X(f)$, *is the smallest integer* d *such that* $f \notin \mathtt{ParDirCollapse}(X, \emptyset, d)$.

The death date of a face indicates how many layers of free faces should be removed from a complex X, using alg. 27, before removing completely the face from X. We now define the *birth date* of a face:

Definition 6.3.2 *Let* $f \in X \preceq \mathbb{F}^n$, *the* birth date of f in X, *denoted by* $\mathtt{Birth}_X(f)$, *is the minimum between the smallest integer* b *such that* f *is a facet of* $\mathtt{ParDirCollapse}(X, \emptyset, b)$, *and* $\mathtt{Death}_X(f)$.

The birth date indicates how many layers of free faces must be removed from X with Alg.27 before transforming f into a facet of X (we consider a face "lives" when it is a facet). Finally, we can define the *lifespan* of a face :

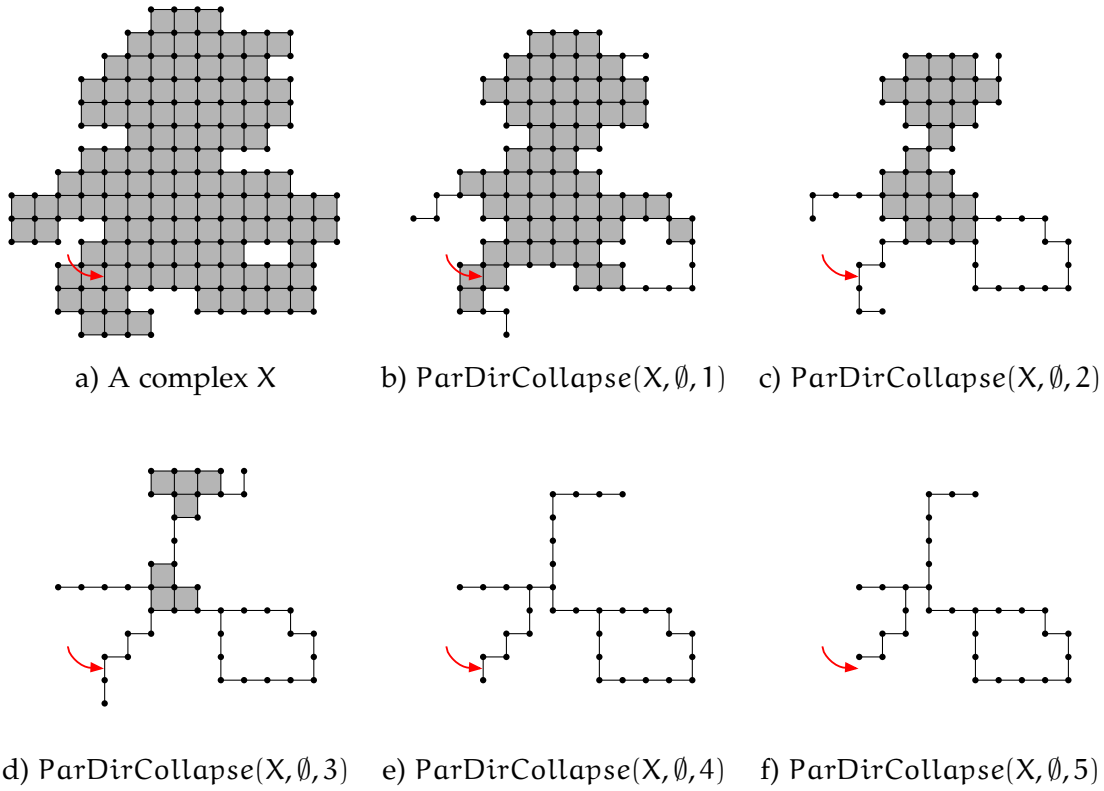
Definition 6.3.3 *Let* $f \in X \preceq \mathbb{F}^n$, *the lifespan of* f *in* X *is the integer*

$$\mathtt{Lifespan}_X(f) = \begin{cases} +\infty & \text{if } \mathtt{Death}_X(f) = +\infty \\ \mathtt{Death}_X(f) - \mathtt{Birth}_X(f) & \text{otherwise} \end{cases}$$

The three parameters previously defined are dependant on the order of direction and orientation chosen for algorithm $\mathtt{ParDirCollapse}$.

The lifespan of a face f of X indicates how many "rounds" this face "survives" as a facet in X, when removing free faces with algorithm 27. The lifespan, the death and the birth, as defined here, are dependant of Alg. 27, used for performing the thinning. It is of course possible to use another algorithm for performing the thinning, leading to other values of birth, death and lifespan. It is recommended, in order to have "comparable" values, to compute these three parameters with the same thinning technique.

Algorithm 31 computes the lifespan of all faces of a complex. The algorithm is not linear in time, however, a linear implementation of such algorithm exists and is presented in the appendix. On Fig. 78, we show a sequence of collapse allowing to compute the lifespan of an edge.



a) A complex X | b) ParDirCollapse(X, ∅, 1) | c) ParDirCollapse(X, ∅, 2)

d) ParDirCollapse(X, ∅, 3) | e) ParDirCollapse(X, ∅, 4) | f) ParDirCollapse(X, ∅, 5)

Figure 78: **Computing the lifespan of a face -** Let e be the edge of the complex X, pointed out by a red arrow on **a**. Two iterations of ParDirCollapse algorithm are necessary before e becomes a facet, as depicted on **c**. Therefore, $\mathrm{Birth}_X(e) = 2$. Five iterations of ParDirCollapse algorithm are necessary for removing e from the complex, as depicted in **f**. Therefore, $\mathrm{Death}_X(e) = 5$. We have $\mathrm{Lifespan}_X(e) = 3$.

The lifespan is a good indicator of how important a face can be in an object. Typically, higher the lifespan is, and more representative of an object's visual feature the face is. The lifespan, also called *saliency*, was used in [LCLJ10] (under the name "medial persistence") in order to propose a thinning algorithm in cubical complexes based on two parameters.

6.3.2 DISTANCE MAP AND OPENING FUNCTION

In addition to the lifespan of a face, the proposed homotopic thinning method will use information on distance between faces in order to decide if a face should be kept safe from deletion. We define hereafter the various notions needed for this, based on distance in the DT framework.

Two points $x, y \in \mathbb{Z}^n$ are 1-*neighbours* if the Euclidean distance between x and y is equal or inferior to 1 (also called direct neighbours). A 1-*path from* x *to* y is a sequence $\mathcal{C} = (z_0, ..., z_k)$ of points of $\mathbb{Z}^n$ such that $z_0 = x$, $z_k = y$, and for all $j \in [1; k]$, z_j and z_{j-1} are 1-neighbours. The length of $\mathcal{C}$ is k.

We set $d_1(x, y)$ as the length of the shortest 1-path from x to y. Let $S \subset \mathbb{Z}^n$, we set $d_1(x, S) = \min_{y \in S} d_1(x, y)$. The 1-*ball of radius* r *centered on* x is the set $\mathbb{B}^1_r(x) = \{y \in \mathbb{Z}^n | d_1(x, y) < r\}$. Remark that d_1 is indeed the so-called 4-distance in the 2d DT

Algorithm 31: $\mathrm{Lifespan}(X)$

Data: A cubical complex $X \preceq \mathbb{F}^n$
Result: The map giving the lifespan of all faces of X

```
for all f ∈ X do
    Death(f) = +∞;
    if f ∈ X^+ then
        Birth(f) = 0;
    end
    else
        Birth(f) = +∞;
    end
end
Y = X; l = 0;
while there exists free faces for Y do
    Y = ParDirCollapse(Y, ∅, 1);
    l = l + 1;
    for all f ∈ Y^+ such that Birth(f) = +∞ do
        Birth(f) = l;
    end
    for all f ∉ Y such that Death(f) = +∞ and f ∈ X do
        Death(f) = l;
    end
end
for all f ∈ X do
    Birth(f) = min(Birth(f), Death(f));
    if Death(f) = +∞ then
        Lifespan(f) = +∞;
    end
    else
        Lifespan(f) = Death(f) − Birth(f);
    end
end
return Lifespan;
```

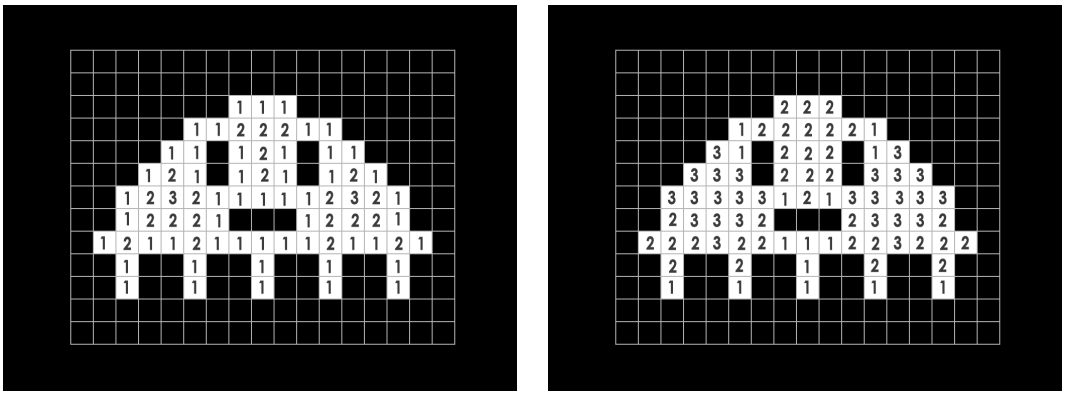
framework, and the 6-distance in the 3d DT framework. Given $X \subset \mathbb{Z}^n$, the *maximal 1-ball of* X *centered on* x is the set $\hat{\mathbb{B}}^1_X(x) = \mathbb{B}^1_{d_1(x,\overline{X})}(x)$.

We set, for all $x \in X$, $\omega_1(x, \overline{X}) = \max_{x \in \hat{\mathbb{B}}^1_X(y)} d_1(y, \overline{X})$: this value indicates the radius of the largest maximal 1-ball contained in X and containing x. If $x \in \overline{X}$, we set $\omega_1(x, \overline{X}) = 0$. The map ω_1 is known as the opening function (based on the 1-distance): it allows to compute efficiently results of openings by balls of various radius, and gives information on the local thickness of an object on each of its points. We show some examples of the opening function on Fig. 79 and 80.

Given $X \preceq \mathbb{F}^n$, the value of $\omega_1(x, \overline{X})$ of every $x \in X$ can be computed by performing successive dilations of values of the map d_1. The algorithm presented in Alg.32 is naive,

a) 4-distance map b) 4-opening map c) Decenterness function

Figure 79: **Global view of the opening function and of the decenterness function of a shape - a)** The 4-distance map of the shape depicted on Fig. 30a, p. 77. **b)** The 4-opening function of the same shape. **c)** The decenterness function of the same shape.



a) 4-distance map b) 4-opening map

Figure 80: **Detailed view of the opening function of a shape - a)** The 4-distance map of a shape, p. 77. **b)** The 4-opening function of the same shape.

and a more efficient implementation (linear in time depending on the image's size) is discussed in the appendix (see Sec. 9.2.5, p. 209).

Algorithm 32: $Opening(X)$

Data: A set $X \subset \mathbb{Z}^n$
Result: The 1-distance opening function of X, that is, for all $x \in X$, $\omega_1(x, \overline{X})$

```
for all x ∈ X do
    ω1(x) = −1
end
for all x ∈ X do
    for all y ∈ B̂1X(x) do
        if ω1(y) < d1(x, X̄) then
            ω1(y) = d1(x, X̄)
        end
    end
end
return e1;
```

In order to extend d_1 and ω_1 to the cubical complex framework, let us introduce the map Φ^{-1}, inverse of the bijective map Φ defined in Sec. 5.2.3. It is used to project any n-face of $\mathbb{F}^n$ into $\mathbb{Z}^n$. We indifferently use Φ^{-1} as a map from $\mathbb{F}^n_n$ to $\mathbb{Z}^n$, and as a map from $\mathcal{P}(\mathbb{F}^n_n)$ to $\mathcal{P}(\mathbb{Z}^n)$.

Given $X \preceq \mathbb{F}^n$, we set $X^v = \Phi^{-1}(X \cap \mathbb{F}^n_n)$: the set X^v is a subset of $\mathbb{Z}^n$. We define the map $\tilde{D_1}(X) : \mathbb{F}^n \to \mathbb{N}$ as an extension of d_1 to the cubical complex framework: for all $f \in \mathbb{F}^n$,

$$\tilde{D_1}(X)(f) = \begin{cases} d_1(\Phi^{-1}(f), \overline{X^v}) & \text{if } f \text{ is an } n\text{-face} \\ \max\limits_{g \in \check{f}^*} \tilde{D_1}(X)(g) & \text{else} \end{cases}$$

The same way, we define $\tilde{\Omega_1}(X) : \mathbb{F}^n \to \mathbb{N}$ as an extension of ω_1 to the cubical complex framework.

6.3.3 PARAMETER-FREE THINNING BASED ON THE LIFESPAN, OPENING FUNCTION AND DECENTERNESS

Thanks to these notions, we can now define sets of faces that will help preserve the visual aspect of an object during thinning. First, let us define the decenterness of a complex as the map $Decenter(X) = \tilde{\Omega_1}(X) - \tilde{D_1}(X)$. For each face of a complex X, the decenterness value of this face gives an information on how well a face is centered inside a visual feature of the object: the lower this value is, the better centered the face is. On Fig. 79, we give an example of the decenterness map of a shape, where the highest values are represented by the darkest colours.

Faces relevant of the visual aspect of a complex must have a high lifespan ("survive" long to the homotopic thinning process) and a low decenterness (is centered in the object). In a 3-dimensional complex (resp. a 2-dimensional complex), squares (resp. lines) whose lifespan is higher than the decenterness will be chosen as relevant of the surfacic (resp. curvilinear) parts of the complex.

Definition 6.3.4 *Given* $X \preceq \mathbb{F}^n$, *the* k-LC axis *(stands for "Lifespan Centerness") of* X *is the set*

$$\mathcal{LC}_k(X) = \{f \in X \mid \dim(f) = k \text{ and } \mathtt{Lifespan}_X(f) > \mathtt{Decenter}(X)(f)\}^-.$$

The LC-axis fails in selecting a good set of curves relevant of the curvilinear parts of a three-dimensional complex. Indeed, given a 3-complex X, the set $\mathcal{LC}_1(X)$ contains generally too many curves. This happens because the algorithm `ParDirCollapse` "takes more time" to eliminate lines than to eliminate squares in a 3-complex. Consequently, lines of a 3-complex tend to have a high lifespan even though they are not representative of any curvilinear part of the complex. The thicker is the input object, and the more important is the phenomenon.

Lines relevant of the curvilinear parts of a 3-complex have a high lifespan and a low decenterness, especially in the thick parts of the complex. Lines whose lifespan is higher than the decenterness added to the local thickness of the complex will be relevant of the curvilinear parts of the complex.

Definition 6.3.5 *Given* $X \preceq \mathbb{F}^n$, *the* k-LOC axis *(stands for "Lifespan Opening Centerness") of* X *is the set*

$$\mathcal{LOC}_k(X) = \{f \in X \mid \dim(f) = k \text{ and } \mathtt{Lifespan}_X(f) > \tilde{\Omega}_1(X)(f) + \mathtt{Decenter}(X)(f)\}^-.$$

The various sets previously defined represent faces which should be kept safe from deletion during homotopic thinning of a complex in order to obtain a pruned skeleton containing visual features from the original object. The set $\mathcal{LC}_1$ of a 2-complex represents curvilinear parts to keep in this complex, the set $\mathcal{LOC}_1$ of a 3-complex represents curvilinear parts to keep in this complex, and the set $\mathcal{LC}_2$ of a 3-complex represents surfacic parts to keep in this complex.

Lemma 6.3.6 *Given* $X \preceq \mathbb{F}^n$, *for every* $k < n$, $\mathcal{LC}_k(X) \subseteq \mathtt{CollapseFacet}(X, \emptyset)$ *and* $\mathcal{LOC}_k(X) \subseteq \mathtt{CollapseFacet}(X, \emptyset)$.

Lemma 6.3.6 is straightforward once observed that $\mathtt{CollapseFacet}(X, \emptyset)$ is indeed the set of k-faces of X ($k < n$) which have a strictly positive lifespan in X. Based on these sets, we propose three algorithms: one for computing a 1d skeleton from a bi-dimensional complex, one for computing a 2d skeleton (which can contain 1d parts) from a three-dimensional complex, and one for computing a curvilinear skeletons from a three-dimensional complex. This last algorithm should produce one-dimensional results (a set of lines) however, the output can be a two-dimensional complex.

Algorithm 33: `1DSkeleton(X)`

Data: A cubical complex $X \preceq \mathbb{F}^2$
Result: A cubical complex $Y \preceq \mathbb{F}^2$ such that $\dim Y \leqslant 1$

```
1 W = LC_1(X);
2 ParDirCollapse(X, W, +∞);
3 return X;
```

Algorithm 33 produces as output a 1d complex, result of the homotopic thinning of a bi-dimensional complex. The output skeleton is a set of lines representative of the

Algorithm 34: SurfaceSkeleton(X)

Data: A cubical complex $X \preceq \mathbb{F}^3$
Result: A cubical complex $Y \preceq \mathbb{F}^3$ such that $\dim Y \leqslant 2$
1 $W = \mathcal{LC}_2(X) \cup \mathcal{LOC}_1(X)$;
2 ParDirCollapse(X, W, $+\infty$);
3 **return** X;

Algorithm 35: CurvilinearSkeleton(X)

Data: A cubical complex $X \preceq \mathbb{F}^3$
Result: A cubical complex $Y \preceq \mathbb{F}^3$ such that $\dim Y \leqslant 2$
1 $W = \mathcal{LOC}_1(X)$;
2 ParDirCollapse(X, W, $+\infty$);
3 **return** X;

shape (and of the topology) of the input. Algorithm 34 produces as output a 2d or 1d complex, result of the homotopic thinning of a three-dimensional complex. The skeleton produced by this algorithm contains squares representative of the surfacic parts of the input, and lines representative of the curvilinear parts of the input.

Finally, algorithm 35 produces as output a 2d or 1d complex, result of the homotopic thinning of a three-dimensional complex. The skeleton produced by this algorithm should be a set of lines describing the shape of the input, however, it may contain squares representative of the topology of the initial object (if the input complex contains a cavity, the skeleton will contain surfacic parts).

Figure 79c depicts the decenterness function of a shape. Dark zones of the image are zones where it is difficult for the faces to survive the thinning presented in Alg. 33 (high lifespan necessary), while bright zones represent zones where faces easily survive the thinning (low lifespan is sufficient).

The following is a direct consequence of Prop. 5.2.6 p.126, Prop. 6.2.1 p.136 and Lem. 6.3.6 p.144.

Proposition 6.3.7 *Given* $X \preceq \mathbb{F}^2$, *algorithm* 1DSkeleton(X) *produces a* 1*-dimensional skeleton.*

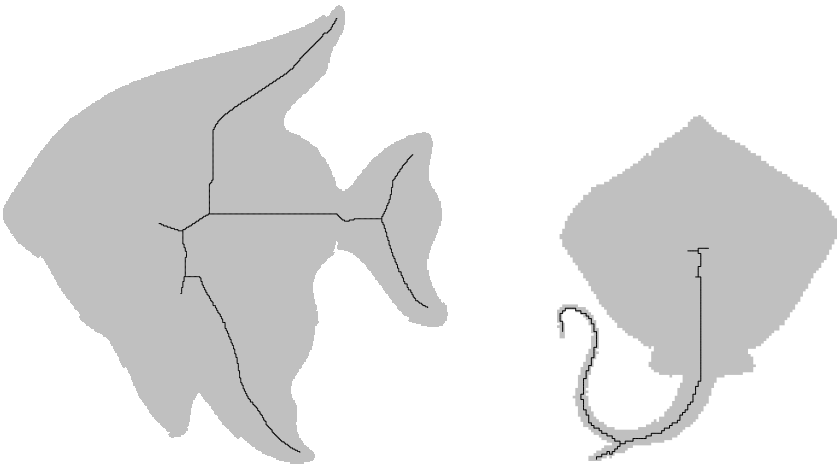
Given $X \preceq \mathbb{F}^3$, *algorithms* SurfaceSkeleton(X) *and* CurvilinearSkeleton(X) *produce a* 2*-dimensional skeleton.*

Despite the use of constraint sets (sets of faces which should not be removed during thinning) in the three algorithms, they produce thin results.

6.3.4 VISUAL RESULTS OF THINNING ALGORITHMS AND POSSIBLE ENHANCEMENTS IN 3D

We show some results of the three algorithms on Fig. 81 for 2d, and Fig 96, 97, 98, 99, 100 and 102, all at the end of this chapter, for 3d. Visually, algorithms 33 and 34 achieve well in preserving the visual aspect of the input. Moreover, the result of algorithm 34 success in containing only lines in the curvilinear parts of the input, and squares in the surfacic parts.

On the objects of our test collection, algorithm 35 always gave a curvilinear output. When the input is a curvilinear object, the output skeleton is a curvilinear object describing well the shape of the original object. On surfacic parts, the skeleton produced by Alg. 35 may contain spurious branches.



a) b)

Figure 81: **A few examples of skeletons obtained with Alg. 33**. The original shape is shown in grey, the skeleton is in black. The result of **a** can be compared with Fig. 36 p. 84, and the result of **b** can be compared with Fig. 35 p. 83. On the PDF version, a zoom on the figures shows that both skeletons are made of edges and vertices (no voxels).

It is possible, for the three algorithms, to propose a version with a filtering parameter. This is also discussed in the appendix (see Sec. 9.2.4, p. 208) and in the general conclusion.

6.3.4.1 EROSION ON THE BORDERS

As it is the case for most thinning methods, our algorithm produces an "eroded" output, which means that the skeleton does not "stick" to the borders of the original complex. Although a methodology for completely reducing this erosion phenomenon is under investigation, we can already propose a general method for reducing the erosion around the object's holes.

On Fig. 82a, we show a surfacic object X with holes. On Fig. 82b, we show the results of Alg. 34 applied to X: the skeleton S does not stick to the original borders of X. In order to "glue back" the skeleton to the borders of the holes of X, it is possible to fill these holes with a hole-filling algorithm (see [JCB10] for example), thus obtaining the complex X', and compute the skeleton W of X' with Alg. 34. Then, we compute the skeleton of X with Alg. 27, using the set W as an inhibitor set. The result is shown on Fig. 82c.

6.4 EVALUATING THINNING ALGORITHMS PERFORMANCES

As we did for the lambda-medial axis in section 4.4, we compare our parallel parameter-free thinning methods with other parallel thinning methods, in 2d and 3d frameworks. Previous works comparing thinning methods were already proposed ([Tam78], [DP81],

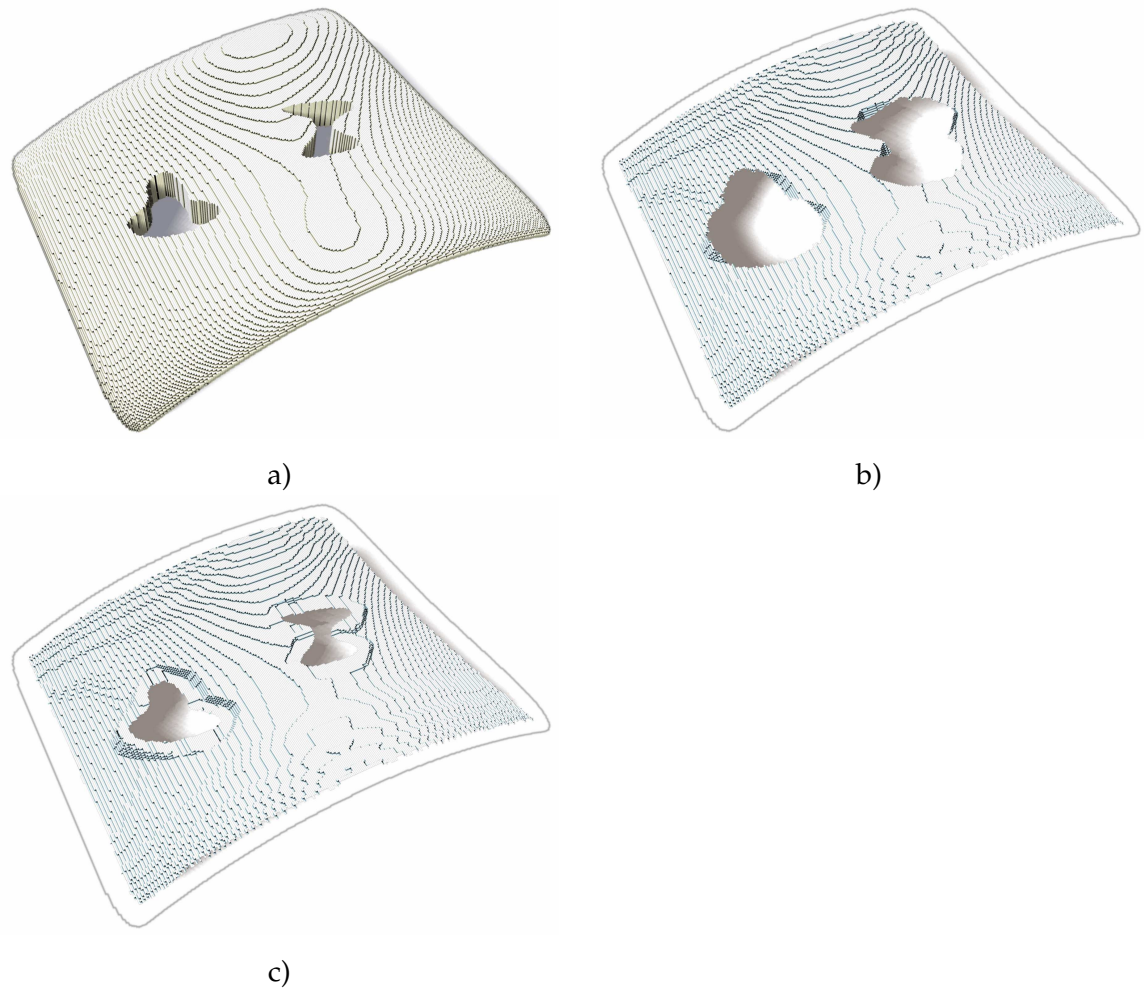


Figure 82: **Avoiding holes enlargement during thinning - a)** The original shape. **b)** The homotopic thinning of the shape using Alg. 34. **c)** With a hole-filling algorithm as post-processing, it is possible to reduce the enlargement of the shape's holes during the thinning.

[Hil83]), but they either focus on implementation or referencing specific properties of thinning algorithms (symmetric, asymmetric, etc). We propose hereafter a comparison based on the stability to rotation and to noise (the detailed methodology was previously described in Sec. 4.4), and also based on the visual results of algorithms. We point out the fact that the noise added to the object preserves the topology of the input (and can bring strong contour deformation) while the rotation algorithm does not necessarily preserve the topology (and contour deformation remains limited).

As we explain in the following, another criteria than stability must be taken into account when comparing thinning algorithms.

6.4.1 NO COMMON GROUND BETWEEN ALGORITHMS

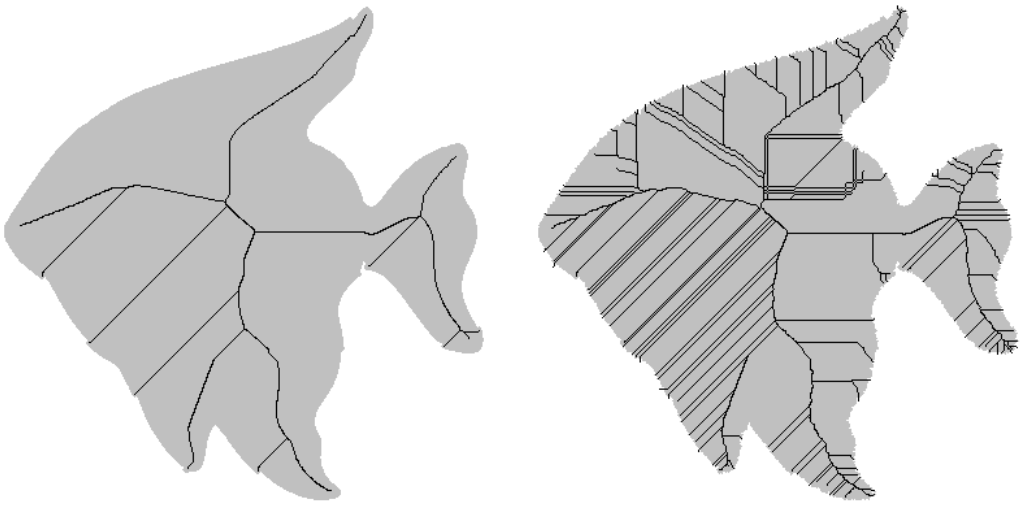
In section 4.4, various medial axes were tested and evaluated following this method: a filtering parameter was chosen in order for medial axes to give same residual, and then, stability to noise and rotation was tested. The major difference between parameter-free thinning algorithms and medial axes is the filtering criterion: when testing medial axes, a "common ground" can be built by choosing a filtering criterion giving the same residual when trying to rebuild the original object from one of its medial axis using the reverse Euclidean distance transform.

Parameter-free thinning algorithms do not have any filtering criterion, therefore, it is impossible to decide to compare them based on the residual they give when trying to rebuild the original object, as this residual is fixed for each algorithm (no parameter can be tuned in order to change the residual).

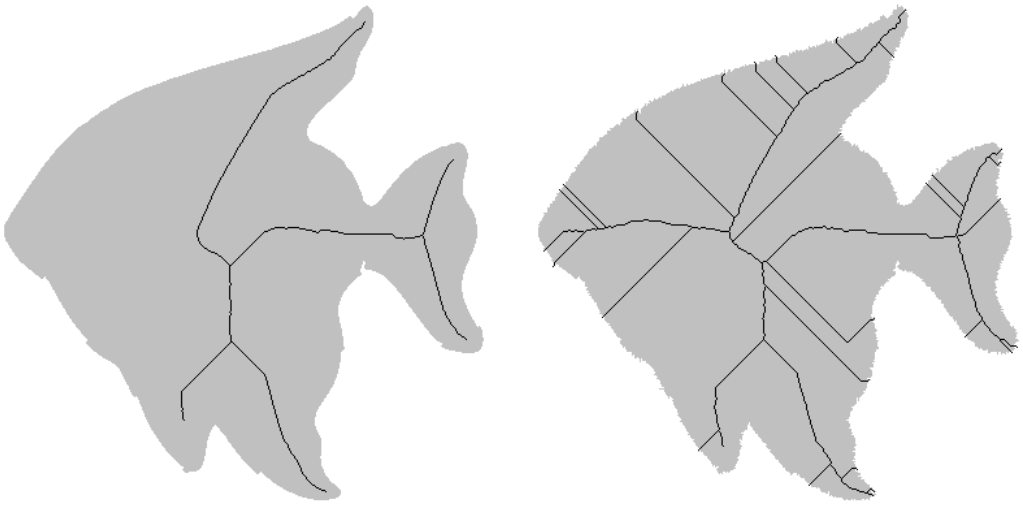
The lack of common ground between algorithms will give an advantage, in our evaluation tests, to algorithms producing "noisy" skeletons, with many spurious branches and surfaces. Indeed, our tests (noise and rotation) both consists in producing small perturbations of the object's border and measuring the effect on the skeleton.

A common idea is that a skeleton which does not change much under object's perturbations is a good skeleton. However, this assertion can be debated. Indeed, consider the example shown on figure 83. Two skeletons are shown on the left column: one with many spurious branches (on top), and one with less spurious elements (on the bottom). The second one may be commonly thought as better, as it still keeps important information about the original shape, while containing less points. Adding noise to the object's border have an effect on both skeletons, and, as in section 4.4, we measure this effect using a dissimilarity measure.

Results show that the "noisy" skeleton (top row of Fig. 83) changed less after the noise addition than the other skeleton, which was thought to be better. Indeed, the noise added on the object's border has an impact on the skeleton, but the measure of this impact is "diluted" into the existing noise of the skeleton (new branches does not represent an important change of the original skeleton). On the other hand, the impact of the noise on the filtered skeleton is greater. In order to have a fair evaluation of the skeletonization algorithms, we need to take into account, in addition to the stability to noise and rotation, the visual aspect of the skeleton.



a) Alg. [Rut66] on original shape b) Alg. [Rut66] on noisy shape



c) Alg. [Ros75] on original shape d) Alg. [Ros75] on noisy shape

Figure 83: **Noisy skeletons are more stable to noise addition - a)** The original shape and its skeleton (in black) computed with the algorithm presented in [Rut66]. **b)** The same shape with noise added to its boundary, and its skeleton (in black) computed with the algorithm presented in [Rut66]. The dissimilarity between the two skeletons is approximately equal to 20. **c)** The original shape and its skeleton (in black) computed with the algorithm presented in [Ros75]. **d)** The same shape with noise added to its boundary, and its skeleton (in black) computed with the algorithm presented in [Ros75]. The dissimilarity between the two skeletons is approximately equal to 40.

6.4.2 EVALUATING VISUAL ASPECT OF SKELETONS

When evaluating the visual aspect of a skeleton, noisy skeletons must have a bad grade compared to filtered skeletons. Therefore, the evaluation should definitely give a score to each skeletons based on the number of points of the skeleton (lower being better). On the other hand, skeletons which do not carry enough visual information about the original object (because of a too high filtering like, for example, ultimate skeletons) should be penalised. Our evaluation should therefore take into account some information about the quality of the reconstruction of the original object from its skeleton using the reverse Euclidean distance (the higher the better).

6.4.2.1 QUALITY OF THE RECONSTRUCTION

The quality of the reconstruction is evaluated by counting, when reconstructing the original object from its skeleton, the number of missed points situated on the border of the original object. Given a binary object $X \subset \mathbb{Z}^n$ and $S \subseteq X$, the *normalized missed reconstructed border of* S *in* X is

$$\mathcal{R}_{\mathcal{B}}{}_X^S = \frac{|Border(X) \backslash Border(REDT_X(S))|}{|Border(X)|}$$

We remind the reader that the reverse Euclidean distance transform (REDT), defined in section 4.4, allows to know how much information of X can be retrieved from S. The result $\mathcal{R}_{\mathcal{B}}{}_X^S$ is a real value between 0 (perfect reconstruction) and 1 (bad reconstruction).

6.4.2.2 SIZE OF THE SKELETON

The size of the skeleton can be easily computed by counting the number of points of the skeleton. In order to normalize this quantity, we must take into account a "reference" skeleton. In the digital topology framework, a skeleton constrained to contain all the centers of maximal balls will achieve a perfect reconstruction: we choose the skeleton obtained with the Euclidean distance transform as a priority function and the Euclidean medial axis as inhibitor set, as a reference skeleton.

Let X be a subset of $\mathbb{Z}^n$ and $S \subseteq X$, the *normalized skeleton size of* S *in* X is

$$\mathcal{C}_X^S = \frac{|S|}{|\mathtt{CenteredMedialSkeleton}(X)|}$$

6.4.2.3 VISUAL QUALITY FACTOR OF A SKELETON

Given a subset X of $\mathbb{Z}^n$, and a subset S of X (S should be a skeleton), the *visual quality factor* of S in X is

$$\mathcal{VQF}_X(S) = \sqrt{\mathcal{C}_X^S{}^2 + \mathcal{R}_{\mathcal{B}}{}_X^S{}^2}$$

The lower the visual quality factor of a skeleton is, the better.

In order to test the robustness of this evaluation factor, we test it with the Euclidean medial axis, and the lambda' medial axis. Given a subset X of $\mathbb{Z}^2$ or $\mathbb{Z}^3$, we compute $\mathtt{EMA}(X, \rho)$ (the Euclidean medial axis of X thresholded at the value ρ) and we then compute $\mathcal{VQF}_X(\mathtt{EuclideanSkeleton}(X, \mathtt{EMA}(X, \rho)))$, for various ρ. On Fig. 84, we show, for a given shape, how the visual quality factor of the skeleton evolves with the filtering

parameter ρ. In Fig. 85, we give the results of 3d skeletons constrained to contain the Euclidean medial axis achieving the best visual quality factor. In most cases, the skeleton misses important parts of the shape: this is mainly due to the fact that filtering the Euclidean medial axis usually requires to choose between spurious elements or weak reconstruction.

The same process repeated with the DL'MA, instead of the EMA, fares better results. From Fig 58 p. 113, up to Fig. 63 p. 118, each skeleton at position c was obtained with the DL'MA achieving the best visual quality factor for the shape.

In the following, we use the visual quality factor as a criteria for evaluating how visually good is the result produced by a thinning technique.

6.4.3 RESULTS OF EVALUATION

6.4.3.1 GENERAL EVALUATION METHODOLOGY

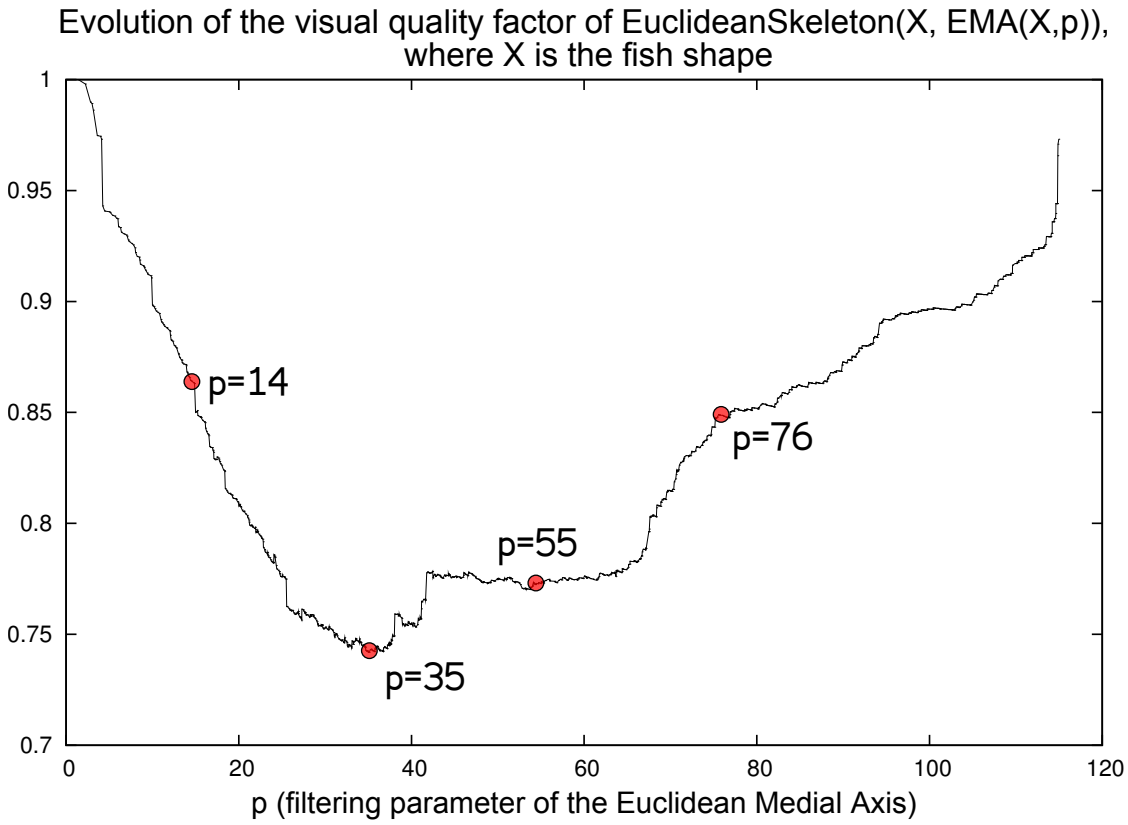
In the following, we compare robustness to noise and rotation of various 2d and 3d parameter-free thinning algorithms. Our algorithms operate in the cubical complexes framework, while the other algorithms operate in the digital topology framework. In order to obtain a fair comparison between algorithms, we need to embed the results of our thinning algorithms into the DT framework. Algorithm 36 embeds a cubical complex $S \preceq X$ into $\mathbb{Z}^n$. Typically, the input $X \preceq \mathbb{F}^n$ is a pure n-complex and S is a collapse of X such that $\dim(S) = (n-1)$. Algorithm 36 scans each face f of S, and tries to find the last n-face of S which contained f during the collapse sequence: this face is added to a set R if there does not already exist a face in R which contains f. The set R is a set of n-face which can be embedded in the DT framework thanks to the Φ^{-1} application (see Sec. 6.3.2).

Algorithms 37, 38 and 39 allows to use previously defined thinning algorithms in the cubical complex framework in order to obtain "thin" subsets of $\mathbb{Z}^2$ and $\mathbb{Z}^3$. In order to do so, a subset X of $\mathbb{Z}^2$ or $\mathbb{Z}^3$ is embedded in the cubical complex framework using the Φ application, resulting in a pure 2 or 3-complex $\Phi(X)$. This complex is thinned using one of the thinning algorithm previously defined, and the resulting skeleton is embedded in the DT framework using Alg. 36. The result W is not necessarily homotopic to the initial object X: a last thinning step of X constrained to contain W must take place in order to obtain a skeleton of X as output.

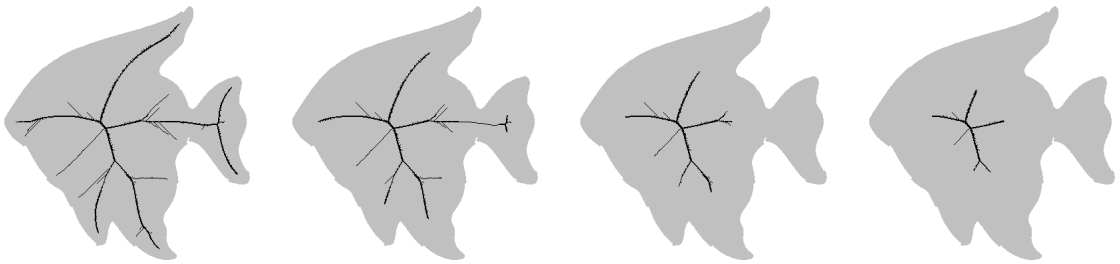
Thanks to these algorithms, the results produced by thinning algorithms working on voxel objects and the results produced by thinning algorithms working on cubical complexes can be fairly compared. Results of 3d thinning based on these algorithms can be seen on Fig. 95, 96 and 100, at the end of this chapter.

6.4.3.2 RESULTS IN 2D

We compare our 2d thinning algorithm (see Alg. 37), with other 2d parallel thinning algorithms. The comparison covers stability properties to noise, to rotation, and visual quality. Visual examples of the behaviour of each tested thinning algorithm to noise addition and rotation is shown on Fig. 92 and 93, at the end of this chapter. The results are summarised in table 5 p. 155, as well as in Fig. 86 and 87. The algorithm "Euclidean medial axis" is the homotopic thinning guided by the Euclidean distance



a)



b) $\rho = 14$ c) $\rho = 35$ d) $\rho = 55$ e) $\rho = 76$

Figure 84: **Evolution of the visual quality factor - a)** Evolution of the visual quality factor of $\mathrm{EuclideanSkeleton}(X, \mathrm{EMA}(X, \rho))$ in X, against the filtering parameter ρ, where X is the fish shape shown in Fig. 83. There exists a value ρ where the visual quality reaches a minimal value: this parameter gives the filtered Euclidean medial axis with has the best visual quality factor. **b), c), d), e)** Euclidean skeletons constrained to an Euclidean medial axis filtered at various values. The shape **c)** achieves the best visual quality factor.

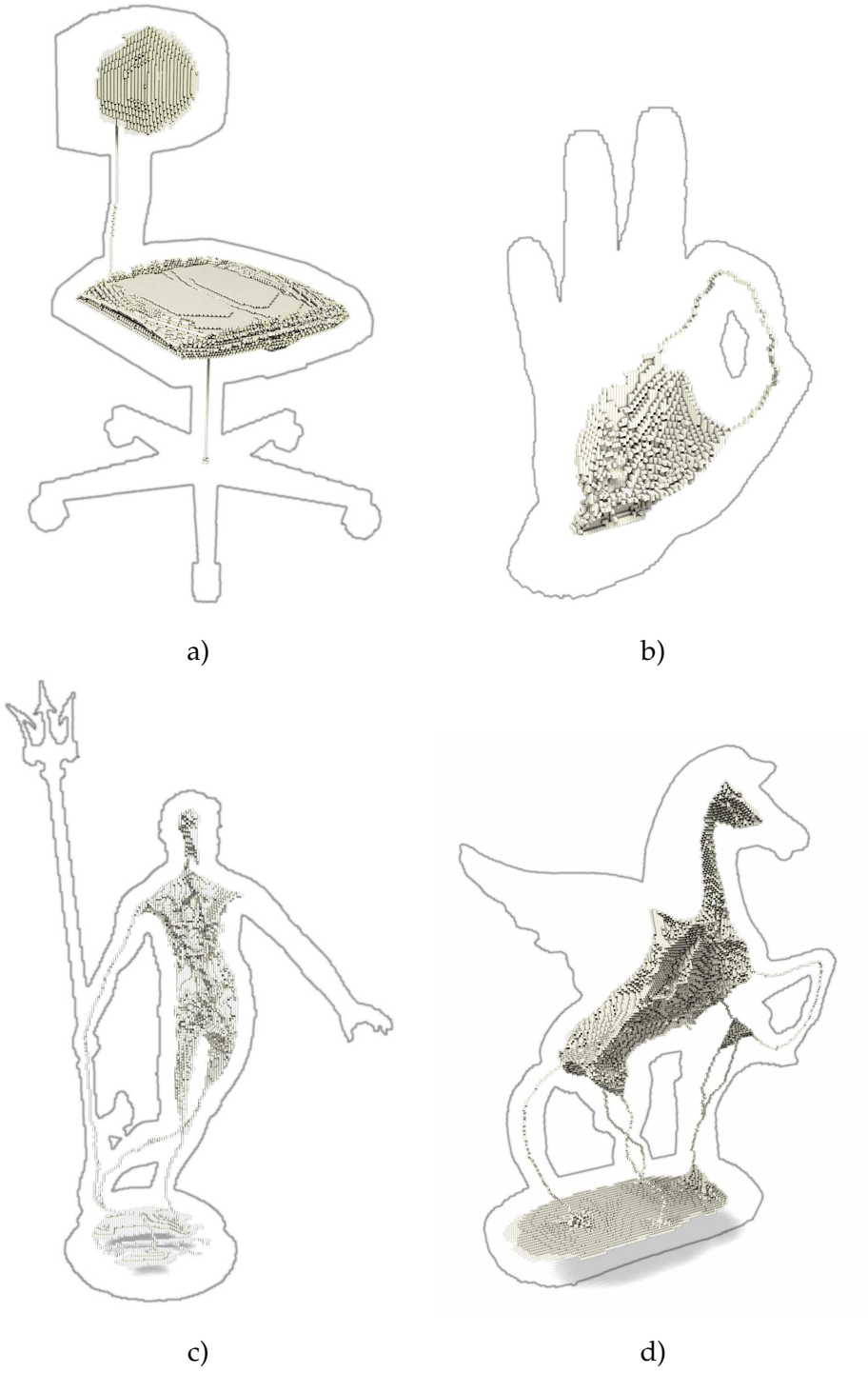


Figure 85: **Visual quality factor and Euclidean medial axis -** Various shapes X were skeletonized using a thinning algorithm constrained by a filtered Euclidean medial axis. In each case, the chosen filtered medial axis is the one achieving the best visual quality factor for the shape. **a)** The original image can be found in Fig. 60, p.115. **b)** The original image can be found in Fig. 100, p.178. **c)** The original image can be found in Fig. 58, p.113. **d)** The original image can be found in Fig. 62, p.117.

Algorithm 36: $\mathrm{CCtoDT}(X, S)$

Data: A cubical complex $X \preceq \mathbb{F}^n$, a complex $S \preceq X$
Result: A set $V \subseteq \mathbb{R}^n$

```
R = ∅;
C = {e ∈ X | dim(e) = n};
foreach f ∈ S do
    B = f̌ ∩ C;
    if B ∩ R == ∅ then
        Let y ∈ arg max_{g∈B} Death_X(g);
        R = R ∪ {y};
    end
end
return Φ(R)^{-1};
```

Algorithm 37: $\mathrm{DT2-1}(X)$

Data: A set $X \subseteq \mathbb{Z}^2$
Result: A skeleton of X

```
C = 1DSkeleton(Φ(X)^-);
W = CCtoDT(Φ(X), C);
D = Euclidean distance transform of W̄;
return Basic Thinning with Priority(X, W, −D, 8);
```

Algorithm 38: $\mathrm{DT3-2}(X)$

Data: A set $X \subseteq \mathbb{Z}^3$
Result: A skeleton of X

```
C = SurfaceSkeleton(Φ(X)^-);
W = CCtoDT(Φ(X), C);
D = Euclidean distance transform of W̄;
return Basic Thinning with Priority(X, W, −D, 26);
```

Algorithm 39: $\mathrm{DT3-1}(X)$

Data: A set $X \subseteq \mathbb{Z}^3$
Result: A skeleton of X

```
C = CurvilinearSkeleton(Φ(X)^-);
W = CCtoDT(Φ(X), C);
D = Euclidean distance transform of W̄;
return Basic Thinning with Priority(X, W, −D, 26);
```

map and retaining all elements of the Euclidean medial axis. The visual quality factor is computed only on the original images (not on the deformed images).

Results show that our algorithm fares, in average, the best stability results (to both noise and rotation, with both distance measures). We must however point out the fact that, when noise level is very high, the stability of our algorithm is bad compared to other algorithms: indeed, when noise level becomes high, the algorithm cannot make

the difference between noise and visual features of the object. Some spurious branches appear on our skeleton, and stability drops. This matter is discussed in the general conclusion.

Even if our algorithm does not score the best visual quality factor (the best visual quality is reached by [JC92]), it does produce visually good results (it ranks third based on the visual quality factor). Indeed, our algorithm has a little low visual quality factor because it tends to produce skeletons with little low reconstruction capacities, and with few points. Thanks to this, our 2d thinning algorithm has globally the best stability to noise (as long as the deformation is not too high) and rotation, and produces visually satisfying skeletons.

	Rotation		Noise		$\mathcal{VQF}_X(S)$
	Haus.	Dub.	Haus.	Dub.	
[Ros75]	16.34	5.02	25.18	9.65	0.563
[Pav80]	13.56	3.39	15.58	4.50	0.881
[CWSI87]	16.65	4.50	19.16	5.92	0.760
[HSCP87]	11.90	2.89	19.73	6.98	0.505
[Hal89]	15.85	4.27	19.74	7.05	0.532
[JC92]	9.60	2.52	18.08	6.24	0.49
[GH92]	15.45	4.15	19.10	6.47	0.533
[EM93]	13.59	3.49	15.60	4.53	0.817
[JC93]	13.62	3.53	15.60	4.54	0.797
[BM99]	15.29	3.9	19.00	6.34	0.627
Bertrand 2007	13.53	3.46	15.55	4.53	0.853
[BC08] (AK2)	13.50	3.49	15.53	4.54	0.864
DT2-1	6.32	1.81	12.20	3.51	0.528
Euclidean medial axis	10.89	2.12	13.71	3.73	1.000

Table 5: **Results of tests on 2d thinning algorithms -** Results of the tests performed on the 216 shapes of Kimia database. Each shape X was thinned using one of the algorithms A presented one the first column of the table. Columns 2 and 3 present the average Dubuisson and Hausdorff distance between $R_\theta(A(X))$ and $A(R_\theta(X))$, with θ varying from 0 to 90 degrees. Columns 4 and 5 present the average Dubuisson and Hausdorff distance between $A(X)$ and $A(E(X,n))$, with n varying from 0 to 10 percent of the shape's area. Column 6 present the average visual quality factor of $A(X)$ in X. Lowest values are highlighted in grey.

6.4.3.3 RESULTS IN 3D: GENERAL SKELETONS

We compare our 3d thinning algorithm (see Alg. 38), with other 3d parallel thinning algorithms. The comparison covers stability properties to noise, to rotation, and visual quality. Visual examples of the behaviour of each tested thinning algorithm to noise addition is shown on Fig. 94, at the end of this chapter. The results are summarised in table 6 p. 158, as well as in Fig. 88 and 89. The algorithm "Euclidean medial axis" is the

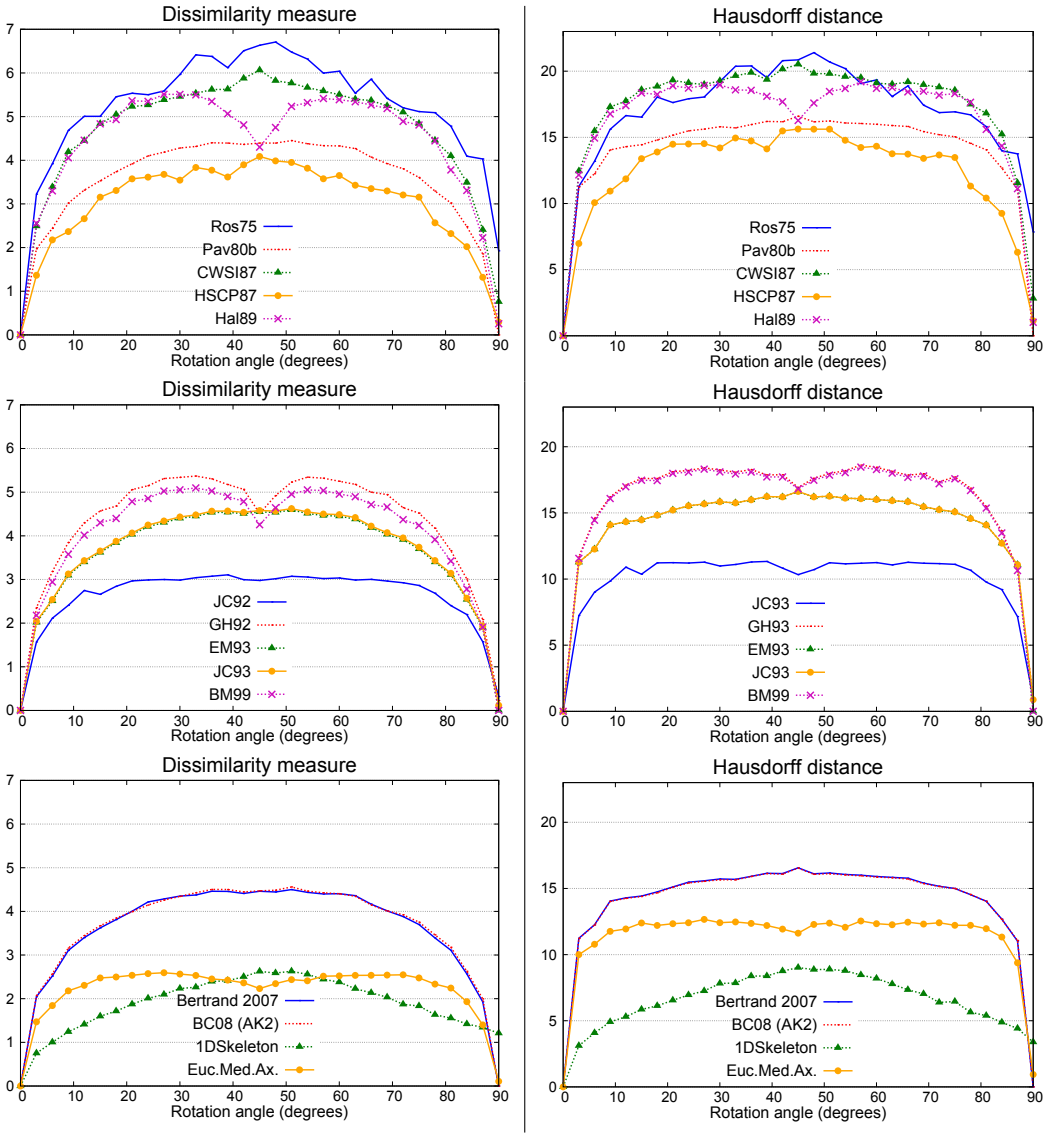


Figure 86: **Stability of 2d thinning algorithms to rotation -** Dissimilarity measure (on the left) and Hausdorff distance (on the right) between $A(R_\theta(X))$ and $R_\theta(A(X))$ on the 216 shapes of Kimia's 2D image database (shape X), using various thinning algorithms A.

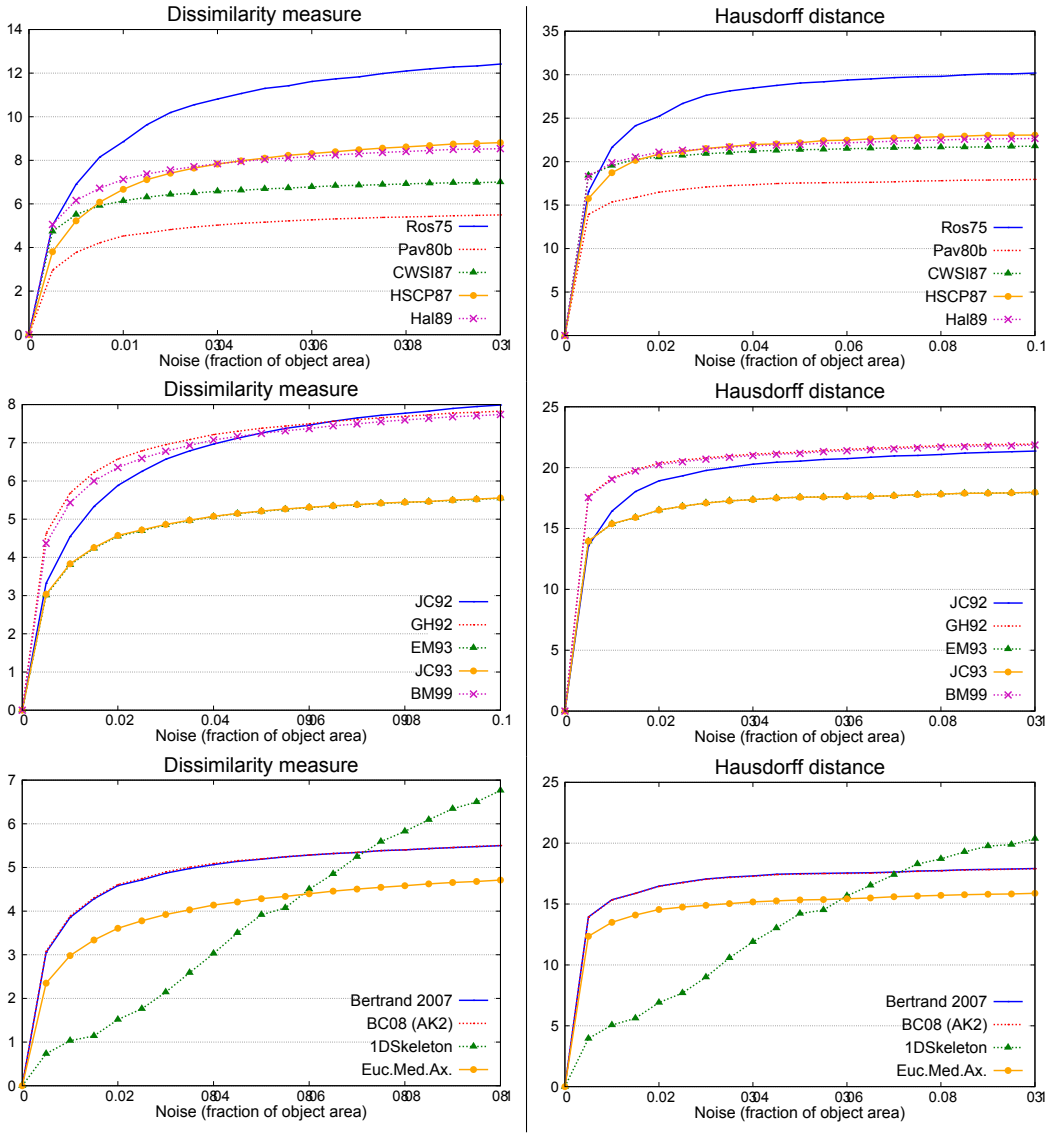


Figure 87: **Stability of 2d thinning algorithms to noise -** Dissimilarity measure (on the left) and Hausdorff distance (on the right) between $A(E(X,n))$ and $E(A(X),n)$ on the 216 shapes of Kimia's 2D image database (shape X), using various thinning algorithms A.

homotopic thinning guided by the Euclidean distance map and retaining all elements of the Euclidean medial axis. The visual quality factor is computed only on the original images (not on the deformed images).

Although the Euclidean medial axis fares good results on some points, it can be ignored: its good performances are explained by its bad visual quality factor (as detailed in Sec. 6.4.1). Among the algorithms with good visual quality factor (with visual quality factor inferior to 1, therefore producing filtered skeleton), our algorithm fares the best results, in average, for rotation stability and noise stability. We point out the fact that our thinning method is outperformed by [Palo2] and [BCo6] (SK3) for noise stability measured with Hausdorff distance: this can be explained by the fact that our algorithm produces filtered skeleton, with less points than other skeletons, and the apparition of spurious branches when noise level is high gives bad results on noise stability measured with the Hausdorff distance.

Our thinning method does not have the best score for visual quality factor: this is explained by the fact that our method produces highly filtered skeleton, with lower reconstruction capacity than other algorithms. However, as shown on Fig. 94, it gives the most "understandable" skeleton, even when the noise level increases. Our algorithm produces visually good skeletons, and has, among algorithms producing filtered skeletons (visual quality factor inferior to one), the best average stability to noise and rotations.

	Rotation		Noise		$\mathcal{VQF}_X(S)$
	Haus.	Dub.	Haus.	Dub.	
[Palo2]	8,87	2,11	9,46	2,44	0,62
[BCo6] (SK3)	10,50	2,36	11,05	3,28	0,52
[Palo8]	8,66	1,53	9,53	1,98	1,02
[BC] (DSK3)	10,81	1,61	12,79	3,57	0,41
DT3-2	6,60	1,32	11,72	2,01	0,58
Euclidean Medial Axis	7.25	1.06	8.41	1.72	1.00

Table 6: **Results of tests on 3d thinning algorithms -** Results of the tests performed on 3d shapes. Each shape X was thinned using one of the algorithms A presented one the first column of the table. Columns 2 and 3 present the average Dubuisson and Hausdorff distance between $R_\theta(A(X))$ and $A(R_\theta(X))$, with θ varying from 0 to 90 degrees. Columns 4 and 5 present the average Dubuisson and Hausdorff distance between $A(X)$ and $A(E(X,n))$, with n varying from 0 to 10 percent of the shape's volume. Column 6 present the average visual quality factor of $A(X)$ in X. Lowest values are highlighted in grey.

6.4.3.4 RESULTS IN 3D: CURVILINEAR SKELETONS

The visual quality factor is not relevant in this case Evaluation of curvilinear skeletons in 3d is more difficult. Performing a curvilinear skeletonization on a curvilinear shape makes sense, and the evaluation methodology used previously holds. However, a curvilinear skeletonization on a general shape which contains surfacic parts can have two goals: either the skeleton should allow a good reconstruction of the initial object

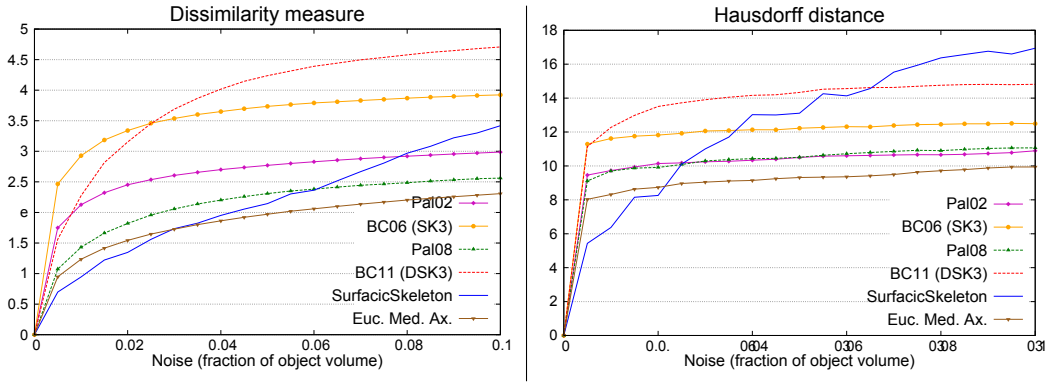


Figure 88: **Stability of 3d thinning algorithms to noise -** Dissimilarity measure (on the left) and Hausdorff distance (on the right) between $A(E(X,n))$ and $E(A(X),n)$ on 3d shapes of our database (shape X), using various 3d thinning algorithms A.

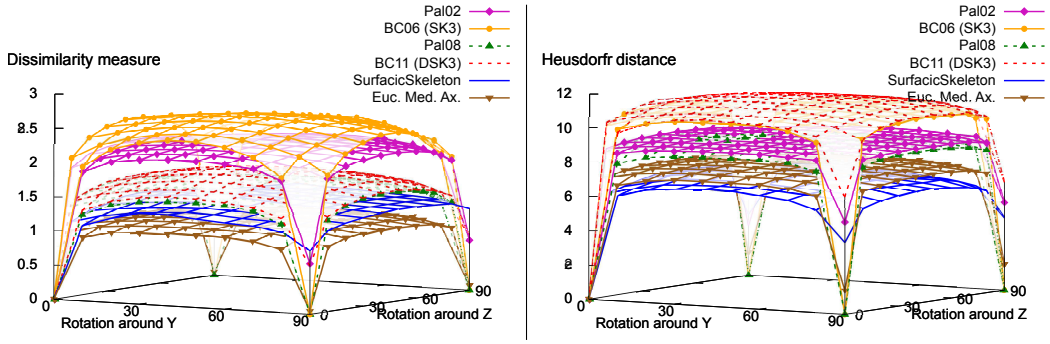


Figure 89: **Stability of 3d thinning algorithms to rotation -** Dissimilarity measure (on the left) and Hausdorff distance (on the right between $A(R_\theta(X))$ and $R_\theta(A(X))$ on 3d shapes of our database (shape X), using various 3d thinning algorithms A.

(and should therefore contain spurious branches in the surfacic parts) or it should remain as simple as possible and contain few spurious branches (specifically in the surfacic parts).

The visual quality factor is built around a "reference point" which is the result of the skeletonization constrained to contain all points of the medial axis; indeed, the number of points and the reconstruction capacity of the tested skeletons are normalized depending on the results of this particular skeleton (this skeleton achieves total reconstruction of the border). However, this skeleton is generally a surface skeleton when the input shape contains surfacic parts, and using a surface skeleton as "reference point" for comparing curvilinear skeletons does not make sense. Doing so will give advantage to spurious curvilinear skeletons (who, even though they have lots of points, have much less points than a surface skeleton, and propose a fair reconstruction) over filtered skeletons (little points, but bad reconstruction of surfacic parts).

For this reason, we don't use the visual quality factor in this evaluation: instead, we show, for each skeletons, the reconstruction capacity and the number of points of the skeleton (with no normalization).

Results We compare our 3d thinning algorithm (see Alg. 39), with other 3d curvilinear parallel thinning algorithms. The comparison covers stability properties to noise, to rotation, and visual quality (reconstruction capacity and number of points of the skeleton). Visual examples of the behaviour of each tested thinning algorithm to noise addition is shown on Fig. 94, at the end of this chapter. The results are summarised in table 7 p. 160, as well as in Fig. 90 and 91.

In average, the best stability results are achieved by [BC] (CK3). However, the number of points of the skeleton is, in average, far superior to other skeletons: as shown on Fig. 94, this algorithm produces skeletons with many spurious branches (therefore achieving a good reconstruction), which explains this stability to noise and rotation (see Sec. 6.4.1). If we ignore this algorithm, the second best results are achieved, in average, by our thinning: lowest number of points in the skeleton (while keeping a correct reconstruction capacity, given the fact that it is a curvilinear thinning algorithm), best stability to noise, and first/second best algorithm for rotation stability (outperformed by [PTHS06] for rotation stability measured with the Dubuisson dissimilarity).

Our algorithm produces visually good results (as can be seen on Fig. 94), and is very stable to noise and rotation. We should also point out the algorithm of [BC] (ACK3A), which is very stable as long as the border deformation is not too important, and produces also visually good results, with few spurious branches.

	Rotation		Noise		Visual quality	
	Haus.	Dub.	Haus.	Dub.	$\lvert A(X)\rvert$	$\mathcal{R}_{\mathcal{B}_X^{A(X)}}$
[PK98]	16,16	4,41	18,17	6,18	448,90	0,64
[PTHS06]	12,72	2,95	19,32	6,29	501,60	0,61
[BC] (ACK3A)	13,62	3,45	20,05	7,27	329,25	0,73
[BC] (CK3A)	14,48	4,02	17,43	5,95	958,25	0,61
[BC] (CK3B)	14,79	4,08	17,82	6,09	999,40	0,60
[BC] (CK3)	11,37	2,94	11,97	3,35	2677,65	0,31
DT3-1	12,06	3,38	15,90	4,41	306,80	0,77

Table 7: **Results of tests on 3d curvilinear thinning algorithms -** Results of the tests performed on 3d shapes. Each shape X was thinned using one of the algorithms A presented one the first column of the table. Columns 2 and 3 present the average Dubuisson and Hausdorff distance between $R_\theta(A(X))$ and $A(R_\theta(X))$, with θ varying from 0 to 90 degrees. Columns 4 and 5 present the average Dubuisson and Hausdorff distance between $A(X)$ and $A(E(X,n))$, with n varying from 0 to 10 percent of the shape's volume. Column 6 present gives the average size of the output skeleton, and column 7 gives the normalized missed reconstructed border of $A(X)$ in X.

6.5 CONCLUSION

We showed in this chapter how cubical complexes could be used for performing homotopic thinning. We proposed a parallel thinning algorithm, and showed how to use the medial axes (of the DT framework) in order to propose, in the cubical

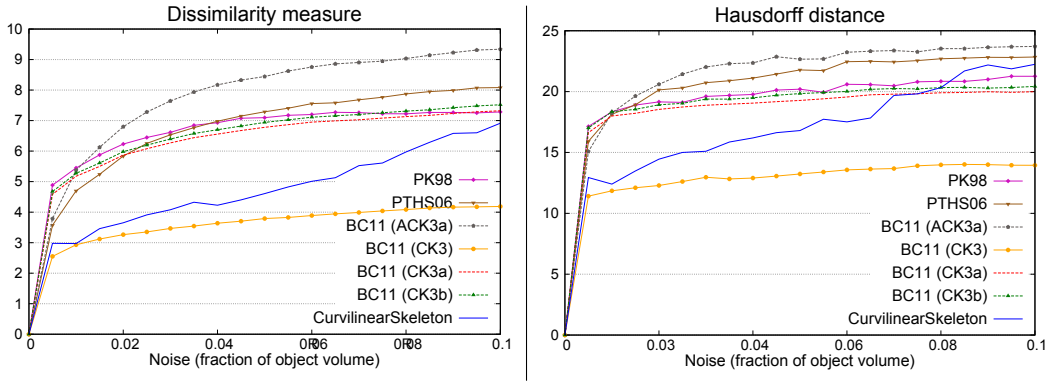


Figure 90: **Stability of 3d curvilinear thinning algorithms to noise -** Dissimilarity measure (on the left) and Hausdorff distance (on the right) between $A(E(X,n))$ and $E(A(X),n)$ on 3d shapes of our database (shape X), using various 3d curvilinear thinning algorithms A.

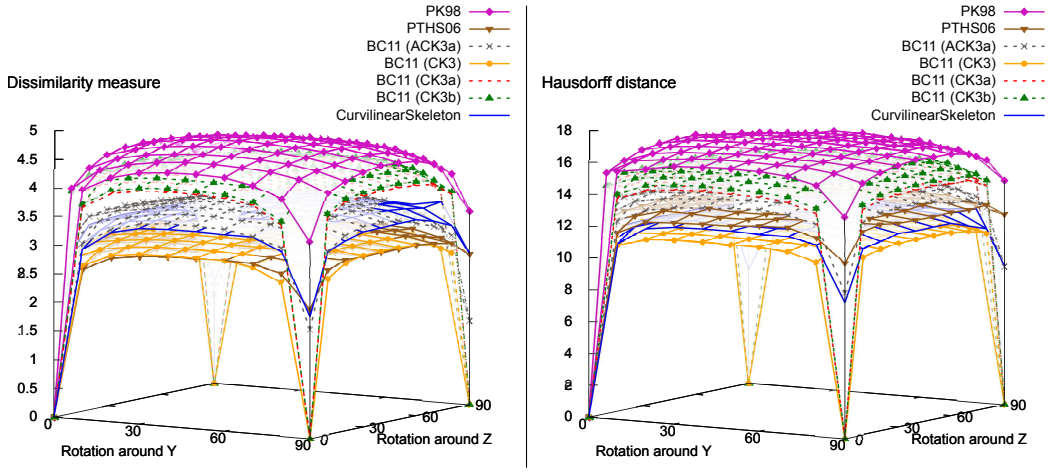


Figure 91: **Stability of 3d curvilinear thinning algorithms to rotation -** Dissimilarity measure (on the left) and Hausdorff distance (on the right between $A(R_\theta(X))$ and $R_\theta(A(X))$ on 3d shapes of our database (shape X), using various 3d curvilinear thinning algorithms A.

complex framework, skeletons yielding good reconstruction properties (containing visual information of the original shape).

We then proposed three parameter-free thinning algorithms for obtaining, in the cubical complex framework, skeletons which preserve the visual aspect of the input shape. Thanks to a general methodology allowing to embed a cubical complex in the voxel space, we were able to transpose our algorithms to the DT framework. Finally, we compared our algorithms to other thinning algorithms, and concluded that, in general, our algorithms provide the best stability/visual quality ratio from all the other skeletons.

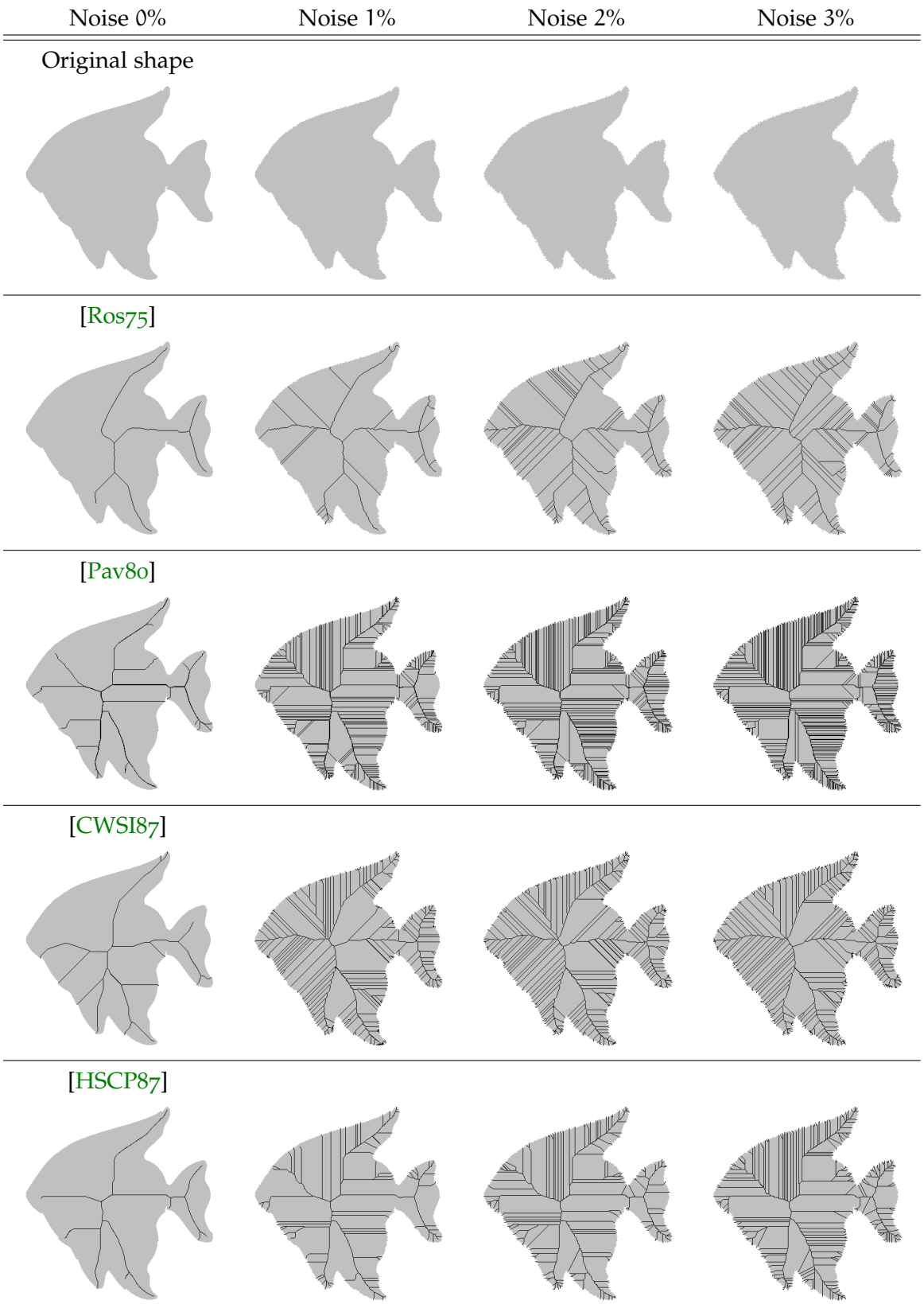


Figure 92: **Example of the behaviour of various thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input.

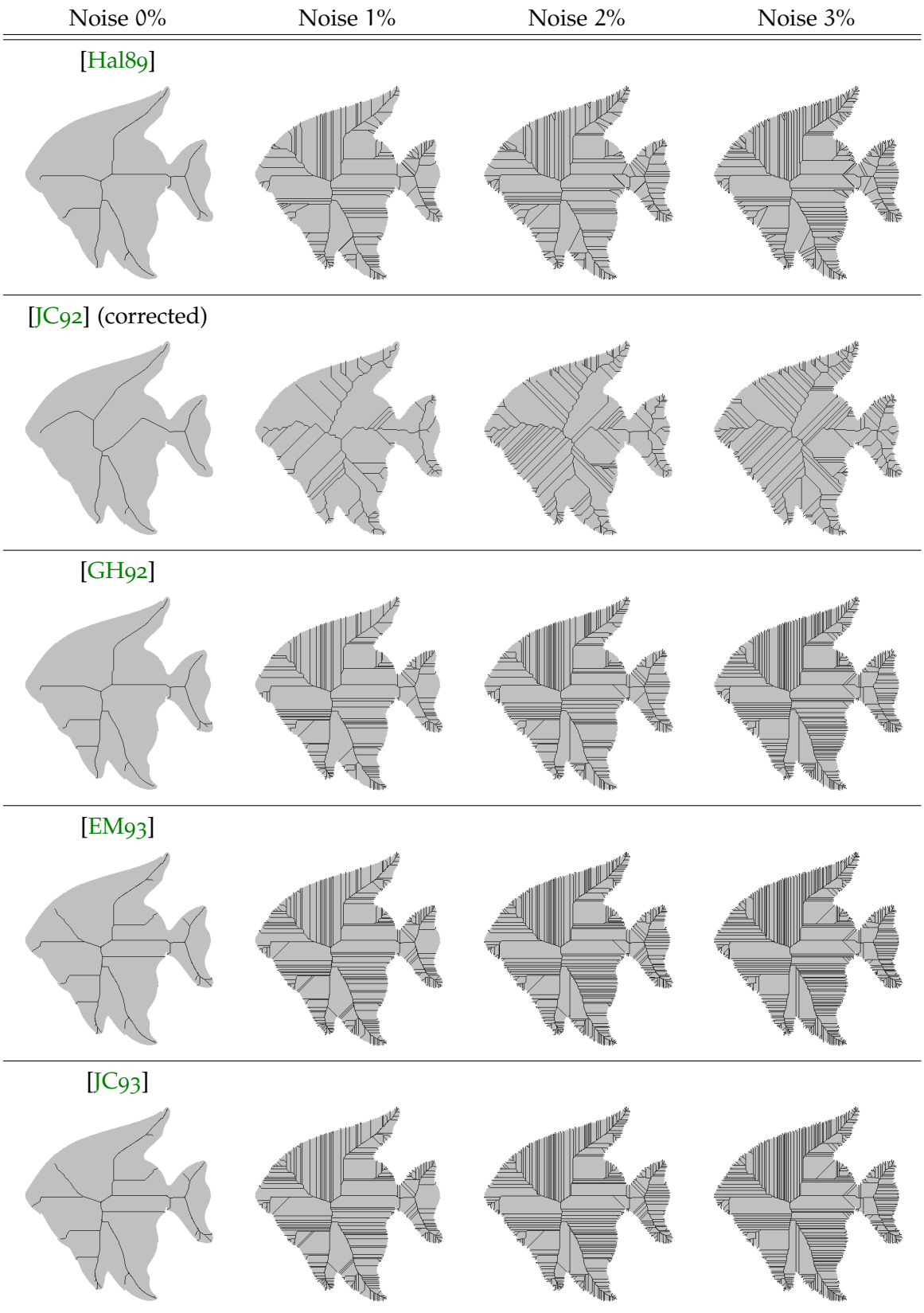


Figure 92: **Example of the behaviour of various thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input.

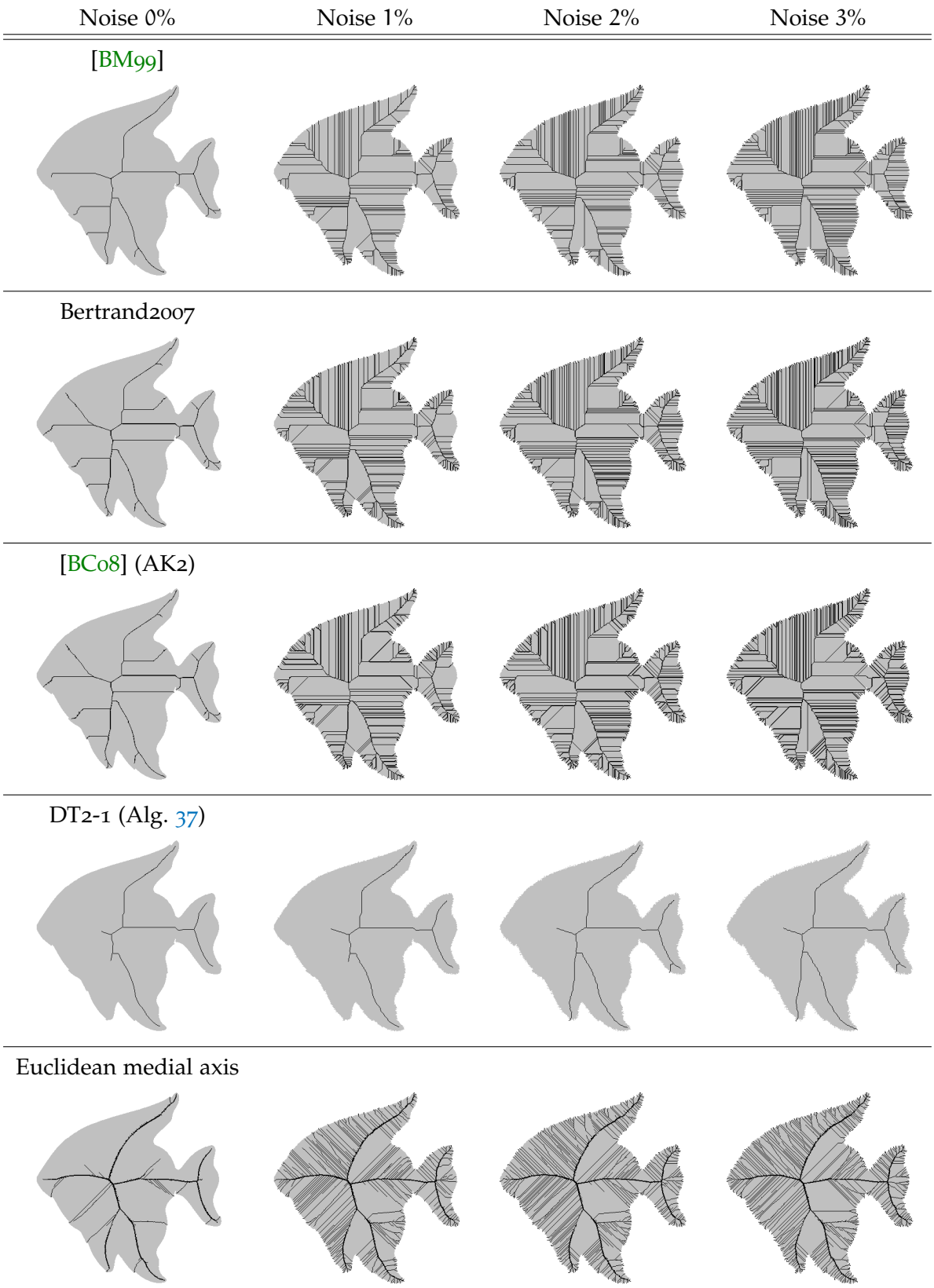


Figure 92: **Example of the behaviour of various thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input. Algorithm "Euclidean medial axis" is the homotopic thinning guided by the Euclidean distance map, and retaining all elements of the Euclidean medial axis.

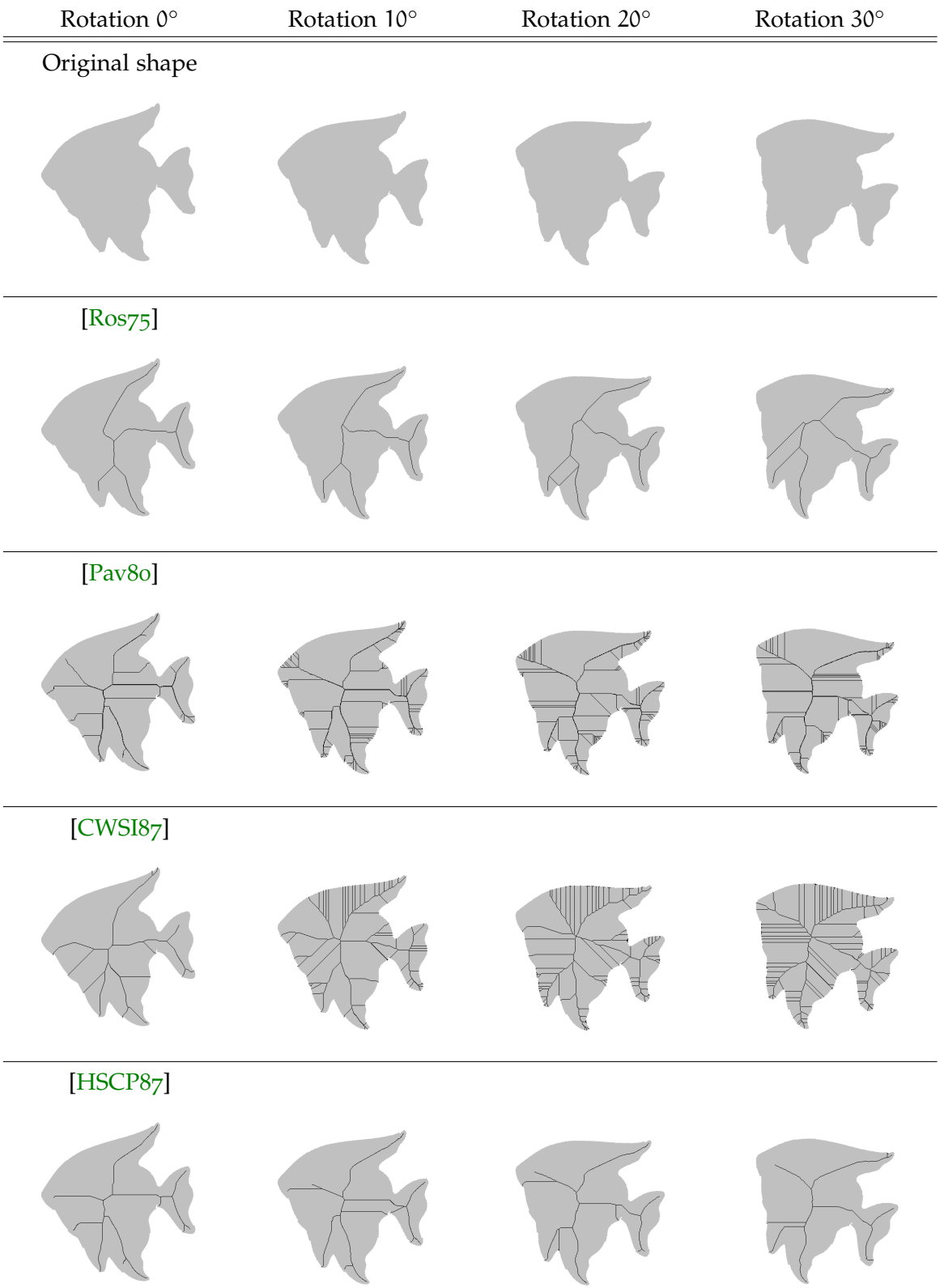


Figure 93: **Example of the behaviour of various thinning algorithms to rotation -** From left to right, the input is rotated with a greater angle.

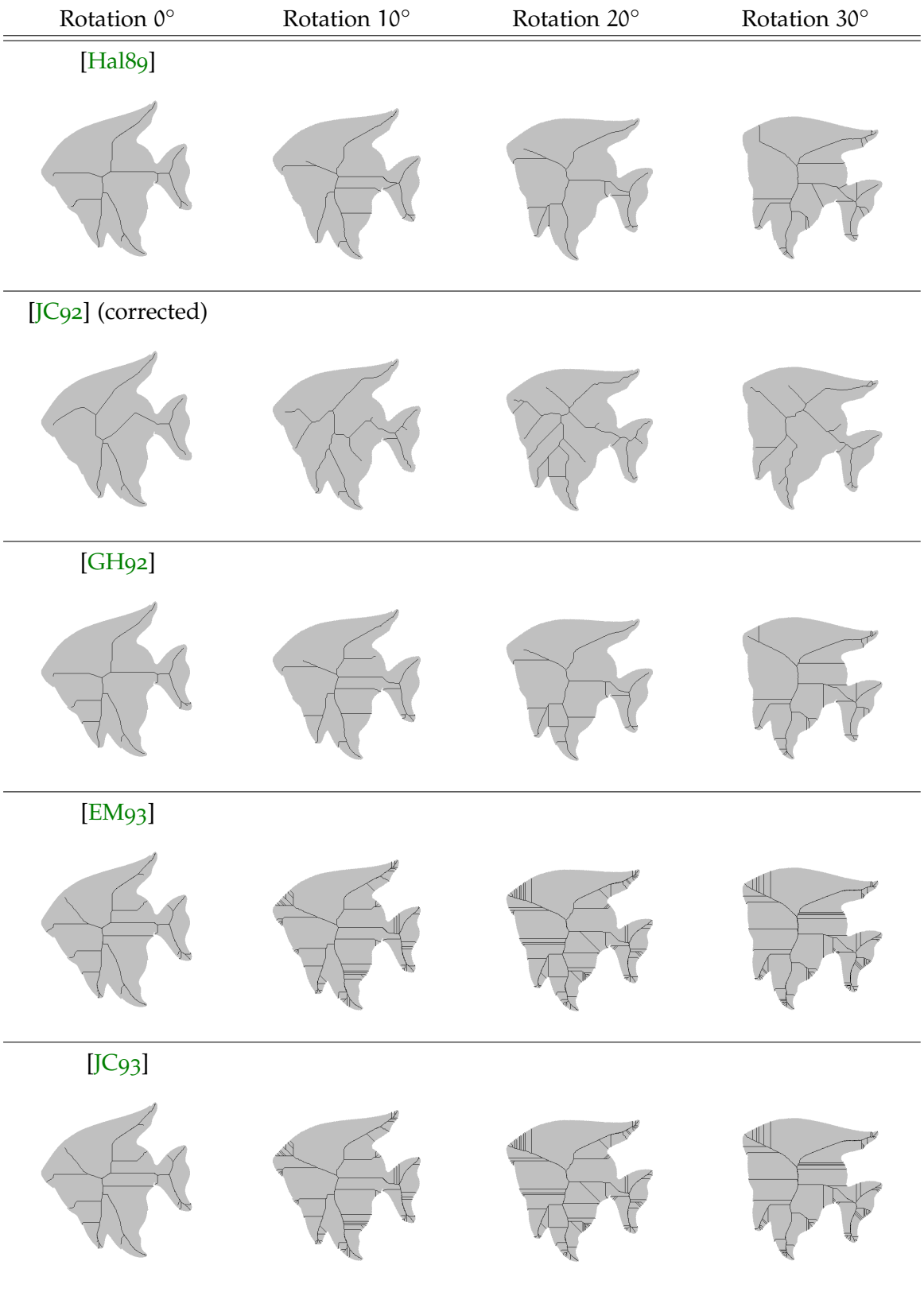


Figure 93: **Example of the behaviour of various thinning algorithms to rotation -** From left to right, the input is rotated with a greater angle.

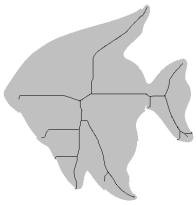
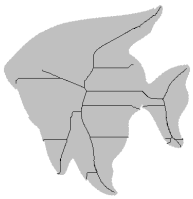
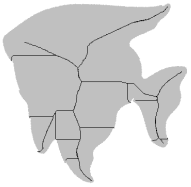
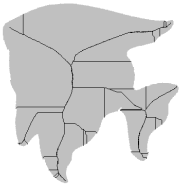
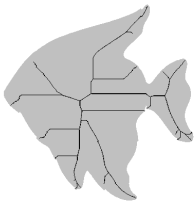
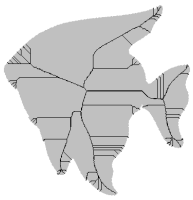
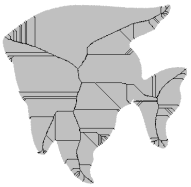
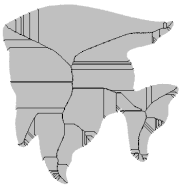
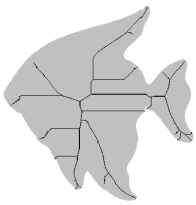
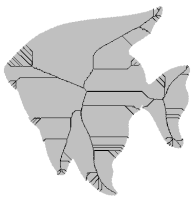
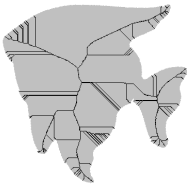
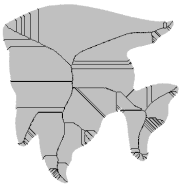
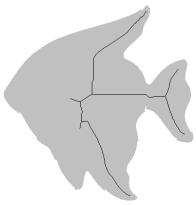
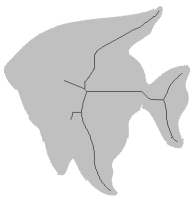
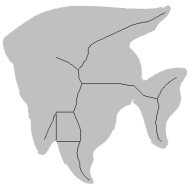
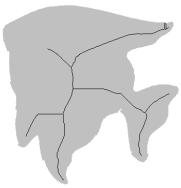
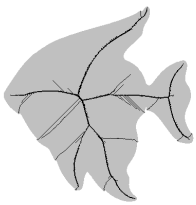
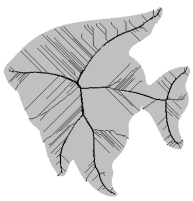
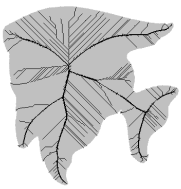
Rotation 0°	Rotation 10°	Rotation 20°	Rotation 30°
[BM99]			
			
Bertrand2007 (symmetric)			
			
[BC08] (AK2)			
			
DT2-1 (Alg. 37)			
			
Euclidean medial axis			
			

Figure 93: **Example of the behaviour of various thinning algorithms to rotation -** From left to right, the input is rotated with a greater angle. Algorithm "Euclidean medial axis" is the homotopic thinning guided by the Euclidean distance map, and retaining all elements of the Euclidean medial axis.

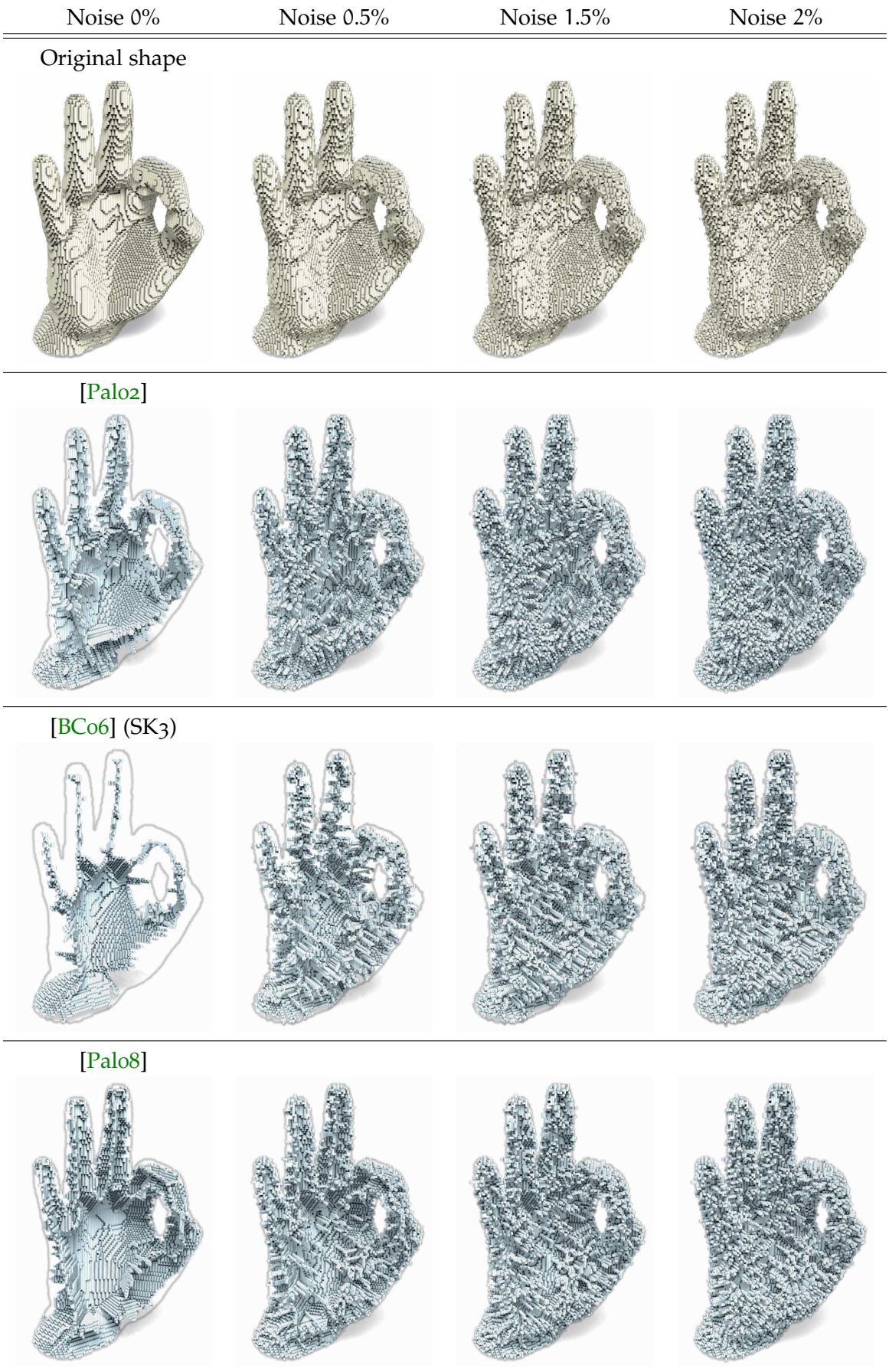


Figure 94: **Example of the behaviour of various 3d thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input.

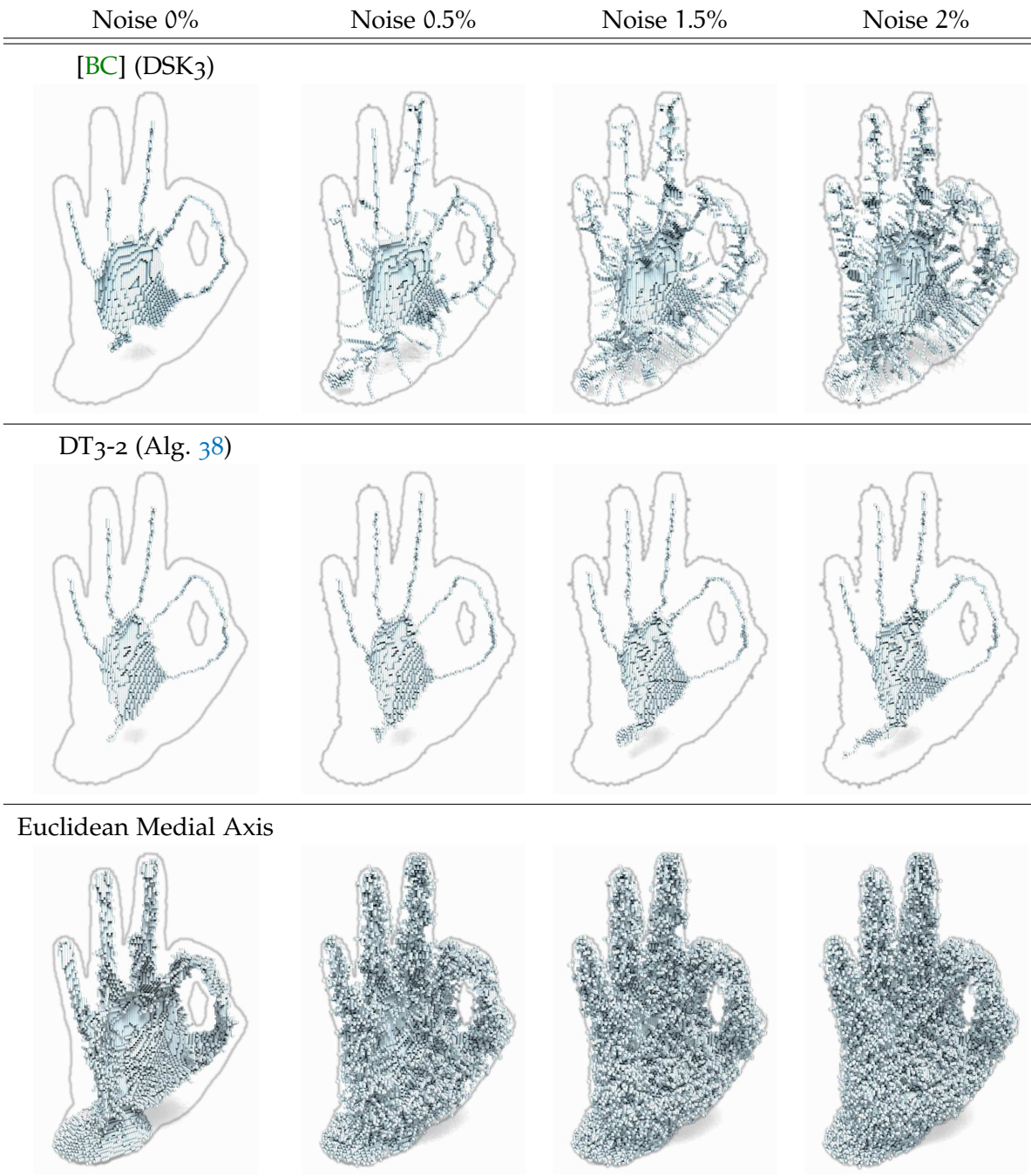


Figure 94: **Example of the behaviour of various 3d thinning algorithms to noise addition** - From left to right, more noise is added to the border of the input. Algorithm "Euclidean medial axis" is the homotopic thinning guided by the Euclidean distance map, and retaining all elements of the Euclidean medial axis.

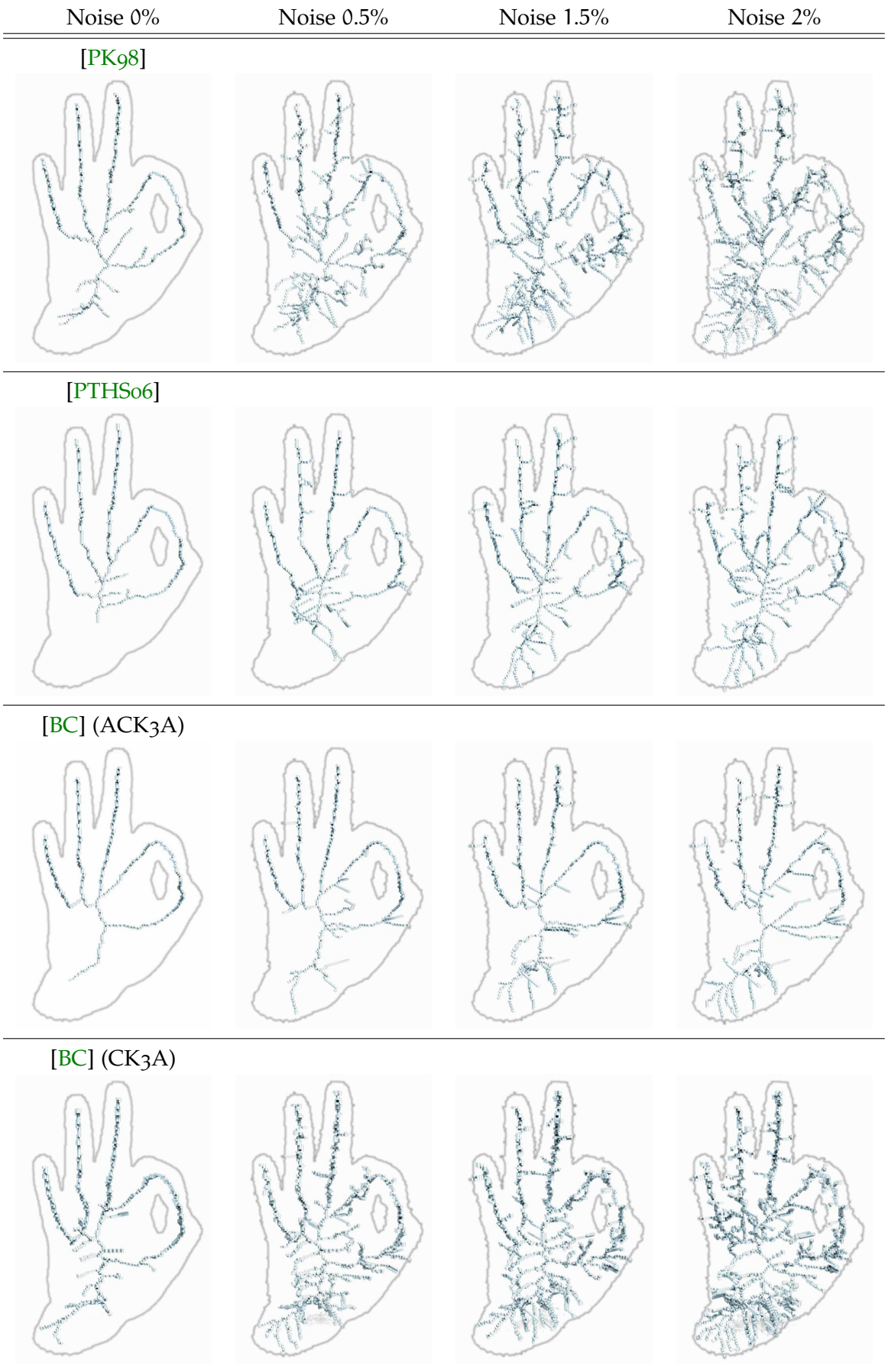


Figure 94: **Example of the behaviour of various 3d curvilinear thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input.

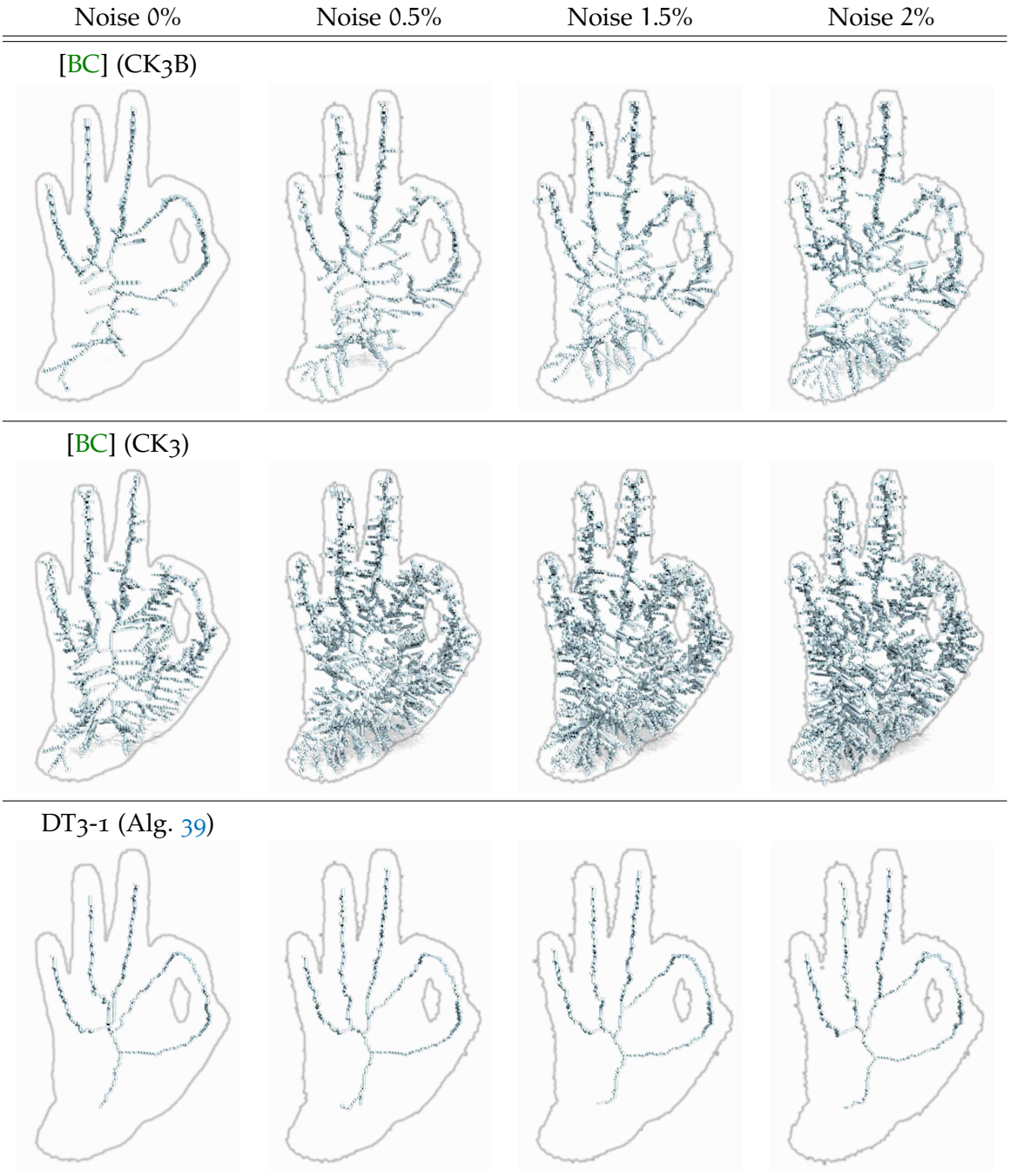


Figure 94: **Example of the behaviour of various 3d curvilinear thinning algorithms to noise addition -** From left to right, more noise is added to the border of the input.

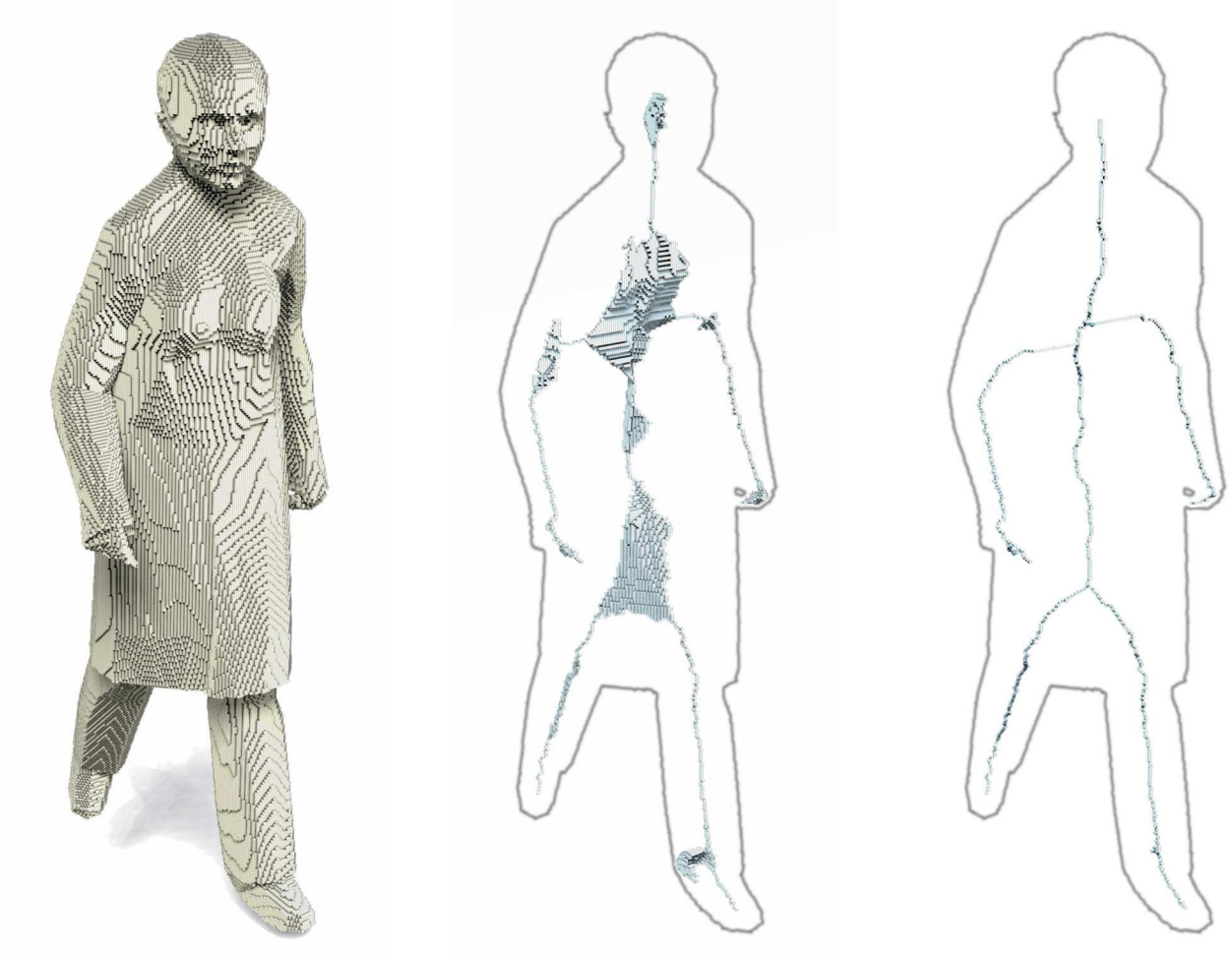


Figure 95: **Walking girl -** Homotopic thinning of a human silhouette. **On the left**, the original image. **In the middle**, the result of DT3-2 algorithm (with the border of the original image superimposed). **On the right**, the result of DT3-1 algorithm (with the border of the original image superimposed).

a) b) c)

Figure 96: **Chair -** Homotopic thinning of a chair. **a)**, the original image (already presented before). **b)**, the result of DT3-2 algorithm (with the border of the original image superimposed). **c)**, the result of DT3-1 algorithm (with the border of the original image superimposed). The image continues on the next page.

d) e)

Figure 96: **Chair -** Homotopic thinning of a chair (continuation of previous page). **d)**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **e)**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the two skeletons were obtained using the decomposition algorithm presented in Sec. 7.1.2.

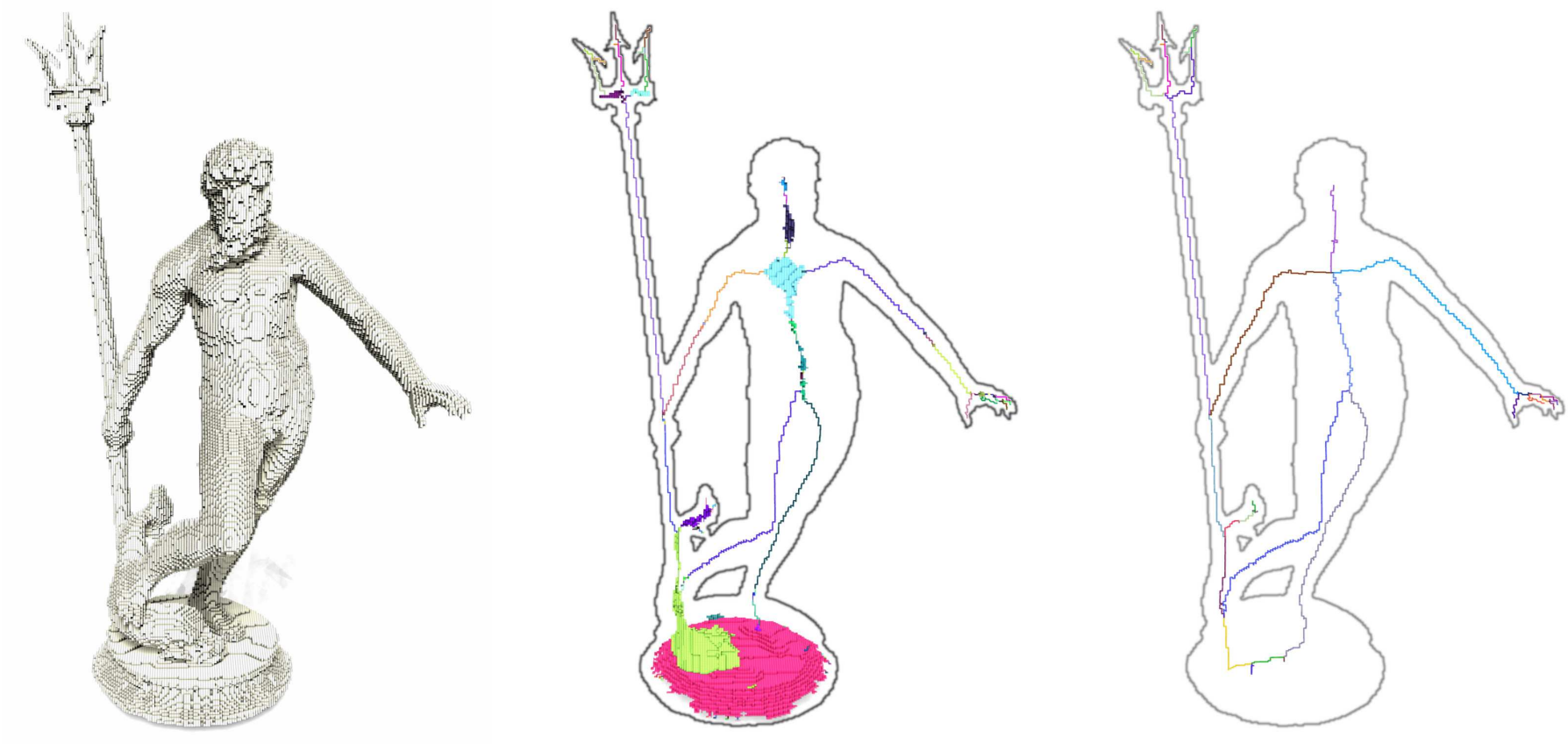


Figure 97: **Neptune -** Homotopic thinning of a statue representing Neptune. **On the left**, the original image (already presented before). **In the middle**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **On the right**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the skeleton were obtained using the decomposition algorithm presented in Sec. 7.1.2.

Figure 98: **Fertility -** Homotopic thinning of the Fertility statue. **On the left**, the original image (already presented before). **In the middle**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **On the right**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the skeleton were obtained using the decomposition algorithm presented in Sec. 7.1.2.

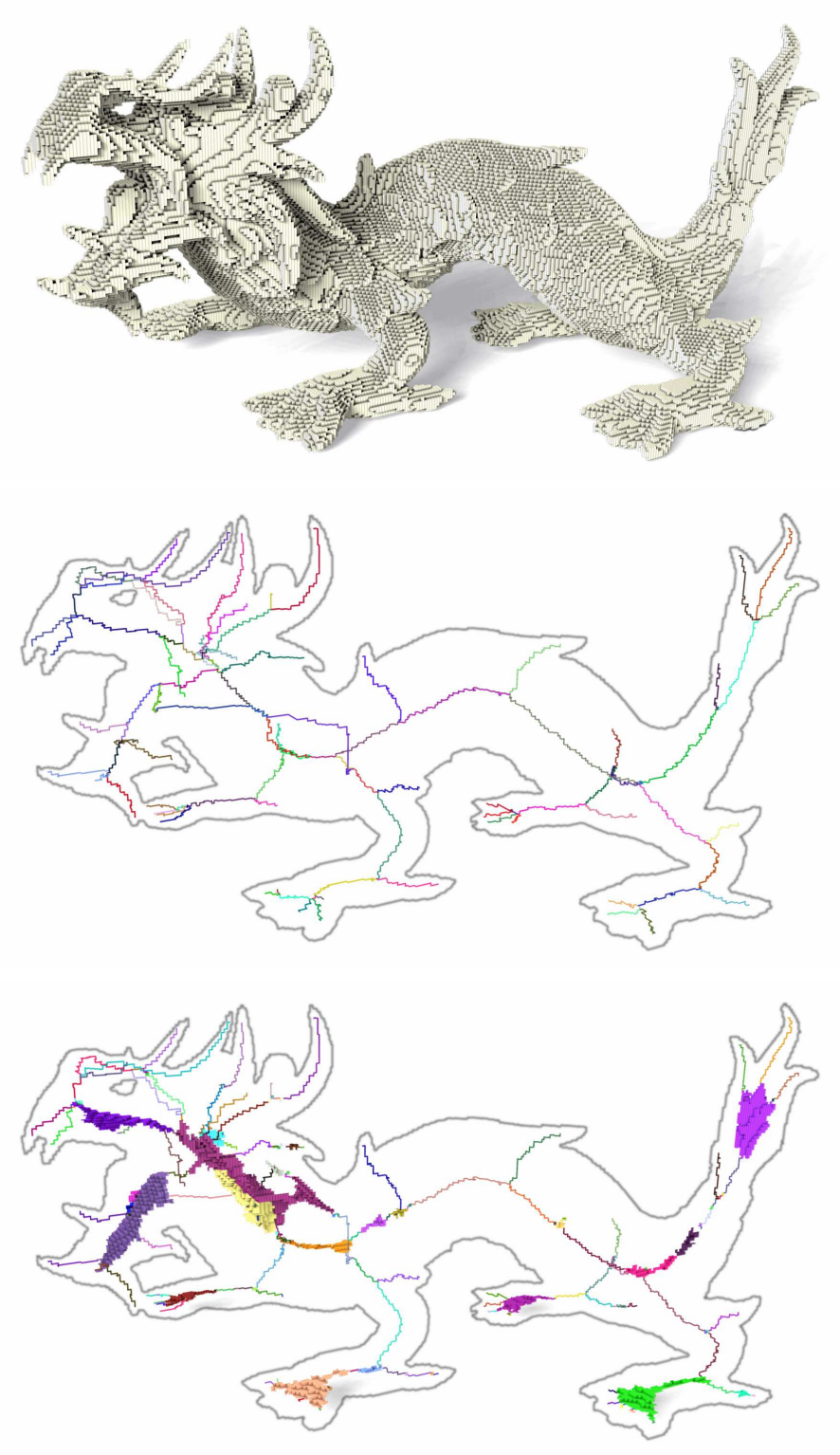
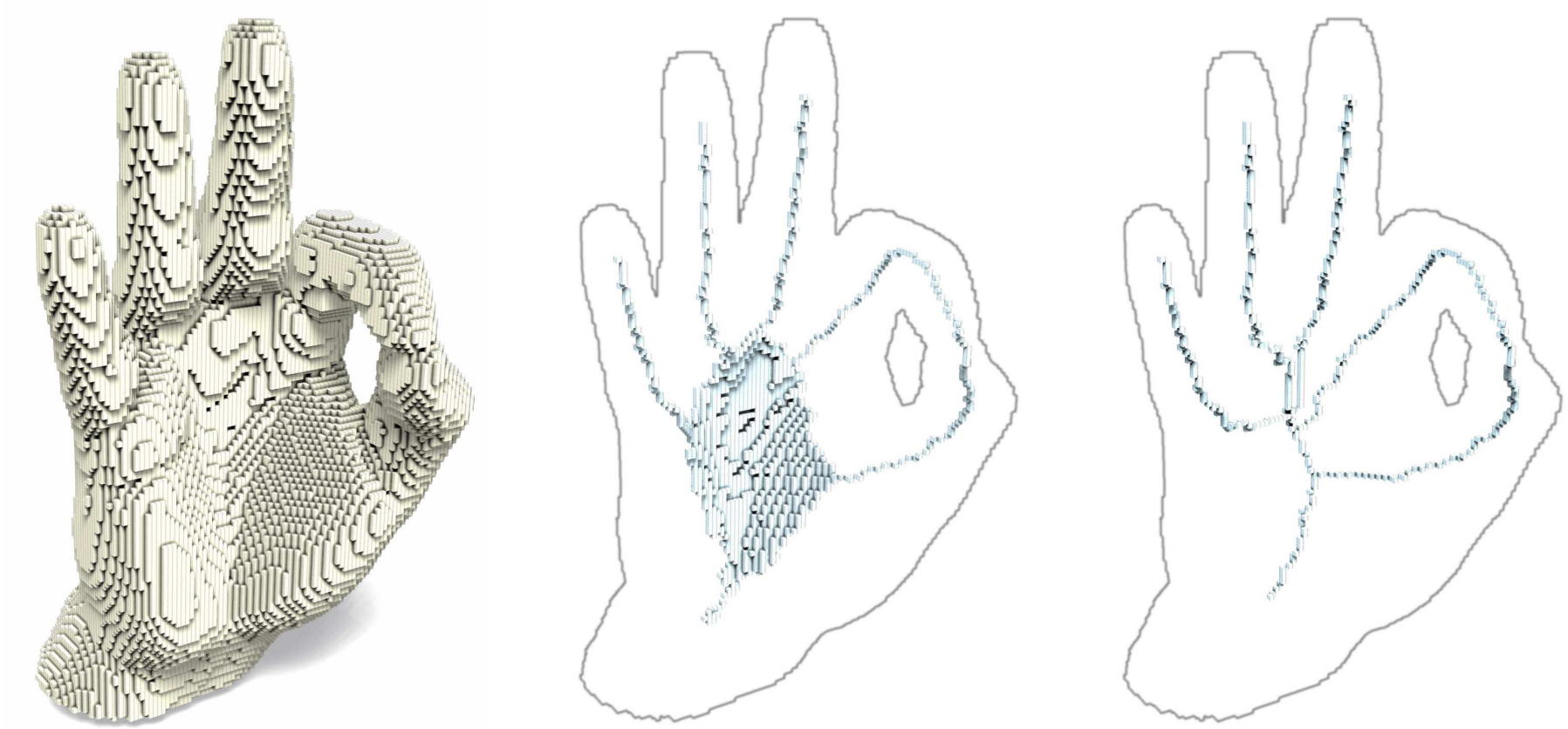
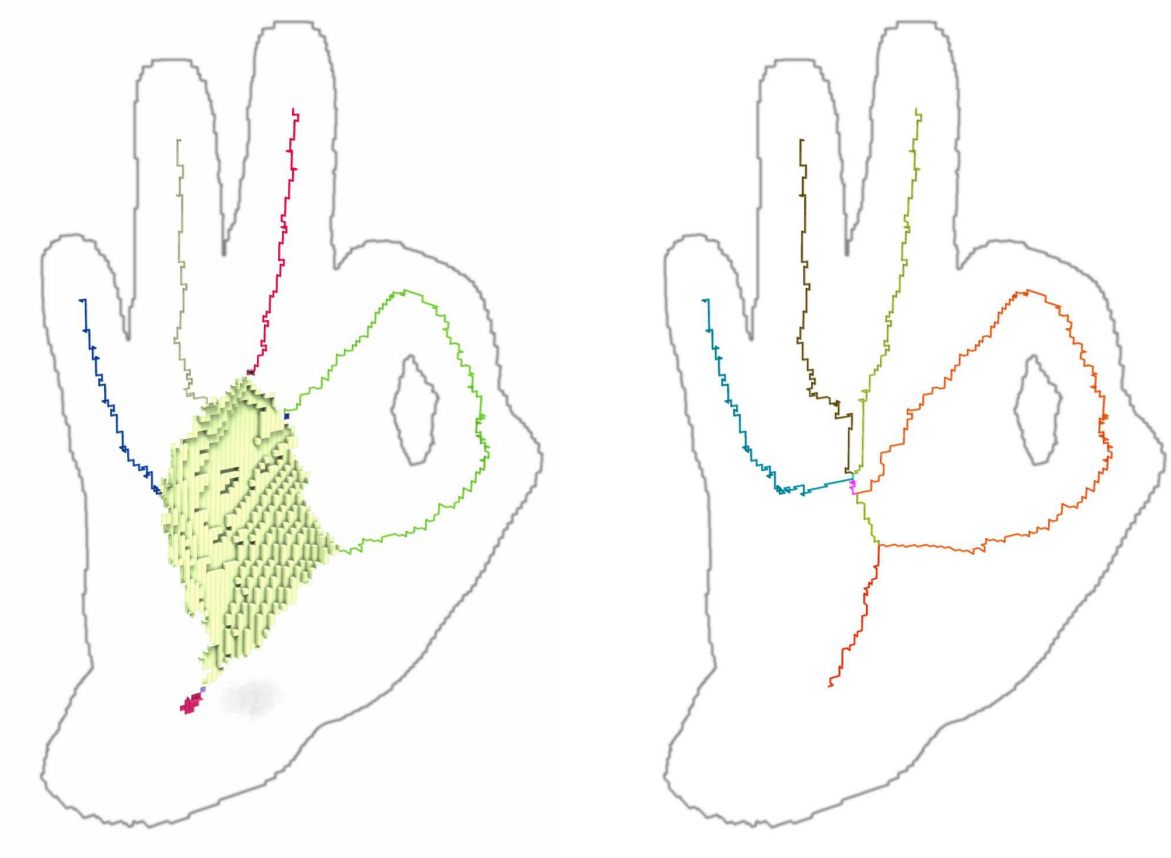


Figure 99: **Chinese dragon -** Homotopic thinning of the Chinese dragon statue. **On the left**, the original image. **In the middle**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **On the right**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the skeleton were obtained using the decomposition algorithm presented in Sec. 7.1.2.



a) b) c)

Figure 100: **Hand -** Homotopic thinning of a hand. **a)**, the original image (already presented before). **b)**, the result of DT3-2 algorithm (with the border of the original image superimposed). **c)**, the result of DT3-1 algorithm (with the border of the original image superimposed). The image continues on the next page.



d) e)

Figure 101: **Hand -** Homotopic thinning of a hand (continuation of previous page). **d)**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **e)**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the skeleton were obtained using the decomposition algorithm presented in Sec. 7.1.2.

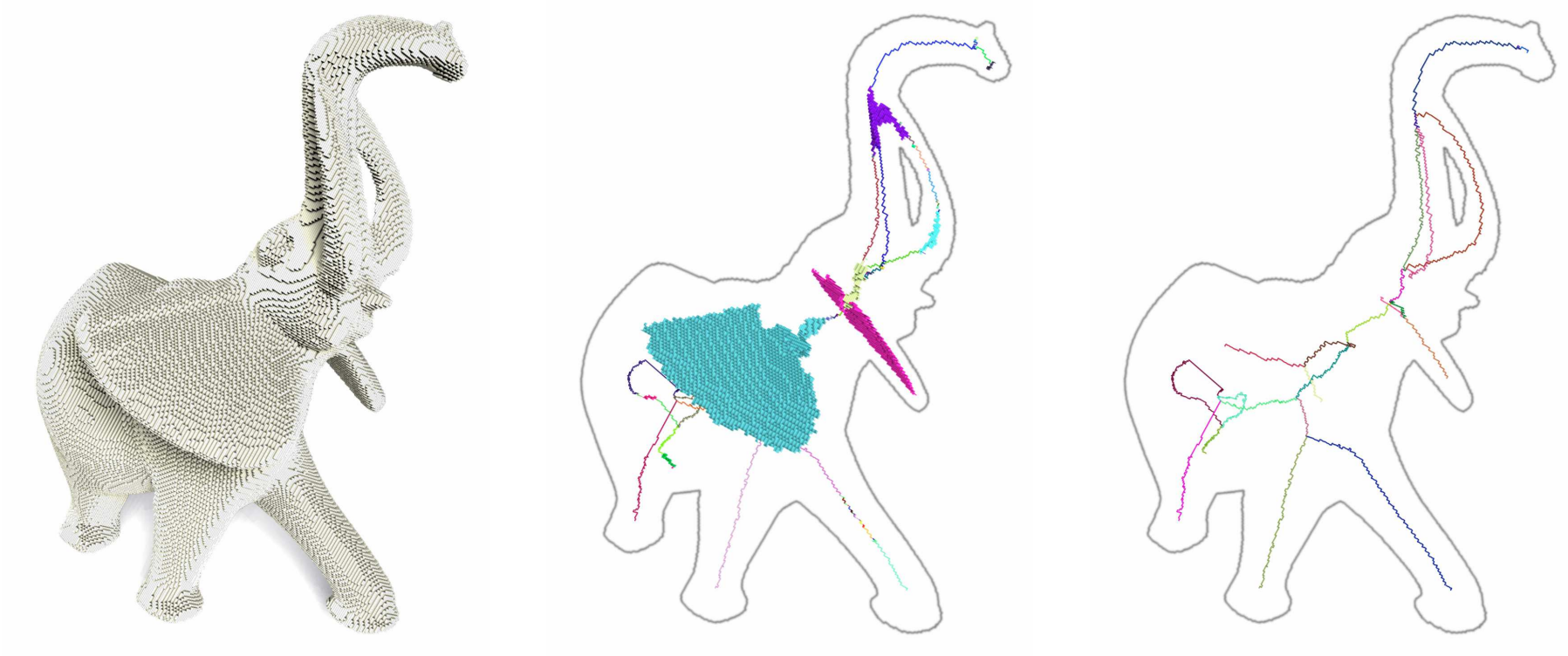


Figure 102: **Elephant -** Homotopic thinning of a statue representing an elephant. **On the left**, the original image (already presented before). **In the middle**, the result of SurfaceSkeleton algorithm (with the border of the original image superimposed). **On the right**, the result of CurvilinearSKeleton algorithm (with the border of the original image superimposed). The colours of the skeleton were obtained using the decomposition algorithm presented in Sec. 7.1.2.

7

DECOMPOSITION OF THE SKELETON INTO BASIC COMPONENTS

In this chapter, we explain how a skeleton, in the cubical complex framework, can be decomposed into basic parts, and at the same time, we show that the choice of the cubical complex framework was good in order to obtain sound decomposition of a skeleton.

In Sec. 7.1, we explain how simple can the decomposition of a skeleton be done in this framework. Then, in Sec. 7.2, we study more closely how to characterize intersection between components, and we give a nice property that edges labelled as intersections between surfaces have in 3d. Finally, in Sec. 7.3, we show that the decomposition into components can have disappointing results, and we propose to use the thinning operation in order to enhance them.

Contents

In 3d, a skeleton should be a bidimensional object, made of curves, surfaces, and junctions (junction between curves, between surfaces, between curves and surfaces, ...). Thanks to the use of cubical complexes and of our thinning algorithm, we have the guarantee that the results of algorithms 34 and 35 are bidimensional, and the results of algorithm 33 are one-dimensional.

7.1 CHARACTERIZING SIMPLE COMPONENTS IN THE CUBICAL COMPLEX FRAMEWORK

7.1.1 SOME DEFINITIONS

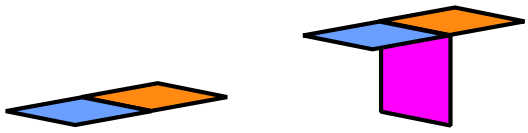
Regular neighbours is an important notion for understanding simple components in the cubical complex framework:

Definition 7.1.1 *Given* $X \preceq \mathbb{F}^n$ *and two* k*-faces* f *and* $g \in X^+$*. The two faces* f *and* g *are* regular neighbours in X *if there exists a* $(k-1)$*-face* e *such that* $(\hat{f} \cap \hat{g}) = \hat{e}$ *and* $\check{e}^* \cap X = \{f, g\}$.

A face $e \in X$ *is* a non-regular face of X *if there exists* f *and* $g \in X^+$ *such that* f *and* g *are not regular neighbours in* X *and* $(\hat{f} \cap \hat{g}) = \hat{e}$.

In the three-dimensional cubical complex framework, two squares are regular neighbours in a complex X if they are facets of X and if they share an edge that is contained in only two squares (see Fig. 103); two edges are regular neighbours in X if they are facets of X and if they share a vertex that is contained in only two edges.

A non-regular face of dimension k is either contained in three or more facets of dimension $(k+1)$, either contained in two or more facets of various dimension, or either contained in two or more faces of dimension $(k+2)$ or more. For example, in Fig. 103b, the edge shared by the three squares is a non-regular face. Still in the three-dimensional cubical complex framework, the vertex located at the intersection between a square and an edge, both being facets of a complex, is a non-regular face. A vertex located at the intersection of two squares which do not share an edge is also a non-regular face.



a) b)

Figure 103: **Regular neighbours :** in **a)**, the two squares are regular neighbours. However, in **b)**, non of the three squares is a regular neighbour of another square.

We now define *connected components of a set of faces*:

Definition 7.1.2 *Given a set of faces* X*, a set* $S \subseteq X$ *is a* connected component of X *if*

- *for all* $f, g \in S^+$*, there exists a sequence* $(h_1, ..., h_i)$ *of faces of* S *such that* $f = h_1$, $g = h_i$, *and for each* $j \in [1; i-1]$, $(h_j \subseteq h_{j+1})$ *or* $(h_{j+1} \subseteq h_j)$.
- *The set* S *is maximal in the sense of the inclusion: there exists no sets* $S' \subseteq X$ *such that* S' *possesses the previous property and strictly contains* S.

This definition is similar to the usual definition of connected components. We finally define *simple components of a complex*:

Definition 7.1.3 *Given* $X \preceq \mathbb{F}^n$, *let* $I = \{f \in X | f$ *is a non-regular face of* $X\}$. *Let* S *be a connected component of* $(X \backslash I^-)$: *the subcomplex* S^- *is a* simple component of X.

In the previous definition, the set I^- acts as a barrier separating simple components of X: two faces must share a face outside of this barrier in order to belong to the same simple component. Two facets of X of different dimension either share a non-regular face, or do not share anything: therefore, a simple component is necessarily a pure complex. A *simple* k*-component* is a simple component of dimension k. Finally, we have the following property: two faces which are regular neighbours in X belong to the same simple component.

In the three-dimensional cubical complex framework, we call *surface* of $X \preceq \mathbb{F}^n$ a simple 2-component of X, and *curve* of X is a simple 1-component of X (this key word holds also in the two-dimensional cubical complex framework).

The simple components of a complex is what we commonly called, previously, "basic components", "elements" or "parts".

7.1.2 ALGORITHM FOR DECOMPOSING A COMPLEX INTO SIMPLE COMPONENTS

We give in the following two algorithms: algorithm 40 allows to find the simple component which contains a given facet of a complex, while algorithm 41 labels each facets of a complex depending on which simple component it belongs to. Algorithm 40 "assembles" regular neighbours together in order to form simple components (it is similar to the usual algorithm for labelling connected components with a list). Algorithm 41 gives as output the set of all simple components (represented as a set of integers) to which belongs the faces of X: faces of X which belong to more than one simple component are located at the junction between simple components. Both algorithms take as input a set W, which represents "forbidden faces" that two faces should not share in order to be considered as part of a same simple component: it will be useful later in this work (however, for the moment, it can be ignored).

Algorithm 40: GetAllFacesOfComponent(X, W, f)

Data: A cubical complex $X \preceq \mathbb{F}^n$, a cubical complex $W \preceq \mathbb{F}^n$, and a face $f \in X^+$
Result: A cubical complex $S \preceq \mathbb{F}^n$ such that $f \in S$ and S is a simple component of X

```
S = ∅; T = {f};
while there exists g ∈ T do
    S = S ∪ ĝ;
    T = T\{g};
    foreach h ∈ X+ such that g and h are regular neighbours in X and (g ∩ h) ∉ W do
        T = T ∪ {h};
    end
end
return S;
```

Algorithm 41: LabelComponent(X, W)

Data: A cubical complex $X \preceq \mathbb{F}^n$ and a cubical complex $W \preceq \mathbb{F}^n$
Result: A map $L : X \to \mathcal{P}(\mathbb{Z})$ such that, for all $f, g \in X$, $L(f) \cap L(g) \neq \emptyset$ iff f and g belong to a same simple component of X

```
foreach f ∈ X do
    L(f) = ∅;
end
cpt = 1;
while there exists f ∈ X⁺ such that L(f) = ∅ do
    S = GetAllFacesOfComponent(X, ∅, f);
    foreach g ∈ S do
        L(g) = L(g) ∪ {cpt};
    end
    cpt = cpt + 1;
end
return L;
```

Examples of skeleton decomposition can be found from Fig. 96 p. 173, up to Fig. 102 p. 180: in these skeletons, the various simple components have different colours.

7.2 INTERSECTIONS BETWEEN SIMPLE COMPONENTS

7.2.1 INTERSECTIONS IN THE n-DIMENSIONAL CUBICAL COMPLEX FRAMEWORK

We characterize in the following *intersection faces and border faces of a simple component*:

Definition 7.2.1 *Let $X \preceq \mathbb{F}^n$. The* border of X, *denoted by Border(X), is the closure of the set $\{f \in X \mid f$ is free in $X\}$. A face belonging to the border of X is called* a border face.

A face $f \in X$ is an intersection face of X *if $(\check{f} \cap X)^-$ contains more than one simple component. We denote by $X^\cap$ the set of all intersection faces of X.*

Note that an intersection face is necessarily a non-regular face. The border of an n-dimensional complex is at most an $(n-1)$-complex, and the set of intersection faces of a complex is also at most an $(n-1)$-complex. We now define the intersections of a complex. It would be tempting to say that an intersection of X is a simple component of $X^\cap$, however, in some configurations, this might lead to unwanted results, as shown on Fig 104 and explained later.

We define pinch faces:

Definition 7.2.2 *Let $X \preceq \mathbb{F}^n$ and let f, g be two k-faces belonging to $X^\cap$, such that $e = f \cap g$ is a $(k-1)$-face. The face e is* smooth for X *if, for every face $h \in (\check{f} \cap X^+)$, there exists a face $i \in (\check{g} \cap X^+)$ such that h and i belong to the same simple component of $\check{e} \cap X^-$.*

The face e is a pinch face for X *if it not smooth for X.*

We now define intersections of a complex:

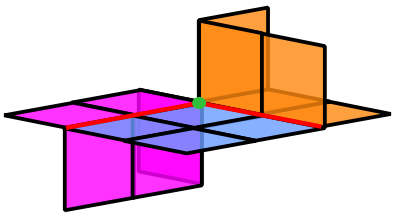


Figure 104: **An example of a pinch face -** The vertex in green is a pinch face, the edges in red are intersection edges.

Definition 7.2.3 *Let* $X \preceq \mathbb{F}^n$ *and let* $P = \{e \in X | e$ *is a pinch face for* $X\}$*. Let* C *be a connected component of* $(X^{\cap} \backslash P^{-})$*: the subcomplex* C^{-} *is an* intersection between simple components of X.

Thanks to pinch faces, we have the following property on intersections:

Proposition 7.2.4 *Let* $X \preceq \mathbb{F}^n$ *and let* C *be an intersection between simple components of* X*. For every face* $f, g \in C^{+}$*, if* f *belongs to a simple component* S *of* X*, then* g *belongs also to* S*.*

Proof This property is a natural consequence of the use of pinch faces in Def. 7.2.3. The faces f and g being in the same intersection, there exists a sequence $(e_1, ..., e_i)$ of facets of C such that $e_1 = f$, $e_i = g$ and, for each $j \in [1, i-1]$, e_j and e_{j+1} are regular neighbours in $X^{\cap}$: for each $j \in [1, i-1]$, $h_j = (e_j \cap e_{j+1})$ is such that $\dim(h_j) = \dim(e_j) - 1$ and h_j is not a pinch face. Therefore, for each $j \in [1, i-1]$ and for all facet of X in $\check{e}_j$ belonging to a simple component S, there exists a facet of X in $\check{e_{j+1}}$ belonging to S. In conclusion, for all facet of X in $\check{f}$ belonging to a simple component S, there exists a facet of X in $\check{g}$ belonging to S.

In the three-dimensional framework, given a cubical complex X, some simple components $S_1, ..., S_i$ of X (being, for example, surfaces), and an intersection C (being, for example, a curve) such that one facet of C is an intersection between $S_1, ..., S_k$, property 7.2.4 states that all faces of C are intersections between $S_1, ..., S_k$. On Fig. 104, the intersection edges (in red) form a curve : the pinch face (in green) marks the delimitation between the intersection of the purple and blue surfaces, and the intersection of the orange and blue face.

7.2.2 INTERSECTIONS IN THE 2D AND 3D CUBICAL COMPLEXES FRAMEWORKS

In the following, we characterize more precisely all faces of an $(n-1)$-complex in an n-dimensional cubical complex framework (as would be a skeleton computed with the thinning methodologies previously proposed), with $n = 2$ or $n = 3$.

In the two-dimensional cubical complex framework, the decomposition of a 1-dimensional complex X results in a set of curves and isolated vertices. Any edge of X is part of a curve of X, and the characterization of intersection faces is straightforward:

Proposition 7.2.5 *Let* $X \preceq \mathbb{F}^2$ *be such that* $\dim(X) = 1$*, the* intersection faces of X *are the vertices of* X *which are contained in three edges of* X *or more.*

Label of the 0-face $f \in X$	Value of $\lvert \check{f}^* \cap X \rvert$
Isolated	$= 0$
Border of a curve	$= 1$
Interior of a curve	$= 2$
Curves intersection	> 2

Table 8: Labels of a 0-face f of a one-dimensional complex $X \preceq \mathbb{F}^2$.

Label of the 1-face $f \in X$	Value of $\lvert \check{f}^* \cap X \rvert$
Part of a curve	$= 0$
Border of a surface	$= 1$
Interior of a surface	$= 2$
Surfaces intersection	> 2

Table 9: Labels of a 1-face f of a bi-dimensional complex $X \preceq \mathbb{F}^3$.

Table 8 gives an overview of all possible configurations that a vertex of a 1-complex can have in 2d.

In the three-dimensional cubical complex framework, the decomposition of a 2-dimensional complex X results in a set of surfaces, curves and isolated vertices. Any square of X is part of a surface of X, and the characterization of intersection edges is straightforward:

Proposition 7.2.6 *Let* $X \preceq \mathbb{F}^3$ *be such that* $\dim(X) = 2$*, an edge of* X *is an* intersection face of X *if it is contained in three squares of* X *or more.*

Classifying a vertex of a 2-complex still requires to study the decomposition into simple components of its star. Tables 9 and 10 give an overview of all possible configurations that an edge and a vertex of a 2-complex can have in 3d.

7.3 RESULTS AND ENHANCEMENTS OF THE DECOMPOSITION

In the cubical complexes framework, the decomposition of a skeleton have sound properties: a skeleton is a thin complex (typically, in 3d, a skeleton contains no volumes), and intersections between simple components have the same properties than in the continuous framework (intersections between surfaces is a set of curves or points, and intersections between curves is a point).

Label of the 0-face $f \in X$	Number of simple 2-components of $(\check{f} \cap X)^-$	Number of simple 1 components of $(\check{f} \cap X)^-$
Isolated	$= 0$	$= 0$
Part of a curve	$= 0$	$= 1$
Curves intersection	$= 0$	> 2
Part of a surface	$= 1$	$= 0$
Surface/Curve intersection	$= 1$	$= 1$
Surface/Curves intersection	$= 1$	$\geqslant 2$
Surfaces intersection	> 2	$= 0$
Surfaces/Curve intersection	> 2	$= 1$
Surfaces/Curves intersection	> 2	$\geqslant 2$

Table 10: Label of a 0-face f of a bidimensional complex $3 \preceq \mathbb{F}^X$.

7.3.1 ANALYSING THE RESULTS OF THE DECOMPOSITION

7.3.1.1 THE ORDER OF REMOVAL OF FACES HAS AN IMPACT ON THE DECOMPOSITION

Unfortunately, even though cubical complexes allow to obtain a good properties on the decomposition, the thinning process can lead to different decompositions based on how the free faces were removed.

In the DT framework, the Bing's house is a famous example of how removal order of simple points can have important consequences on the resulting skeleton. In the complex framework, the Bing's house phenomenon can also occur: homotopic thinning of a ball can either lead to a single vertex, or to a set of surfaces (called the Bing's house) containing no free faces. On Fig. 105, we present a 3-complex with free pairs of faces: the various orders of removal of the free pairs lead different decompositions of the resulting skeleton into simple 2-components.

7.3.1.2 AVOIDING THIS PHENOMENON: TOWARDS MINIMIZING THE NUMBER OF SIMPLE COMPONENTS IN THE SKELETON ?

The reader may wonder if it is possible to avoid this phenomenon by proposing an algorithm which would minimize the number of simple components obtained in the resulting skeleton. Proposing such an algorithm performing such task in an efficient way (which means avoiding to explore all the possible solutions in order to find the optimal one) would give the guarantee that the thinning of a three dimensional ball would lead to a single vertex or curve, thus avoiding the Bing's house configuration. Such problem has yet not been solved (as stated in the conclusion of [MF08]), and many related problems were proved to be NP-hard ([MF08], [Tan09]) or undecidable ([Mar60]); there is therefore little chance to find an efficient thinning algorithm which minimizes the number of simple components of the resulting skeleton.

Moreover, a skeleton with the minimum number of simple components is not, in the general case, giving an interesting result. Indeed, the minimization of the number of

simple components in the output skeleton must be balanced with the capacity of the skeleton to retain important visual information.

The apparition of small simple components in the skeleton, as depicted on Fig. 105, cannot be solved by adopting a global strategy of minimizing the number of simple components in the result during thinning. A local strategy, adopted during thinning, could exist in order to reduce this phenomenon, however, we did not find a solution in order to completely avoid these simple components. Indeed, this phenomenon mainly takes place around intersections between simple components, and we came across a strategy that seems to be producing interesting results. This strategy consists of performing a thinning of an object in order to locate the intersection faces, and then perform a second thinning of the same object which won't delete any face around the intersections previously located. Then, faces around the intersections are removed by testing locally all possible removal order and keeping the one which produces the less simple components in the result. This strategy seems to minimize the number of "small avoidable simple components" in the skeleton.

7.3.2 DIVISION OF SIMPLE COMPONENTS INTO SIMPLE SUB-COMPONENTS

On Fig. 108, we show an image depicting cracks inside concrete. Using our algorithm, we thin these cracks and decompose them into simple components. The results of the decomposition show that, in some cases in 3d, two different surfaces (visually different as they possess different orientation in space) may be joined by a "narrow bridge of squares" and be therefore identified as the same surface component. In order to avoid this phenomenon to happen, it is necessary to "destroy the bridge" before performing the decomposition into simple components of the skeleton.

We identified two different types of "bridges": real bridges, surrounded by the void (as shown on Fig. 106a), and connecting two surfaces, and bridges located near surface intersections (as shown on Fig. 107a). More thinning on the object would remove the first category of bridge (see Fig. 106c), while the second category of bridge would simply be shifted. Moreover, additional thinning removes more information from the skeleton, and we prefer avoid loosing information from the skeletons.

We propose with the following strategy in order to "ignore" the second category of bridges (see Alg. 42): once the skeleton S of the original object X has been computed, it is thinned again (the amount of thinning is decided by the user) in order to remove the first category of bridges and shift the second category (see Alg. 42, l. 2). The result of this second thinning is the skeleton S' see, (see, for example, Fig. 107d).

The skeleton S' is then decomposed into simple components (see Alg. 42, l. 4), using the set $S^{\cap}$ as the set of "forbidden faces" that should not join faces of a same simple component (the set W of Alg. 40): this way, the second group of bridges connecting the surfaces, which were shifted inside the surface, cannot be used any more to propagate the label (see, for example, Fig. 107e). The labels of the squares of S' are then directly applied on the squares of S. Some squares of S still miss a label (the squares on the bridges destroyed by the "over collapse"): the labels of the squares of S are propagated in parallel on the whole skeleton (see, for example, Fig. 107f), so that all squares of S possess a label (see Alg. 42, l. 10). The final result is a decomposition of the skeleton into *simple sub-components*, that is a partition of a simple component S into pure subcomplexes of S, having the same dimension than S.

Results of this method are shown on Fig. 106, 107, and 108.

Algorithm 42: $OverCollapseLabel(S, m)$

Data: A cubical complex $S \preceq \mathbb{F}^3$ such that $\dim S = 2$ (S is usually a skeleton), an integer m which represents the maximum width of "bridges" to ignore when labelling simple components

Result: A map $L' : S \to \mathcal{P}(\mathbb{Z})$ such that, for all $f, g \in S$, $(L'(f) \cap L'(g)) \neq \emptyset \to f$ and g belong to a same simple sub-component of S

```
S' = ParDirCollapse(S, ∅, m);
L' = LabelComponents(S', S^∩);
V = (S^+ \ (S')^+) ∩ F_2^3;
stop = false;
while !stop do
    stop = true;
    foreach f ∈ V do
        if there exists h ∈ S^+ such that f and h are regular neighbours in S and h ∉ V
        then
            for all g ∈ f̂ do
                L'(g) = L'(h);
            end
            V = V \ {f};
            stop = false;
        end
    end
end
cpt = max_{f∈S'} ( max_{i∈L'(f)} i );
while there exists f ∈ S^+ such that L'(f) = ∅ do
    G = GetAllFacesOfComponent(S, f);
    foreach g ∈ G do
        L'(g) = L'(g) ∪ {(cpt + 1)};
    end
    cpt = cpt + 1;
end
return L';
```

7.4 CONCLUSION

In this chapter, we showed that cubical complexes have nice properties, bringing sound decomposition of skeletons into simple components. In this framework, we find many properties of the continuous framework (such as, the intersection between surfaces is a curve), and thanks to this, we can efficiently decompose a skeleton into basic parts. We also closely studied intersections between simple components and, thanks to pinch faces, we have nice properties, in 3d, of intersections between surfaces.

We showed some examples where the decomposition did not match our expectations, and we proposed an algorithm based on more thinning in order to produce better

results (see Alg. 42). The decomposition can also be used as a filtering criterion, in order to remove "spurious" simple components from a skeleton (see Fig. 108c). The cubical complexes reveals to be a very rich framework, giving nice properties for thinning (see previous chapter) and for skeleton decomposition. Some other applications of this work are shown on Fig 109, 110, and 111.

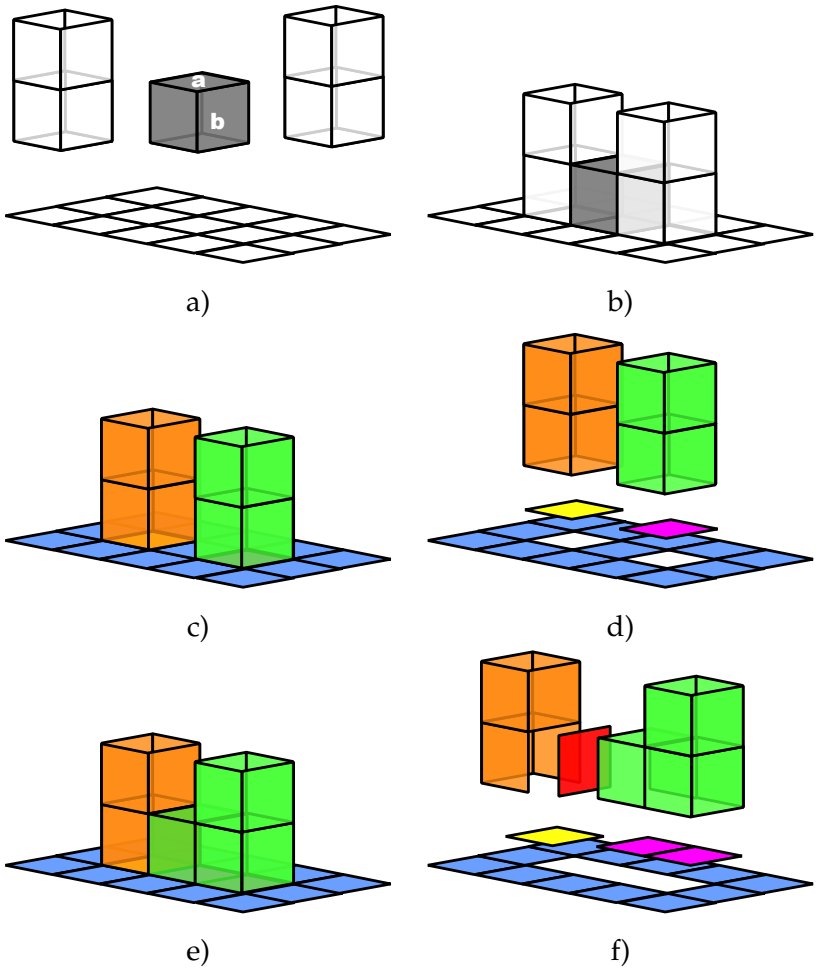


Figure 105: **Various collapse leads to various decomposition -** In **b)**, we present a 3-complex, made of the basic pieces presented on **a)**. The dark cube is a 3-face, and the two squares a and b are free. The other faces of the complex are not free (we consider them as belonging to an inhibitor set). Removing the free pair made of the face a and the cube leads to the decomposition presented on **c)** and **d)**, which is made of five simple 2-components. Removing the free pair made of the face b and the cube leads to the decomposition presented on **e)** and **f)**, which is made of six simple 2-components: there is an extra simple component compared to the decomposition presented in **d)**.

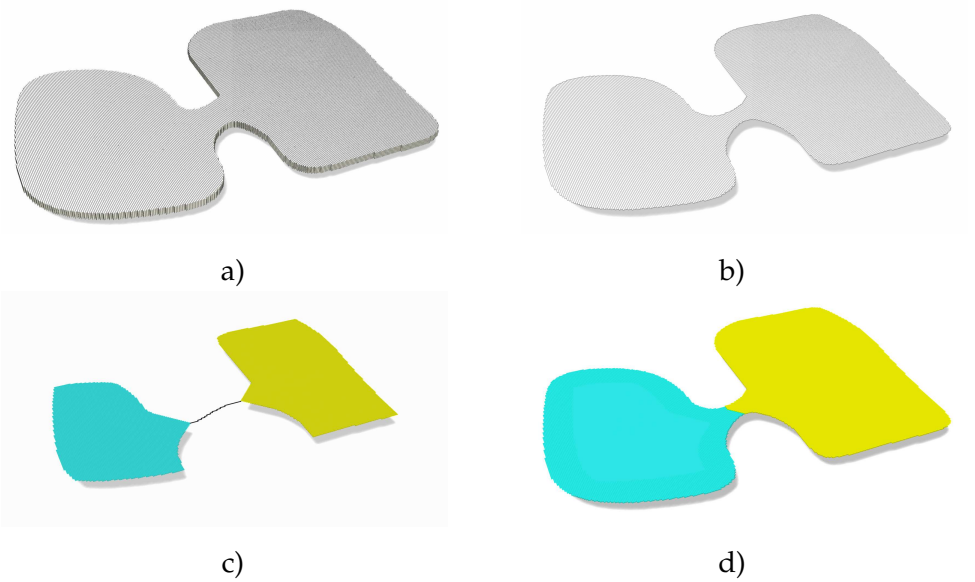


Figure 106: **Two parts joined by a bridge - a)** The original shape is made of two surfacic parts joined by a narrow bridge. **b)** The skeleton of the shape (obtained with Alg. 34 p. 145) is made of one simple component. **c)** Additional thinning of the shape allows to obtain two simple components, but many information has been lost on the border of the skeleton. **d)** Thanks to Alg. 42, it is possible to merge information from **b** and **c** in order to obtain a decomposition of the original skeleton in two simple sub-components.

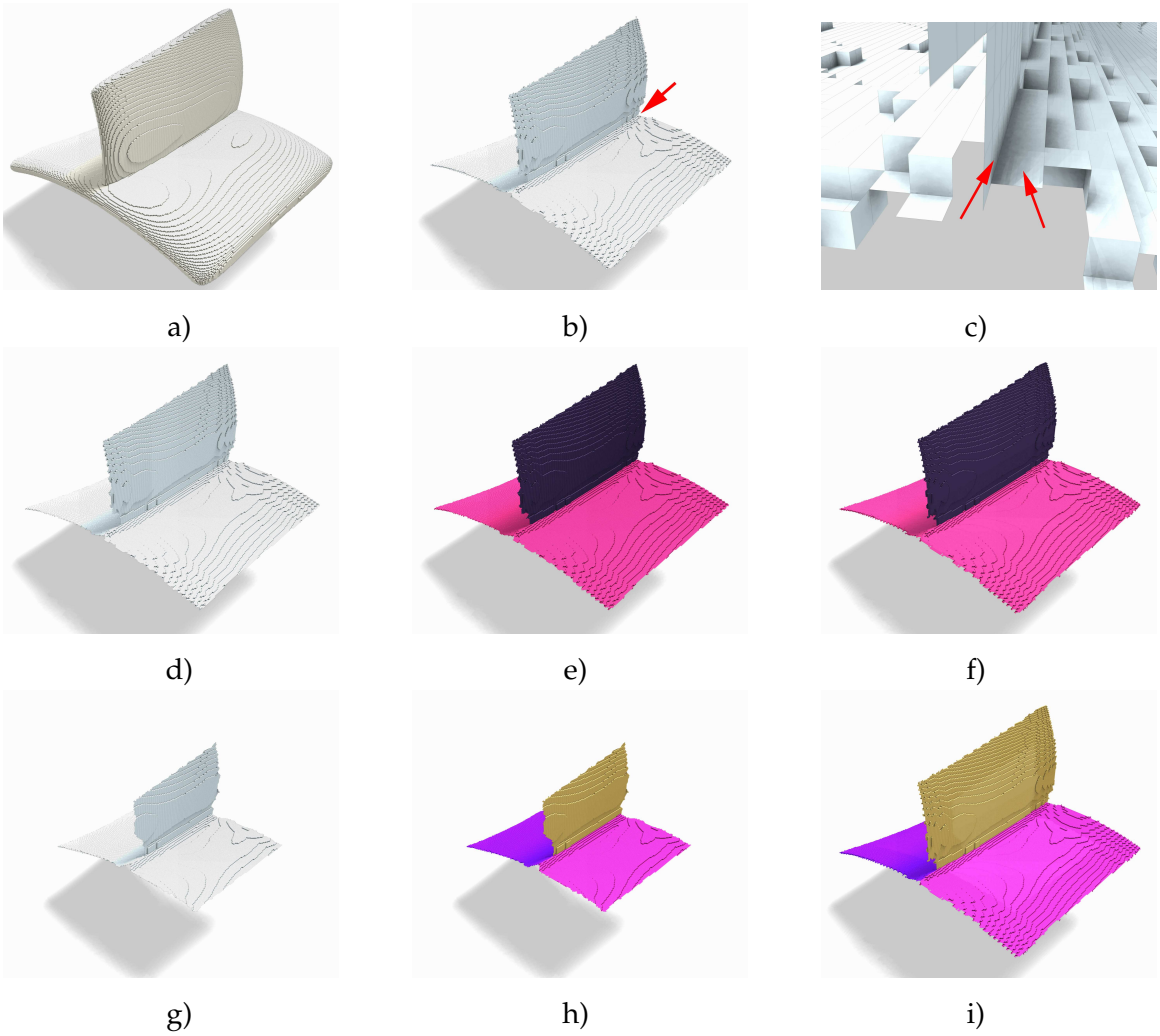


Figure 107: **Three parts intersecting together - a)** The original shape is made of three surfacic parts (one vertical and two horizontal) intersecting together. The two horizontal parts are sharing a narrow bridge. **b)** The skeleton of the shape (obtained with Alg. 34 p. 145) is made of one simple component. In fact, there exists a "small bridge of regular neighbours" (pointed out by the red arrow) linking the vertical part with the horizontal one. **c)** Zoom on the regular neighbours which allowed the vertical part to have the same label than the horizontal part. **d)** Additional thinning allows to remove the small bridge and obtain a decomposition in two simple components of the skeleton on **e)**. This decomposition can be used for decomposing the original skeleton in two simple sub-components with Alg. 42, as shown on **f)**. **g), h), i)** More thinning allows to obtain, with Alg. 42, a decomposition in three simple sub-components of the skeleton.

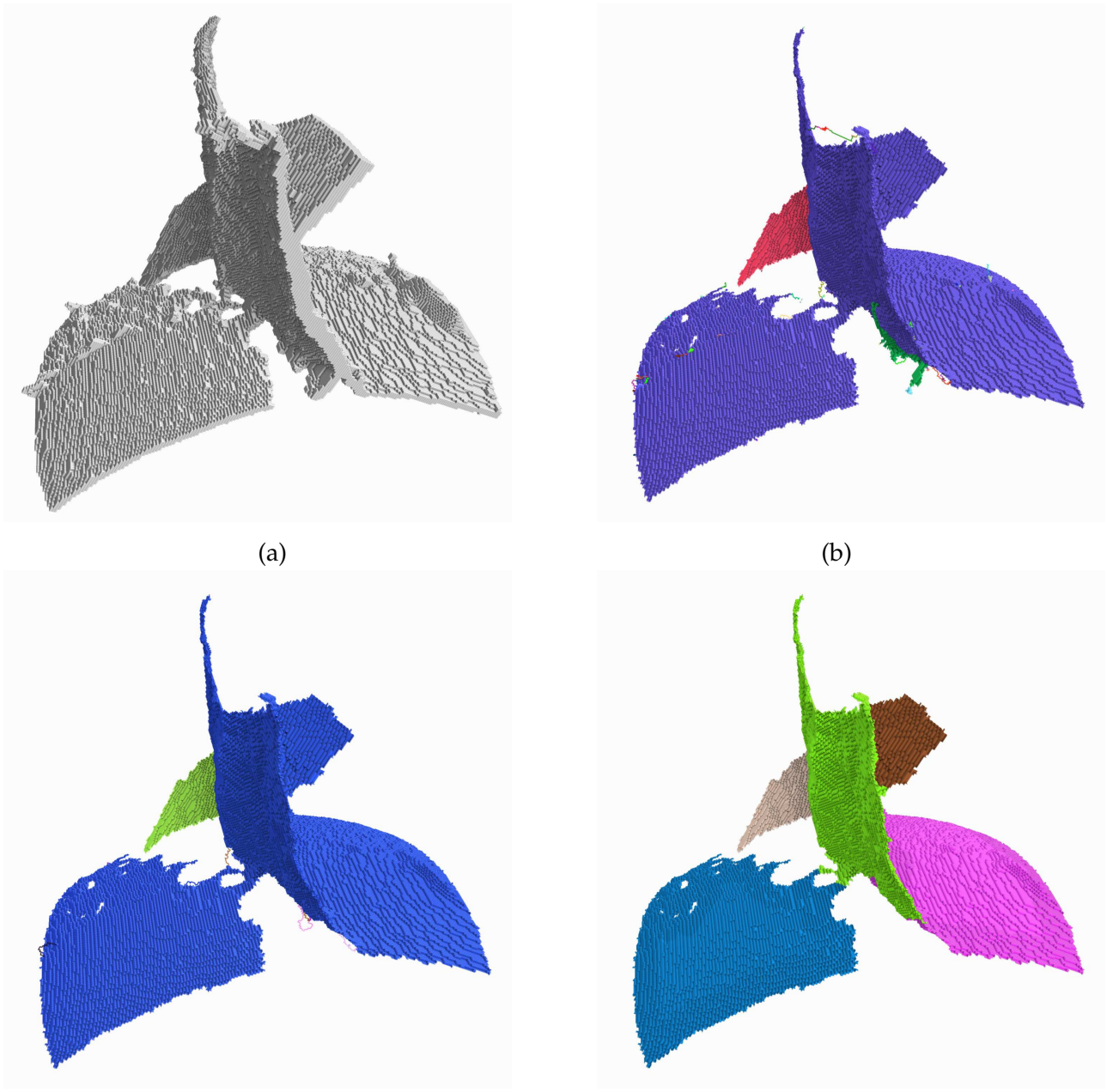


Figure 108: **Cracks in concrete - a)** The original image represents cracks in concrete. Visually speaking, the shape basically consists of five surfaces. **b)** Using SurfacecSkeleton algorithm with hole preserving technique (see Sec. 6.3.4.1 p. 146), we obtain a skeleton whose decomposition gives two large surfaces, and other small simple components. **c)** We remove small components from the skeleton (and use homotopic thinning to make sure we still have a skeleton). **d)** Using `OverCollapseLabel` (a parameter must be specified by the user), the decomposition gives five "sub-surfaces" (even though the three surfaces in green looks like they have the same color, it is not exactly the same green).

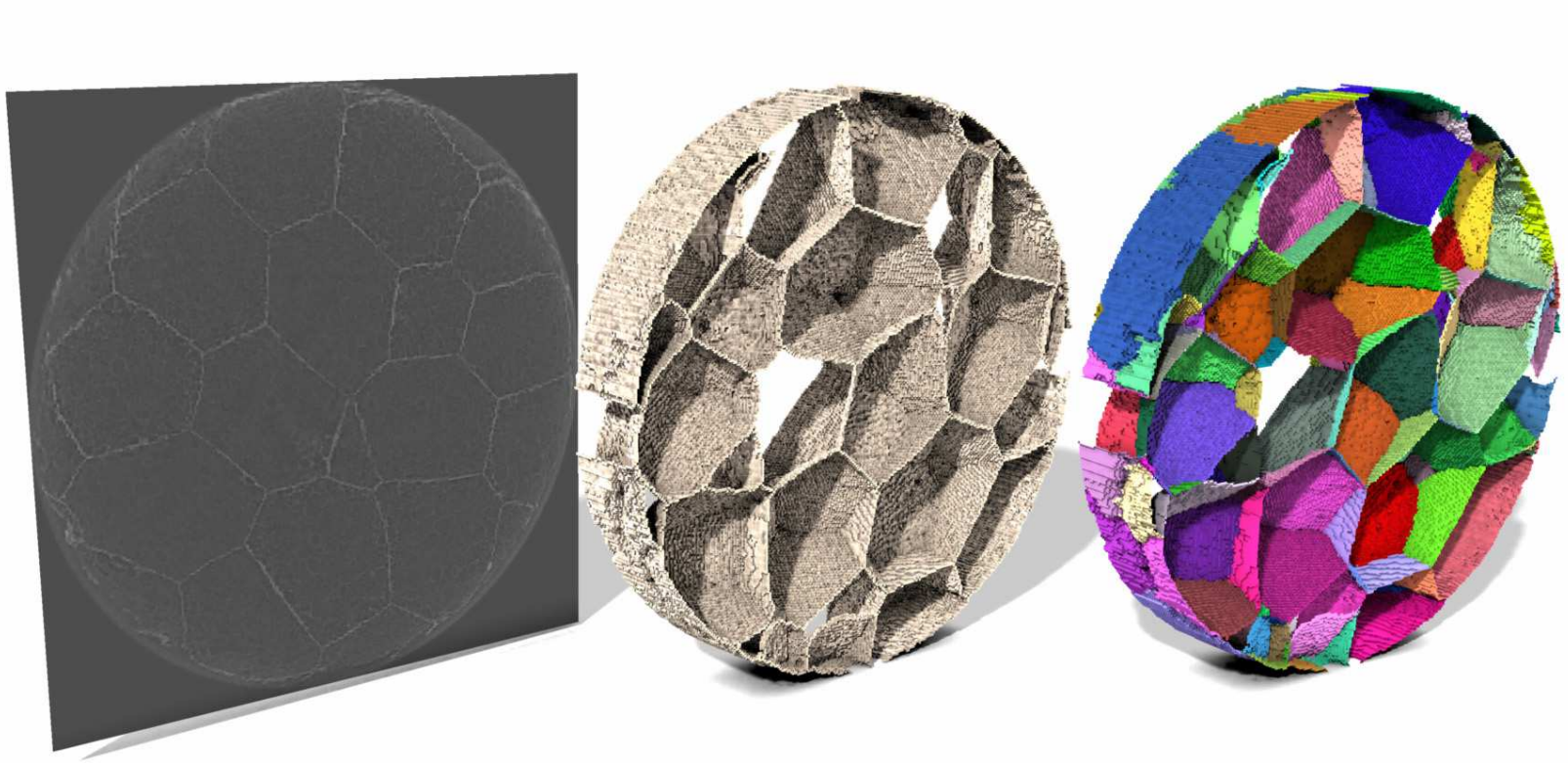


Figure 109: **Metallic foam -** Processing image of metallic foam. **On the left**, a slice of the original image. **In the middle**, using various segmentation method (including watershed), a 3d object is extracted (here, we only show a slice of the whole object). **On the right**, after thinning the object, we decompose the skeleton into simple components.

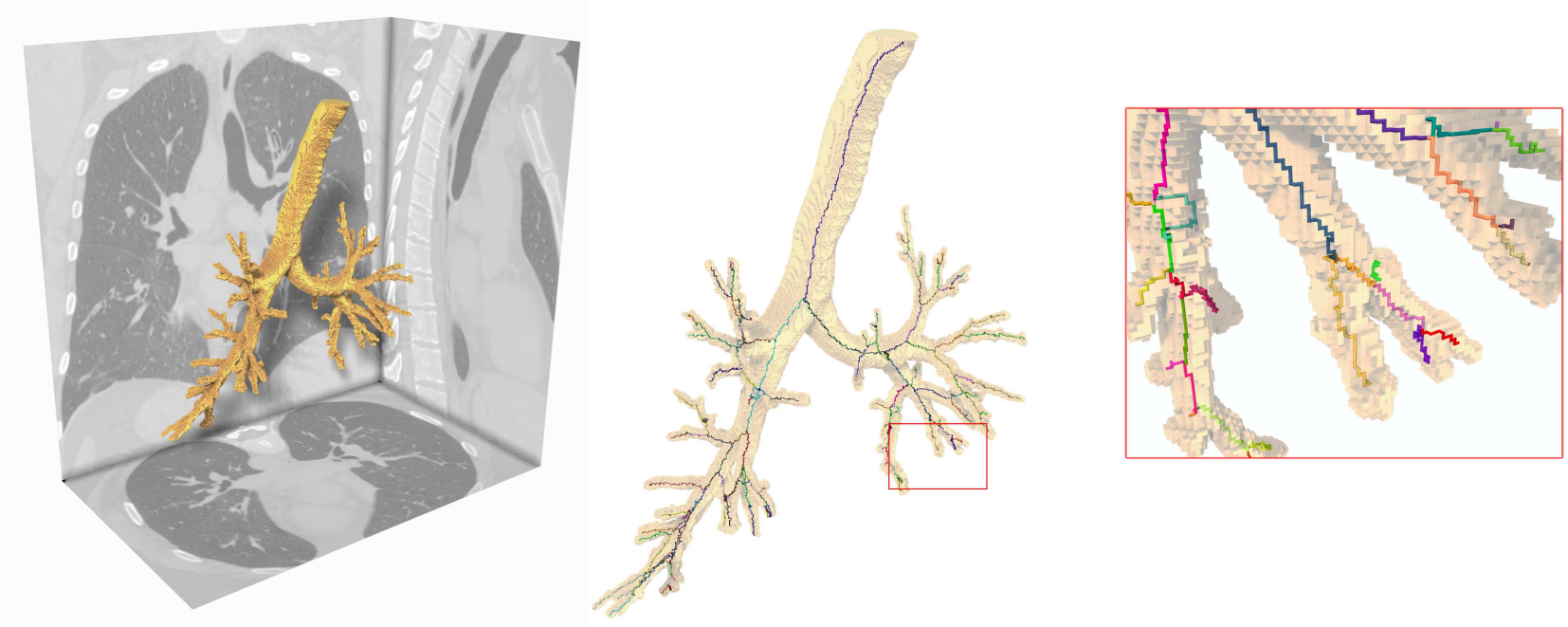


Figure 110: **Lungs -** Processing image of the bronchial tree. This work was realised in partnership with the Technical University of Lodz (Poland). **On the left**, the bronchial tree is segmented using techniques explained in [FJPB09]. **In the middle**, using CurvilinearSkeleton algorithm, we obtain a curvilinear skeleton of the tree, that we then decompose into curves. **On the right**, a close-up on the decomposition of the skeleton.

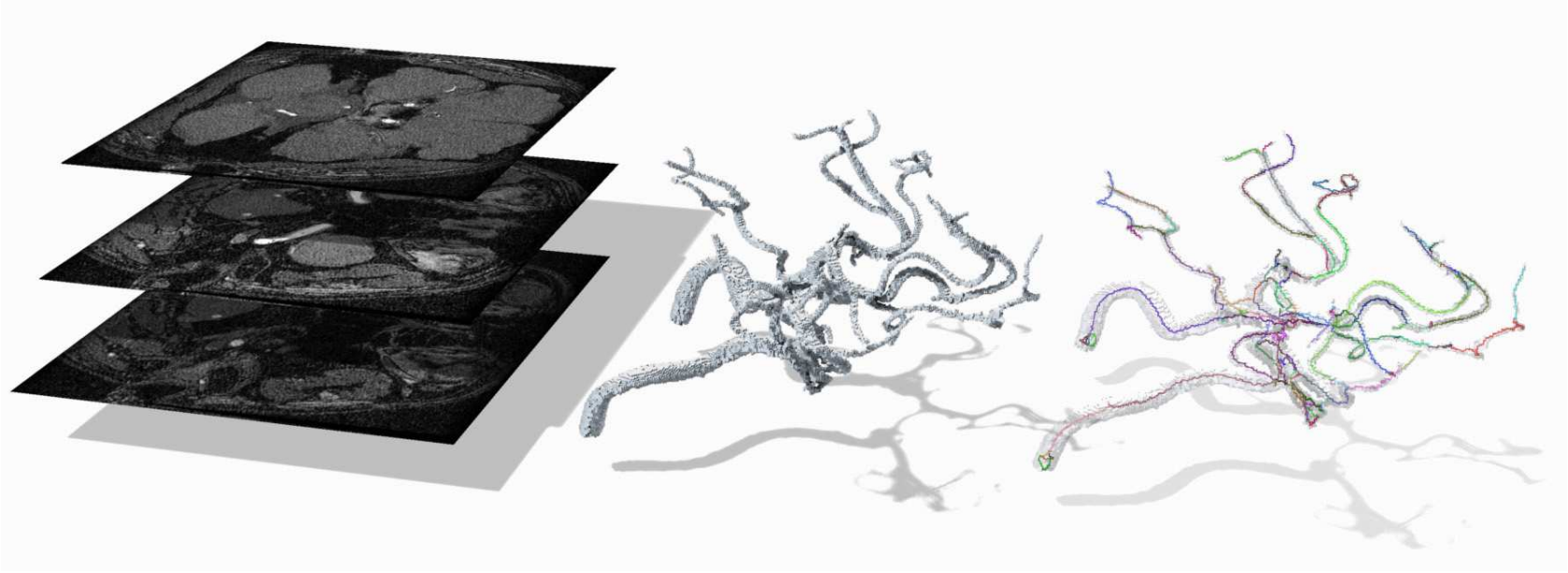


Figure 111: **Cerebral blood vessels -** Processing image of blood vessels situated in the brain. This work was realised in partnership with the LSIIT laboratory in the Strasbourg University (France), and the Centre de Morphologie Mathématiques de l'Ecole des Mines-Paritech (France). **On the left**, some slices from the original image. **In the middle**, the vascular network is extracted using techniques published in [TTDP09]. **On the right**, using `CurvilinearSkeleton` algorithm, we obtain a curvilinear skeleton of the blood vessels structure, that we then decompose into curves.

8

CONCLUSION AND PERSPECTIVES

8.1 CONTRIBUTION OF THIS THESIS

This thesis covered a wide range of subjects related to digital topology and material analysis. The first part explained how fundamental concepts of topology (such as the fundamental group) could help proposing sound environment for fluid flow simulations. We proposed an original algorithm for detecting when an object wraps around a toric space, and we explained how this problem was related with fluid flow simulation.

In the next parts, we treated more general problems which were no more exclusively related to material analysis. In the second part, we introduce the discrete λ-medial axis. We showed how this tool could be used with homotopic thinning algorithms in order to provide skeletons with good reconstruction capacities and few spurious elements. We provided an algorithm allowing to compute, in linear time, a variant of the λ-medial axis and we finally compared the stability to noise and rotation of our axis to the one of other well-known axes. We concluded that the λ-medial axis is a good shape descriptor with nice stability properties.

The DT (or voxel) framework does not provide sound properties for skeleton decomposition. In order to cope with this, we introduced the cubical complex framework in the third part. We showed how to perform parallel thinning in this framework, and how voxel tools (such as the discrete λ-medial axis) could be used in order to provide, in this new framework, skeletons with good reconstruction properties. We then proposed various parameter-free thinning algorithms in the cubical complex framework, and we explained how to adapt these thinning methods to voxel framework. We also prove that our thinning algorithms produce, in the cubical complex framework, thin results. We then compared our thinning methods with other parallel thinning methods; for the sake of a fair comparison, we introduced the visual quality factor of a skeleton, which gives an estimation of how representative of the original shape a skeleton is. Our comparisons showed that our thinning algorithms possess the best stability properties and has among the best visual quality (the results, in 3d, tend to be a little "over filtered" according to the visual quality factor).

Finally, we explained how to decompose a cubical complex into basic elements. We showed that the decomposition of a skeleton in the cubical complex framework can, depending on how the thinning was performed, be more or less satisfactory. We proposed an algorithm that can, thanks to a filtering parameter given by the user, enhance the decomposition of a skeleton (but cannot solve all the problems related with the decomposition). We quickly explained how the decomposition could be used in order to filter the skeleton, and we provided some applications of this work in the material analysis and the medical fields.

In the appendix, we propose different versions of various algorithms presented throughout this thesis, achieving better worst-case complexity.

8.2 PERSPECTIVES

Future developments mostly involve the various thinning techniques developed in the cubical complex framework. The first main development that we would like to achieve is to be able to use the skeleton decomposition in order to decompose the original object into various parts (some preliminary results are shown on Fig. 112). The decomposition of a skeleton in the cubical complex framework could be used in order to provide a decomposition of a voxel object into elements. In order to perform this task, it seems necessary to associate, to each face of a complex, another face related to it by a collapse operation. Based on this, we could build a sort of graph that would allow to associate, to each face of the skeleton, some voxels of the border of the original object.

Figure 112: After decomposing a skeleton into simple components, it is possible to propagate the labels of the skeleton throughout the input object in order to obtain a decomposition of the original volume.

Another development would be to provide a thinning algorithm more robust to noise. Indeed, when the noise level is high on the input shape, some spurious elements appear on our skeletons (as it does for every thinning methods). When using a filtering parameter with the homotopic thinning algorithm (see Sec. 9.2.4), we realise that spurious branches can be characterized by the fact that they can disappear all of a sudden with only a small increase of the filtering parameter (see Fig. 113). Computing a map of the lifespan of all faces of a complex, and studying carefully this map might help in finding and removing spurious branches automatically.

We saw, in the previous chapters, that it can be very difficult to characterize the "visual quality" of a skeleton. We tried to provide the "visual quality factor", but it seems to be globally advantaging "spurious" skeletons. In fact, in the visual quality

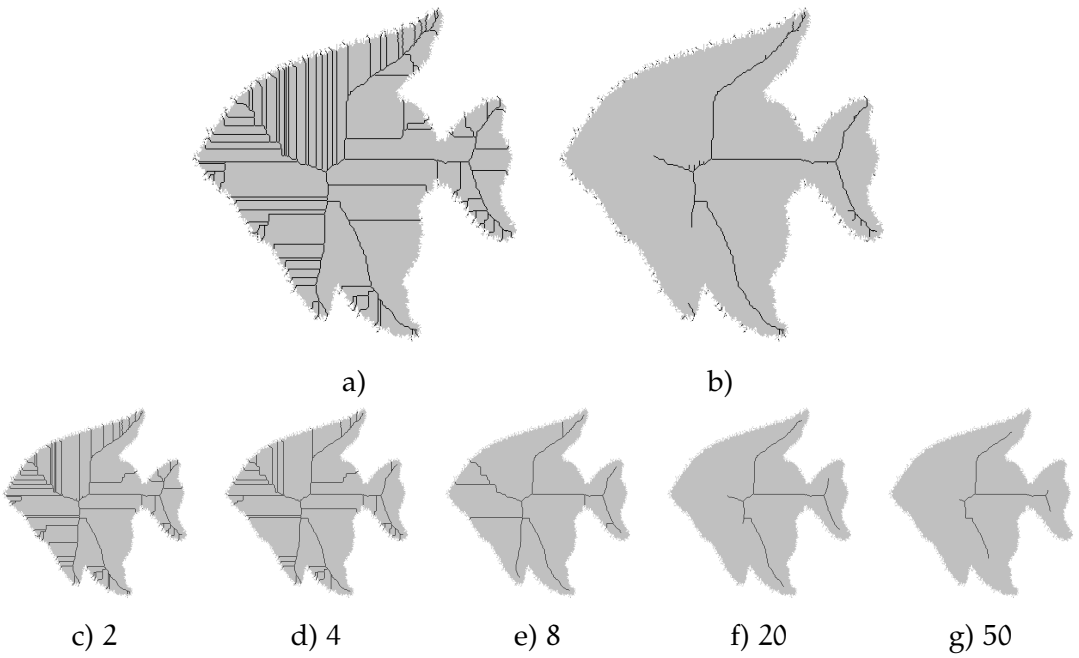


Figure 113: **Homotopic thinning with filtering parameter, and noise - a)** The original shape has a very noisy border: the result of DT2-1 algorithm is a skeleton with many spurious branches. **b)** When isolating only the faces which are kept safe because of their lifespan, and not because of topological considerations, we realise that the spurious branches mainly consist of points that don't have a very high lifespan (only extremities of the spurious branches have a high lifespan). **c), d), e), f), g)** When using a thinning algorithm with a filtering parameter (as explained in 9.2.4), we realise that little increase of the filtering parameter can remove whole spurious branches, while the "interesting branches" have a more "regular" behaviour to filtering parameter increase.

factor, we look at how a skeleton allows to reconstruct the border of the original object. This criteria is a bit too restrictive, and although a skeleton "looks like" the object it was computed from, it may not reconstruct the border very well (especially in 3d). After some test, using the information of the reconstructed volume does not yield good results. A solution might be to compute the dissimilarity between the reconstructed object and the original object. To sum up, the visual quality factor that we introduced previously is not, to our opinion, completely satisfactory, and additional work on it should be achieved.

Finally, we should complete our comparison of thinning algorithms with a study of 3d curvilinear thinning on "curvilinear shaped" objects only, in order to be able to judge of the visual quality of these particular skeletons. Moreover, we would like to add a study on scale invariance of the thinning algorithms.

9

APPENDIX

We give here some extra elements (mainly algorithms) that we decided to isolate from main text in order to keep the reading fluent. Moreover, we give some basic definitions of the main concepts used in digital topology.

Contents

9.1 BASIC DEFINITIONS

Discrete objects An element of the discrete grid $\mathbb{Z}^n$ is called *pixel* when $n = 2$, *voxel* when $n = 3$, and more generally, a *point*. A *discrete object* of $\mathbb{Z}^n$, also called *object* or *shape*, is a subset of $\mathbb{Z}^n$. Given $X \subset \mathbb{Z}^n$, we define the complementary of X as the set $\overline{X} = \{y \in \mathbb{Z}^n | y \notin X\}$. The cardinality of X is denoted by $|X|$. We sometime refer to this framework as *the digital topology (DT) framework*, or as *the voxel space*.

Neighbourhood of a point We define the function $d : \mathbb{Z}^n \times \mathbb{Z}^n \to \mathbb{R}$ which associates, to two points $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ of $\mathbb{Z}^n$, the Euclidean distance between x and y: $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$. The function d can also be denoted by d_e. The neighbourhood of a point x is the set of points of the discrete grid considered to be the closest to x (see Fig. 114 for graphical representation).

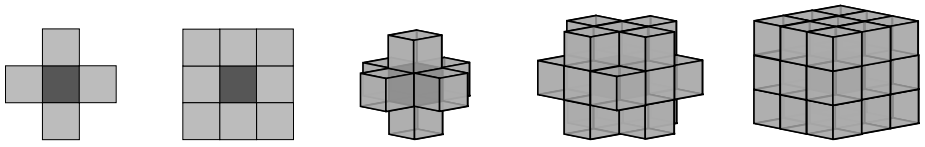
Given $x \in \mathbb{Z}^2$,

- the 4-*neighbourhood* of x is the set $\Gamma_4(x) = \{y \in \mathbb{Z}^2 | d(x, y) \leqslant 1\}$,
- the 8-*neighbourhood* of x is the set $\Gamma_8(x) = \{y \in \mathbb{Z}^2 | d(x, y) \leqslant sqrt(2)\}$.

Given $x \in \mathbb{Z}^3$,

- the 6-*neighbourhood* of x is the set $\Gamma_6(x) = \{y \in \mathbb{Z}^3 | d(x, y) \leqslant 1\}$,
- the 18-*neighbourhood* of x is the set $\Gamma_{18}(x) = \{y \in \mathbb{Z}^3 | d(x, y) \leqslant sqrt(2)\}$,
- the 26-*neighbourhood* of x is the set $\Gamma_{26}(x) = \{y \in \mathbb{Z}^3 | d(x, y) \leqslant sqrt(3)\}$.

More generally speaking, given $x \in \mathbb{Z}^n$, the *direct neighbourhood* of x is the set $N(x) = \{y \in \mathbb{Z}^n | d(x, y) \leqslant 1\}$.



(a) (b) (c) (d) (e)

Figure 114: On **a)**, the elements of $\Gamma_4(x)$ (x is the dark pixel in the centre), and on **b)**, the elements of $\Gamma_8(x)$. On **c)**, the elements of $\Gamma_6(x)$ (x is the dark pixel in the centre that can be seen through transparency of other pixels), on **d)**, the elements of $\Gamma_{18}(x)$, and on **e)**, the elements of $\Gamma_{26}(x)$.

Generally speaking, given two points x and y of $\mathbb{Z}^n$ and an integer k such that Γ_k is a neighbourhood defined on $\mathbb{Z}^n$, we say that x and y are k-neighbours if $y \in \Gamma_k(x)$. The strict k-neighbourhood of a point x is the set $\Gamma_k^*(x) = \Gamma_k(x) \backslash \{x\}$; the same way, we define $N^*(x) = N(x) \backslash \{x\}$.

In the following, when using a k-neighbourhood in the discrete space $\mathbb{Z}^n$, then it will be considered that Γ_k is well defined on $\mathbb{Z}^n$.

In $\mathbb{Z}^n$, Γ_k is an *adjacency relation*, defining which points can be considered as neighbours and which points cannot.

In $\mathbb{Z}^2$, when considering an object X with the 8-adjacency relation, its complementary should be considered with the 4-adjacency relation (and vice-versa) in order to have, for both sets, sound topological properties. In $\mathbb{Z}^3$, when considering an object X with the 26-adjacency relation, its complementary should be considered with the 6-adjacency relation (and vice-versa).

Given $X \subset \mathbb{Z}^n$ a k-connected object, and $\overline{k}$ the connectivity of $\overline{X}$, a point $x \in X$ belongs to the *border of* X if $\Gamma_{\overline{k}}(x) \cap \overline{X} \neq \emptyset$.

Distances and distance transform Given x and $y \in \mathbb{Z}^n$, a k-*path between* x *and* y is a sequence of points $\mathcal{S} = (p_1, ..., p_j)$ such that $x = p_1$, $y = p_j$ and, for each $i \in [1, j-1]$, p_i and p_{i+1} are k-neighbours. The length of $\mathcal{S}$ is $(j-1)$.

The k-*distance between* x *and* y, denoted by $d_k(x, y)$, is the length of the shortest k-path between x and y. Given $Y \subset \mathbb{Z}^n$, we set $d_k(x, Y) = \min_{y \in Y} d_k(x, y)$. Given $X \subset \mathbb{Z}^n$, we set $d_k(X, Y) = \min_{x \in X} d_k(x, Y)$.

The k-*distance transform of* X is the function $Dk_X : \mathbb{Z}^n \to \mathbb{R}$ such that, for each $x \in \mathbb{Z}^n$, $Dk_X(x) = d_k(x, \overline{X})$. The k-distance transform of X allows to know, for each point x of X, the length of the shortest path between x and any element of $\overline{X}$.

The exact same way, we can define the *Euclidean distance transform* of an object X, usually denoted by D_X. Algorithms exist ([Hir96], [MRHoo]) in order to compute the Euclidean distance transform of a shape in linear time (with respect to the size of the input). The Euclidean distance transform of X is also called the *Euclidean distance map* of X.

Discrete balls and Euclidean balls Given $x \in \mathbb{Z}^n$ and $r \in \mathbb{R}$, the k-*ball of center* x *and radius* r is the set $B_k(x, r) = \{y \in \mathbb{Z}^n | d_k(x, y) \leqslant r\}$. The *strict* k-*ball of center* x *and radius* r is the set $B_k^<(x, r) = \{y \in \mathbb{Z}^n | d_k(x, y) < r\}$.

We also define *the Euclidean ball of center* x *and radius* r as the set $B(x, r) = \{y \in \mathbb{Z}^n | d(x, y) \leqslant r\}$, and *the strict Euclidean ball of center* x *and radius* r as the set $B^<(x, r) = \{y \in \mathbb{Z}^n | d(x, y) < r\}$.

Connected components Given $X \subset \mathbb{Z}^n$, X is k-*connected* if, for each $x, y \in X$, there exists a k-path between x and y and entirely lying inside X. A subset $Y \subset X$ is a k-*connected component of* X if it is k-connected and if it is not strictly included in any other k-connected subset of X.

9.2 EFFICIENT ALGORITHMS

Here, we give more efficient (with regards to complexity) versions of some algorithms defined in chapter 6.

9.2.1 ANOTHER DIRECTIONAL PARALLEL THINNING ALGORITHM

The ParDirCollapse algorithm (see Alg. 27, p. 133) is the basic homotopic thinning procedure used by most thinning algorithms proposed in Chap. 6. We propose, in Alg. 44, a modified version that will be useful for building efficient versions of other algorithm based on the ParDirCollapse. This algorithm relies on the map S which, when

correctly initialized with Alg. 43, gives all free faces of a cubical complex X for a given type, orientation and dimension of faces.

Algorithm 44 performs the same task than Alg. 27, only in a more efficient way (with regards to the complexity). First, all the free faces of the cubical complex are classified in the set S according to their type, orientation and direction on line 4. Then, the border of the complex is built based on the set S, and free faces are removed in parallel. Once a removal took place, the algorithms looks for new free faces in the complex only in the star of the removed face, or in the faces it contained (see lines 31 and 37).

Notice that each free pair of faces stays in the set S for at most two iterations of the loop line 8. Indeed, the pair is removed from S in two cases:

- If the pair is no more free for X (see l. 24). In this case, as the pair was free before (only free pairs of faces are added to S), it means that the face of greater dimension of the pair was removed from the complex. The pair of faces will therefore never be added again to S.
- The pair is free for X and belongs to the border of X (see l. 22). In this case, the pair is removed also from the complex, and will therefore never be added again to S.

If the pair is free but does not belong to the border, it remains in S. In this case, the pair will necessarily, at the next iteration, either belong to the border of X (if it is still free at the end of the current iteration) or it won't be free any more: in both cases, the pair will be removed from S and will never be added again to it.

Each free pair is therefore scanned at most twice on lines 22 and 24. The algorithm, for a given dimension, is linear with regards to the number of faces of the input. It is also quadratic with regards to the dimension.

Algorithm 43: $\mathtt{InitSetS}(X, W)$

Data: A cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed.
Result: A map $S : \{0, ..., n-1\} \times \{0, 1\} \times \{1, ..., n\} \to \mathbb{F}^n \times \mathbb{F}^n$

```
1  S = ∅;
2  foreach f ∈ (X\W) do
3  |  foreach g ∈ (f̂\W) do
4  |  |  if (f, g) is free for X then
5  |  |  |  t = Dir(f, g);
6  |  |  |  s = Orient(f, g);
7  |  |  |  d = dim(f);
8  |  |  |  S(t, s, d) = S(t, s, d) ∪ {(f, g)};
9  |  |  end
10 |  end
11 end
12 return S;
```

Algorithm 44: ParDirCollapse2(X, W, ℓ, S)

Data: A cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed, and $\ell \in \mathbb{N}$, the number of layers of free faces which should be removed from X, a map S of all free pairs of faces, categorised by type, orientation and direction (see Alg. 43).

Result: A cubical complex and the map S

$L = \emptyset$;
if S *is the null function* **then**
  $S = \mathrm{InitSetS}(X, W)$;
end
while S *has a non-empty set as image and* $\ell > 0$ **do**
  for $t = 0 \to n-1$ **do**
    for $s = 0 \to 1$ **do**
      for $d = n \to 1$ **do**
        foreach $(f, g) \in S(t, s, d)$ **do**
          $L = L \cup \hat{f}$;
        end
      end
    end
  end
  for $t = 0 \to n-1$ **do**
    for $s = 0 \to 1$ **do**
      for $d = n \to 1$ **do**
        $E = \{(f, g) \in S(t, s, d) \mid (f, g)$ is free for X and $f \in L$ and $g \in L\}$;
        $G = \{(f, g) \in S(t, s, d) \mid (f, g)$ is not free for $X\}$;
        $X = X \setminus E$;
        $S(t, s, d) = S(t, s, d) \backslash (G \cup E)$;
        foreach $(f, g) \in E$ **do**
          foreach $h \in (\hat{f} \cap X)$ *such that* $\dim(h) = (d-1)$ **do**
            foreach $e \in (\check{h} \cap X)$ *such that* $\dim(e) = d$ *and such that* (e, h) *is free for* X *and* $h \notin W$ **do**
              $t = Dir(e, h)$;
              $s = Orient(e, h)$;
              $S(t, s, d) = S(t, s, d) \cup \{(e, h)\}$;
            end
            foreach $i \in \hat{h}$ *such that* $\dim(i) = (d-2)$ *and such that* (h, i) *is free for* X *and* $i \notin W$ **do**
              $t = Dir(h, i)$;
              $s = Orient(h, i)$;
              $S(t, s, d-1) = S(t, s, d-1) \cup \{(h, i)\}$;
            end
          end
        end
      end
    end
  end
  $l = l - 1$;
end
return (X, S);

9.2.2 OTHER COLLAPSE ALGORITHMS BASED ON ANCHOR DETECTION

We give hereafter (see Alg. 28) a linear version (with regards to the number of faces of the object) of the CollapseFacet algorithm (Alg. 28), which removes free faces from an object layer by layer, and detects facets after each layer removed. Thanks to the use of Alg. 44, the algorithm keeps tracks of new free faces after removing one layer of free faces (see l. 8).

Notice that, during the thinning, a facet appears after one of the face of its star was removed. Observe that only parts of the complex which were changed might be locations of new facets after removing one layer of free faces. Therefore, by saving, before each turn of collapse, the sets of free faces (see l. 6) in S', we can limit the research of new facets in the complex to S'. In order to detect the facets already present in the complex before any thinning, we use the set W' which saves all the facets in the complex at the beginning (l. 3), and add them once to the inhibitor set (see l. 23).

As explained previously in Sec. 9.2.1, as each pair of face can only stay in S for at most two turns of the main loop, one can see that Alg. 28 is linear with regards to the number of faces of the object.

The same way, we can propose a linear version of algorithm 29 (CollapseIshtmus), by simply modifying the tests performed on l. 3 and 23 of Alg 45.

9.2.3 ANOTHER ALGORITHM FOR COMPUTING THE LIFESPAN OF A FACE

We give hereafter (see Alg. 46) a linear version (with regards to the number of faces of the object) of the Lifespan algorithm (Alg. 31), which computes the lifespan of all faces of a complex. On l. 24, the detection of new facets in the set Y is only done "around" faces which were in S on l. 14 and are no more in Y: these faces were maybe removed from the complex, so we look if the faces they contained did not become facets. Indeed, new facets can only be located where other faces were removed. As a given pair of free faces can only stay in the set S' for at most two iterations of the main loop, Alg. 46 is linear with regards to the number of faces of the object.

9.2.4 HOMOTOPIC THINNING ALGORITHMS WITH A FILTERING PARAMETER

In Chap. 6, we proposed three homotopic thinning algorithms in the cubical complex framework, which do not require any user input. When the input shape contains much noise, the skeletons produced by the algorithms may contain spurious branches. We propose hereafter another version of these algorithms (see Alg. 47, 48 and 49), taking as input a filtering parameter (a decimal value) in order to remove spurious branches if the user wishes to do so. The same way than we did in Chap. 6, we can adapt these algorithms to the DT framework.

We define the following sets:

$$\mathcal{LC}_k^{>p}(X) = \{f \in X \mid \dim(f) = k \text{ and } \mathrm{Lifespan}_X(f) > \mathrm{Decenter}(X)(f) + p\}^-.$$

and

$$\mathcal{LOC}_k p(X) = \{f \in X \mid \dim(f) = k \text{ and } \mathrm{Lifespan}_X(f) > \tilde{\Omega}_1(X)(f) + \mathrm{Decenter}(X)(f) + p\}^-.$$

Algorithm 45: CollapseFacet2(X, W)

Data: An n-dimensional cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed
Result: A cubical complex

```
1  S = InitSetS(X, W);
2  W' = {f̂ ∈ X | f ∈ X⁺ and dim(f) ⩽ (n − 1)} ;
4  while S has a non-empty set as image do
5      S' = S; E = ∅ ;
7      (X, S) = ParDirCollapse2(X, W, 1, S) ;
9      for t = 0 → n − 1 do
10         for s = 0 → 1 do
11             for d = n → 1 do
12                 foreach (f, g) ∈ S'(t, s, d) do
13                     if f ∉ X then
14                         E = E ∪ {f̂ ∩ X};
15                     end
16                 end
17             end
18         end
19     end
20     foreach e ∈ E do
21         if e ∈ X⁺ and dim(e) ⩽ (n − 1) then
23             W = W ∪ ê ;
24         end
25     end
26     W = W ∪ W';
27     W' = ∅;
28 end
29 return X;
```

9.2.5 ANOTHER ALGORITHM FOR COMPUTING THE OPENING FUNCTION

9.2.5.1 AN OPENING FUNCTION FOR 2D AND 3D IMAGES, IN 8/26 CONNECTIVITY

The opening function gives, for all points x of a subset X of $\mathbb{Z}^n$, the size of the maximal ball of X which contains x (the radius of the ball is the value given to the point). We gave, in Alg. 32 p. 143, a naive version of the opening function which is time consuming.

The opening function based on discrete distances can be seen as the successive dilation of the distance transform of an object, by the values of each points: for example, if the value of the distance transform of a point x is 3, than the value 3 should be copied to all points located at a distance of less than 3 of x. Repeating this procedure for all points, from the lowest value to the highest value, allows to obtain the opening function. This procedure is also time consuming, however, interpretation of the opening function as the successive dilation of values of the distance map helps understanding the following algorithm.

Algorithm 46: Lifespan2(X)

Data: A cubical complex $X \preceq \mathbb{F}^n$
Result: The map giving the lifespan of all faces of X

```
1  for all f ∈ X do
2  |  Death(f) = +∞;
3  |  if f ∈ X⁺ then
4  |  |  Birth(f) = 0;
5  |  end
6  |  else
7  |  |  Birth(f) = +∞;
8  |  end
9  end
10 Y = X;
11 S = InitSetS(Y, W);
12 while S has a non-empty set as image do
13 |  S'_4 = S ;
15 |  (Y, S) = ParDirCollapse2(Y, ∅, 1, S);
16 |  l = l + 1;
17 |  for t = 0 → n − 1 do
18 |  |  for s = 0 → 1 do
19 |  |  |  for d = n → 1 do
20 |  |  |  |  foreach (f, g) ∈ S'(t, s, d) such that f ∉ Y do
21 |  |  |  |  |  foreach e ∈ f̂ do
22 |  |  |  |  |  |  if e ∈ Y⁺ and Birth(e) = +∞ then
24 |  |  |  |  |  |  |  Birth(e) = l;
25 |  |  |  |  |  |  end
26 |  |  |  |  |  |  if e ∉ Y and Death(e) = +∞ and e ∈ X then
27 |  |  |  |  |  |  |  Death(e) = l;
28 |  |  |  |  |  |  end
29 |  |  |  |  |  end
30 |  |  |  |  end
31 |  |  |  end
32 |  |  end
33 |  end
34 end
35 for all f ∈ X do
36 |  Birth(f) = min(Birth(f), Death(f));
37 |  if Death(f) = +∞ then
38 |  |  Lifespan(f) = +∞;
39 |  end
40 |  else
41 |  |  Lifespan(f) = Death(f) − Birth(f);
42 |  end
43 end
44 return Lifespan;
```

Algorithm 47: $\mathrm{1DSkeleton}'(X, p)$

Data: A cubical complex $X \preceq \mathbb{F}^2$, a decimal value p
Result: A cubical complex $Y \preceq \mathbb{F}^2$ such that $\dim Y \leqslant 1$
1 $W = \mathfrak{LC}_1^{>p}(X)$;
2 $\mathrm{ParDirCollapse}(X, W, +\infty)$;
3 **return** X;

Algorithm 48: $\mathrm{SurfaceSkeleton}'(X, p_1, p_2)$

Data: A cubical complex $X \preceq \mathbb{F}^3$, two decimal values p_1 and p_2
Result: A cubical complex $Y \preceq \mathbb{F}^3$ such that $\dim Y \leqslant 2$
1 $W = \mathfrak{LC}_2^{>p_1}(X) \cup \mathfrak{LOC}_1^{>p_2}(X)$;
2 $\mathrm{ParDirCollapse}(X, W, +\infty)$;
3 **return** X;

Algorithm 52 allows to compute efficiently the opening transform of a 26-connected 3d object. It can be easily adapted for computing the opening transform of an 8-connected 2d object. In order to do this, let us first explain the role of Alg. 51: given a one-dimensional array T of size n containing integer values, it produces an array T' such that, for each $i \in \{0, ..., n-1\}$, $T'[i] = m$ iff m is the highest value such that there exists $j < i$ such that $T[j] = m$ and $(i - j) < m$. In Alg. 51, there exists only one array, which is T at the beginning of the algorithm, and which is T' at the end.

Algorithm 51 can be seen as a procedure for spreading, in a one-dimensional array, values to the right of the array: a value of 3 would be spread up to three cells on the rights, while a value of 5 would be spread up to 5 cells on the right (when a pixel can have multiple values, only the highest value is kept). In order to do this, the algorithm scans the array from left to right, while keeping in a list (called MyList) the values that were previously read, and the highest value which should be spread for the moment. An example of the output of the algorithm is presented on Fig. 115.

When a new cell i is read, we first remove, with Alg. 50, all values of the list which are smaller than $T[i]$. Indeed, these values are dominated by $T[i]$ for all cells located after i, so we don't need to keep them. Then, multiple cases can happen. First, the actual maximal value which was dilated until now must no more be dilated (for example, if the value was 5, it was already dilated four times). In this case (l. 8), the actual value is replaced by the highest value of the list which should now spread on the cell i (l. 10).

If the actual maximum value is higher than the value located in the cell i, then the content of the cell i is saved in the list (at the end of the list) and replaced by the actual maximal value (l. 19). On the opposite, if the actual maximum value is lower or equal than the value located in the cell i, then the maximal value is replaced by i (l. 26).

The opening function can be computed by using Alg. 51 one each lines, columns, etc, of an image. The process is detailed for 3d images in Alg. 52: the procedure LineScanIndirect is actually the same than Alg. 51, except that, on l. 5, the array should be scanned from the end to the beginning, and on l. 8, 10 and 14, the addition should be replaced a subtraction, and the " $\leqslant$ " should be replaced by a " $\geqslant$ ".

The function LineScanDirect is linear with regards to the number of cells of the input array. Indeed, in this function, elements are added to the end or to the head of the list (operation performed in constant time). Moreover, the function RemoveSmallerThan scans

Algorithm 49: $\mathtt{CurvilinearSkeleton}'(X, p)$

Data: A cubical complex $X \preceq \mathbb{F}^3$, a decimal value p
Result: A cubical complex $Y \preceq \mathbb{F}^3$ such that $\dim Y \leqslant 2$

```
W = LOC_1^{>p}(X);
ParDirCollapse(X, W, +∞);
return X;
```

(and deletes) from the list, in total, at most $(n-1)$ elements, where n is the number of cells of the input one-dimensional array. In conclusion, the function QuickOpening3d is linear with regards to the number of points of the object.

a)	2	1	3	4	2	2	3	1	2
b)	2	2	3	4	4	4	4	3	3

Figure 115: **Result of Alg. 51:** On **b**, we present the result of running Alg. 51 on array **a**.

Algorithm 50: $\mathtt{RemoveSmallerThan}(\mathtt{MyList}, value, \mathtt{Tab})$

Data: A list MyList containing indexes of Tab, sorted from higher (head) to lower (tail) value in Tab, and a value $value$.
Result: The procedure erases all indexes of MyList which have, in Tab, a value smaller than $value$.

```
while MyList is not empty do
    val = PopTail(MyList);
    if T[val] > value then
        AddToTail(val, MyList);
        break;
    end
end
```

9.2.5.2 DISCUSSION ON OPENING FUNCTION IN 4 OR 6 CONNECTIVITY

In the work exposed previously, we need the opening function of a 2d object in 4-connectivity, and the opening function of a 3d object in 6-connectivity. However, the function QuickOpening3d provides only opening function in 26-connectivity (and can be adapted for 8-connectivity in 2d). In the QuickOpening3d function, we rely on the following:

Proposition 9.2.1 *Let* $X \subseteq \mathbb{Z}^3$*, and let* A *(resp.* B*) be a maximal 26-ball of* X*, of radius* r_a *(resp.* r_b*) and of center* (c, a_1, a_2) *(resp.* (c, b_1, b_2)*) and such that* $r_a > r_b$*.*

For all $x \in X$ *such that* $x = (d, e, f)$ *with* $d \in \{c - r_a + 1, ..., c + r_a - 1\}$*, if* $x \in B$*, then* $X \in A$*.*

In other words, this proposition states that, on a same line of an image, if a 26-maximal ball A "covers" a 26-connected maximal ball B on a point x of X, then A will

Algorithm 51: LineScanDirect(T, n)

Data: A one-dimensional array T of size n
Result: The procedure dilates, from left to right, all values of T (see detailed explanation in the text).

```
1  MyList = GenerateEmptyList();
2  max = 0;
3  for i = 1 → n do
5      RemoveSmallerThan(MyList, T[i], T);
6      if max + T[max] ⩽ i then
8          while max + T[max] ⩽ i and MyList is not empty do
10             max = PopHead(MyList);
11         end
12         if max + T[max] ⩽ i then
14             max = i;
15         end
16     end
17     else if T[max] > T[i] then
19         T[i] = max;
21         AddToTail(MyList, i);
22     end
23     else
24         max = i;
26
27     end
28 end
```

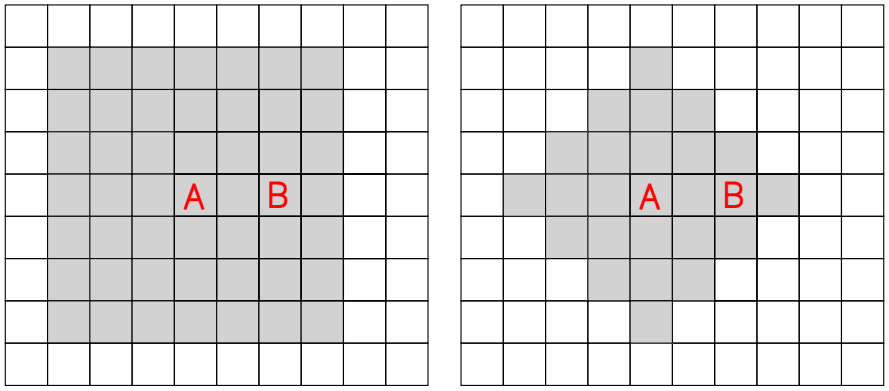
also cover B on the column and rank of x (see Fig. 116a for an example). Thanks to this property, on l. 21 of Alg. 51, only one value is saved in the cell, and not the entire list MyList. Thanks to this, we keep manipulating arrays of integer in Alg. 52, and not arrays of lists.

When working with 6-maximal balls (or 4-maximal balls in 2d), the previous property does not hold (see a counter example on Fig. 116b), and the general scheme or Alg. 52 does not allow to obtain the opening function of an object in 4 or 6 connectivity. However, a non-tested strategy would be to not scan the image by rows, columns, ranks, etc..., but rather scan it in diagonal directions (as if the image had been rotated of 45°). Thanks to this scan strategy, we could build a similar proposition than Prop. 9.2.1 adapted to 6 or 4 connectivity, and propose a version of Alg. 52 for 6 or 4 connectivity.

Algorithm 52: QuickOpening3d(X)

Data: A set $X \subset \mathbb{Z}^3$
Result: The 1-distance opening function of X, that is, for all $x \in X$, $\omega_1(x, \overline{X})$

```
Let D be the 26-distance transform of X, considered as a three-dimensional array;
for k = 0 → DepthOf(D) do
    for j = 0 → HeightOf(D) do
        LineScanDirect(D[k][j][*], LengthOf(D));
        LineScanIndirect(D[k][j][*], LengthOf(D));
    end
end
for k = 0 → DepthOf(D) do
    for i = 0 → LengthOf(D) do
        LineScanDirect(D[k][*][i], HeightOf(D));
        LineScanIndirect(D[k][*][i], HeightOf(D));
    end
end
for j = 0 → HeightOf(D) do
    for i = 0 → LengthOf(D) do
        LineScanDirect(D[*][j][i], DepthOf(D));
        LineScanIndirect(D[*][j][i], DepthOf(D));
    end
end
```



a) b)

Figure 116: **A nice property on squares -** On **a**, we consider the grey 8-ball of center A and radius 4 and the dotted 8-ball of center B and radius 3. For every point x located in the grey ball and located on the same line than A, and for every point y located on a same column than x, if y is contained in the dotted ball, then it is contained in the grey ball. On **b**, we consider the same balls as previously, but in 4-connectivity. The point B is located on the same line than A, is included in the grey ball, however, there exists a point on the same column than B, contained in the dotted ball and not contained in the grey ball (third line from the top, seventh column from the left).

BIBLIOGRAPHY

[ABE94] Yannick Anguy, Dominique Bernard, and Robert Ehrlich. The local change of scale method for modelling flow in natural porous media (I): Numerical tools. *Advances in Water Resources*, 17(6):337–351, 1994.

[ABE09] Dominique Attali, Jean-Daniel Boissonnat, and Herbert Edelsbrunner. Stability and computation of the medial axis — a state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 1–19, 2009.

[AC97] Narendra Ahuja and Jen-Hui Chuang. Shape Representation Using a Generalized Potential Field Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):169–176, 1997.

[AL99] Dominique Attali and Jacques-Olivier Lachaud. Constructing iso-surfaces satisfying the Delaunay constraint: application to the skeleton computation. In *ICIAP 1999: Proceedings of the 10th International Conference on Image Analysis and Processing*, pages 382–387, Venice (Italy), 1999.

[AL01] Dominique Attali and Jacques-Olivier Lachaud. Delaunay Conforming Iso-surface, Skeleton Extraction and Noise Removal. *Computational Geometry: Theory and Applications*, 19:175–189, 2001.

[AM96] Dominique Attali and Annick Montanvert. Modelling noise for a better simplification of skeletons. In *ICIP 1996: Proceedings of the 3rd IEEE International Conference on Image Processing (ICIP)*, volume 3, pages 13–16, 1996.

[AM97] Dominique Attali and Annick Montanvert. Computing and Simplifying 2D and 3D Continuous Skeletons. *Computer Vision and Image Understanding*, 67(3):261–273, 1997.

[Arc81] Carlo Arcelli. Pattern thinning by contour tracing. *Computer Graphics and Image Processing*, 17(2):130–144, 1981.

[AS80] Carlo Arcelli and Gabriella Sanniti Di Baja. Medial Lines and Figure Analysis. In *ICPR 1980: Proceedings of the 5th International Conference on Pattern Recognition*, pages 1016–1018, 1980.

[AS81] Carlo Arcelli and Gabriella Sanniti Di Baja. A Thinning Algorithm Based on Prominence Detection. *Pattern Recognition*, 13(3):225–235, 1981.

[ASM84] Waleed Habib Abdulla, Aladdin O. M. Saleh, and A. H. Morad. A preprocessing algorithm for hand written character recognition. *Pattern Recognition*, 7(1):13–18, 1984.

[BA92] Jonathan Worthen Brandt and V. Ralph Algazi. Continuous skeleton computation by Voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–338, 1992.

[BA94] Gilles Bertrand and Zouina Aktouf. A 3D thinning algorithm using subfields. In *SPIE Proceedings of Conference on Vision Geometry III*, pages 113–124, 1994.

[BC] Gilles Bertrand and Michel Couprie. Three-dimensional parallel thinning algorithms based on critical kernels.

[BC02] Francisco Nivando Bezerra and Michel Couprie. Réduction d'anisotropie des squelettes en niveaux de gris. In *RFIA 2002: Proceedings of 13ème Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle*, volume 3, pages 819–828, 2002.

[BC06] Gilles Bertrand and Michel Couprie. A new 3D parallel thinning scheme based on critical kernels. In *Discrete Geometry for Computer Imagery*, pages 580–591. Springer, 2006.

[BC08] Gilles Bertrand and Michel Couprie. Two-Dimensional Parallel Thinning Algorithms Based on Critical Kernels. *Journal of Mathematical Imaging and Vision*, 31(1):35–56, 2008.

[BCP09] Gilles Bertrand, Michel Couprie, and Nicolas Passat. A note on 3-D simple points and simple-equivalence. *Information Processing Letters*, 109(13):700–704, 2009.

[BDM65] A. E. Brain, Richard O. Duda, and John H. Munson. Graphical data processing research study and experimental investigation. Technical report, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, 1965.

[Bea72] Jacob Bear. *Dynamics of fluids in porous media*. American Elsevier, New York, 1972.

[Ber94] Gilles Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recognition Letters*, 15:1003–1011, 1994.

[Ber95a] Gilles Bertrand. A parallel thinning algorithm for medial surfaces. *Pattern Recognition Letters*, 16(9):979–986, 1995.

[Ber95b] Gilles Bertrand. On P-simple points. *Comptes Rendus de l'Académie des Sciences, Série Mathématiques*, I(321):1077–1084, 1995.

[Ber95c] Gilles Bertrand. Sufficient conditions for 3D parallel thinning algorithms. In R. A. Melter, A. Y. Wu, F. L. Bookstein, and W. D. Green, editors, *SPIE Proceedings of Conference on Vision Geometry IV*, volume 2573 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 52–60, 1995.

[Ber96] Gilles Bertrand. A Boolean characterization of three-dimensional simple points. *Pattern Recognition Letters*, 17(2):115–124, 1996.

[Ber07] Gilles Bertrand. On critical kernels. *Comptes Rendus de l'Académie des Sciences, Série Mathématiques*, I(345):363–367, 2007.

[Beu73] M. Beun. A flexible method for automatic reading of hand written numerals. Technical report, Philips, 1973.

[Bin64] R H Bing. Some aspects of the topology of 3-manifolds related to the Poincaré Conjecture. *Lectures on Modern Mathematics II*, pages 93–128, 1964.

[Blu62] Harry Blum. An Associative Machine for Dealing with the Visual Field and Some of its Biological Implications. In E E Bernard and M R Kare, editors, *Biological Prototypes and Synthetic Systems*, volume 1, pages 244–260, NY, 1962. Plenum Press.

[Blu67] Harry Blum. A transformation for extracting new descriptors of shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[BM92] Serge Beucher and Fernand Meyer. The morphological approach of segmentation: the watershed transformation. In E R Dougherty, editor, *Mathematical Morphology in Image Processing*, pages 433–481. Marcel Dekker, 1992.

[BM94] Gilles Bertrand and Grégoire Malandain. A new characterization of three-dimensional simple points. *Pattern Recognition Letters*, 15(2):169–175, 1994.

[BM99] Thierry M. Bernard and Antoine Manzanera. Improved Low Complexity Fully Parallel Thinning Algorithm. In *ICIAP 1999: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 215, Washington, DC, USA, 1999. IEEE Computer Society.

[BM03] Jasmine Burguet and Rémy Malgouyres. Strong thinning and polyhedric approximation of the surface of a voxel object. *Discrete Applied Mathematics*, 125(1):93–114, 2003.

[BMV10] Raynal Benjamin, Couprie Michel, and Nozick Vincent. Generic Initialization for Motion Capture from 3D Shape. In *Image Analysis and Recognition*, pages 306–315. Springer, 2010.

[BNS99] Gunilla Borgefors, Ingela Nyström, and Gabriella Sanniti Di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

[BNSC05] Dominique Bernard, Ø yvind Nielsen, Luc Salvo, and Peter Cloetens. Permeability assessment by 3D interdendritic flow simulations on microtomography mappings of Al-Cu alloys. *Materials Science and Engineering: A, Structural materials : properties, microstructure and processing.*, 392(1-2):112–120, 2005.

[BPA01] Alexandra Bonnassie, Françoise Peyrin, and Dominique Attali. Shape description of three-dimensional images based on medial axis. In *ICIP 2001: Proceedings of the 8th IEEE International Conference on Image Processing*, volume 3, pages 931–934, Thessaloniki, 2001.

[BRS91] Gunilla Borgefors, Ingemar Ragnemalm, and Gabriella Sanniti Di Baja. The Euclidean distance transform: finding the local maxima and reconstructing the shape. In *SCIA 1991: Proceedings of the 7th Scandinavian Conference on Image Analysis*, volume 2, pages 974–981, 1991.

[BV00] Dominique Bernard and Gérard L. Vignoles. Numerical study of the coupled evolution of micro-geometry and transport properties of simple 3D porous media. In J.-M. Crolet, editor, *Computational Methods for Fluid Flow and Transport in Porous Media*, Theory and Applications of Transport in Porous Media vol. 17, pages 217–226. Springer Verlag, 2000.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[Cal65] L. Calabi. A study of the skeleton of plane figures. Technical Report Technical Report 60429, Parke Mathematical Laboratories, 1965.

[CB97] Michel Couprie and Gilles Bertrand. Topological Grayscale Watershed Transformation. In *Proceedings of SPIE : Vision Geometry V*, volume 3168, pages 136–146, 1997.

[CB08] Michel Couprie and Gilles Bertrand. New characterizations of simple points, minimal non-simple sets and P-simple points in 2D, 3D and 4D discrete spaces. In *DGCI 2008 : Proceedings of the 14th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 4992 of *Lecture Notes in Computer Science*, pages 105–116, Lyon, 2008. Springer-Verlag.

[CB09] Michel Couprie and Gilles Bertrand. New Characterizations of Simple Points in 2D, 3D, and 4D Discrete Spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):637–648, 2009.

[CBB01] Michel Couprie, Francisco Nivando Bezerra, and Gilles Bertrand. Topological operators for grayscale image processing. *Journal of Electronic Imaging*, 10(4):1003–1015, 2001.

[CBC08] John Chaussard, Gilles Bertrand, and Michel Couprie. Characterizing and Detecting Toric Loops in n-Dimensional Discrete Toric Spaces. In *DGCI 2008 : Proceedings of the 14th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 4992 of *Lecture Notes in Computer Science*, pages 129–140, Lyon, 2008. Springer-Verlag.

[CBC10] John Chaussard, Gilles Bertrand, and Michel Couprie. Characterization and Detection of Toric Loops in n-Dimensional Discrete Toric Spaces. *Journal of Mathematical Imaging and Vision*, 36(2):111–124, 2010.

[CBNC10] Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed Cuts: Thinnings, Shortest Path Forests, and Topological Watersheds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:925–939, 2010.

[CC09] John Chaussard and Michel Couprie. Surface Thinning in 3D Cubical Complexes. In *IWCIA 2009: Proceedings of the 13th International Workshop on Combinatorial Image Analysis*, pages 135–148, 2009.

[CCSL09] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. Normal Cone Approximation and Offset Shape Isotopy. *Computational Geometry: Theory and Applications*, 42:566–581, 2009.

[CCT09] John Chaussard, Michel Couprie, and Hugues Talbot. A discrete *lambda*-medial axis. In *DGCI 2009 : Proceedings of the 15th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 5810 of *Lecture Notes in Computer Science*, pages 421–433, Montréal, 2009.

[CCT10] John Chaussard, Michel Couprie, and Hugues Talbot. Robust skeletonization using the discrete lambda-medial axis. *Pattern Recognition Letters*, In Press,, 2010.

[CCZ07] Michel Couprie, David Coeurjolly, and Rita Zrour. Discrete bisector function and Euclidean skeleton in 2D and 3D. *Image and Vision Computing*, 25(10):1543–1556, 2007.

[CH88] Yung-Sheng Chen and Wen-Hsing Hsu. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recognition Letters*, 7(2):99–106, 1988.

[CH89] Yung-Sheng Chen and Wen-Hsing Hsu. A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms. *Pattern Recognition*, 22(3):267–282, 1989.

[Che92] Yung-Sheng Chen. Comments on "A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms". *Pattern Recognition*, 25(12):1545–1546, December 1992.

[CL94] Yian-Leng Chang and Xiaobo Li. Adaptive Image Region-Growing. *IEEE Transactions on Image Processing*, 3(6):868–872, 1994.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH 1996: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 303–312, New York, NY, USA, 1996. ACM.

[CL05] Frédéric Chazal and André Lieutier. The λ-Medial Axis. *Graphical Models*, 67(4):304–331, 2005.

[CM07] David Coeurjolly and Annick Montanvert. Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):437–48, March 2007.

[Coe03] David Coeurjolly. d-dimensional reverse Euclidean distance transformation and Euclidean medial axis extraction in optimal time, 2003.

[Com] Nicolas Combaret. *Modèles de réseaux, pores, et calculs de perméabilité*. PhD thesis, Université de Bordeaux 1.

[Cou06] Michel Couprie. Note on fifteen 2D parallel thinning algorithms, 2006.

[CS86] Yat Keung Chu and Ching Y. Suen. An alternate smoothing and stripping algorithm for thinning digital binary patterns. *Signal Processing*, 11(3):207–222, 1986.

[CWSI87] Roland T. Chin, Hong-Khoon Wan, D. L. Strover, and R. D. Iverson. A one-pass thinning algorithm and its parallel implementation. *Computer Vision, Graphics, and Image Processing*, 40(1):30–40, 1987.

[DC02] Xavier Daragon and Michel Couprie. Segmentation topologique du neo-cortex cérébral depuis des données IRM. In *RFIA 2002: Proceedings of 13ème Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle*, volume 3, pages 809–818, 2002.

[DDL09] Alexandre Dupas, Guillaume Damiand, and Jacques-Olivier Lachaud. Multi-Label Simple Points Definition for 3D Images Digital Deformable Model. In *DGCI 2009 : Proceedings of the 15th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 5810 of *Lecture Notes in Computer Science*, pages 156–167, Montréal, 2009.

[Deu72] Edward S. Deutsch. Thinning algorithms on rectangular, hexagonal, and triangular arrays. *Communications of the ACM*, 15(9):827–837, 1972.

[DJJ94] Marie-Pierre Dubuisson-Jolly and Anil K. Jain. A modified Hausdorff distance for object matching. In *ICPR 1994: Proceedings of the 12th International Conference on Pattern Recognition*, volume 1, pages 566–568, 1994.

[DLPB99] Petr Dokládal, Christophe Lohou, Laurent Perroton, and Gilles Bertrand. Liver Blood Vessels extraction by a 3-D topological approach. In *MICCAI 1999: Proceedings of the 2nd International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 1679–1698, 1999.

[DP81] Edward Roy Davies and A. P. N. Plummer. Thinning algorithms: A critique and a new methodology. *Pattern Recognition*, 14(1-6):53–63, 1981.

[Eck88] Ulrich Eckhardt. A note on Rutovitz' method for parallel thinning. *Pattern Recognition Letters*, 8(1):35–38, 1988.

[Ede61] Murray Eden. A two-dimensional growth process. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probabilities*, volume 4, pages 223–239. Univ. of Calif. Press, 1961.

[EM93] Ulrich Eckhardt and Gerd Maderlechner. Invariant Thinning. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(5):1115–1144, 1993.

[Eul41] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1741.

[FDM06] Timothée Faucon, Etienne Decencière, and Cédric Magneron. Application of Surface Topological Segmentation to Seismic Imaging. In *DGCI 2006 : Proceedings of the 13th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 4245 of *Lecture Notes in Computer Science*, pages 506–517, 2006.

[FJPB09] Anna FabijaÅĎska, Marcin Janaszewski, MichaÅĆ Postolski, and Laurent Babout. Airway Tree Segmentation from CT Scans Using Gradient-Guided 3D Region Growing. In *CIARP 2009: Proceedings of the 14th Iberoamerican Conference on Pattern Recognition*, pages 247–254, Berlin, Heidelberg, 2009. Springer-Verlag.

[FMP+04] Céline Fouard, Grégoire Malandain, Steffen Prohaska, Malte Westerhoff, Francis Cassot, Christophe Mazel, Didier Asselot, and Jean-Pierre Marc-Vergnes. Skeletonization by Blocks for Large 3D Datasets: Application to Brain Microcirculation. In *ISBI 2004: Proceedings of the IEEE International Symposium on Biomedical Imaging*, pages 89–92, Arlington, Virginia, 2004.

[FZBH05] Jianping Fan, Guihua Zeng, Mathurin Body, and Mohand-Said Hacid. Seeded region growing: an extensive and comparative study. *Pattern Recognition Letters*, 26(8):1139–1156, 2005.

[GB90] Weixin Gong and Gilles Bertrand. A simple parallel 3D thinning algorithm. In *ICPR 1990: Proceedings of the 10th International Conference on Pattern Recognition*, pages 188–190. IEEE Comput. Soc. Press, 1990.

[GF96] Yaorong Ge and J. Michael Fitzpatrick. On the generation of skeletons from discrete Euclidean distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1055–1066, 1996.

[GH89] Zicheng Guo and Richard W. Hall. Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373, 1989.

[GH92] Zicheng Guo and Richard W. Hall. Fast fully parallel thinning algorithms. *CVGIP: Image Understanding*, 55(3):317–328, 1992.

[GK03] Chyi-Jou Gau and T. Yung Kong. Minimal non-simple sets in 4D binary images. *Graphical Models*, 65(1-3):112–130, 2003.

[GK04] Peter Giblin and Benjamin B. Kimia. A formal classification of 3D medial axis points and their local geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):238–51, February 2004.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[GN70] G. Gallus and P. W. Neurath. Improved Computer Chromosome Analysis Incorporating Preprocessing and Boundary Analysis. *Physics in Medicine and Biology*, 15(3), 1970.

[Gol69] Marcel J. E. Golay. Hexagonal Parallel Pattern Transformations. *IEEE Transactions on Computers*, 18(8):733–740, 1969.

[Gra06] Leo Grady. Random Walks for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[GS87] V. K. Govindan and A. P. Shivaprasad. A pattern adaptive thinning algorithm. *Pattern Recognition*, 20(6):623–637, 1987.

[Hal89] Richard W. Hall. Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Communications of the ACM*, 32(1):124–131, 1989.

[Hal92] Richard W. Hall. Tests for connectivity preservation for parallel reduction operators. *Topology and its Applications*, 46(3):199–217, 1992.

[Hal93] Richard W. Hall. Optimally Small Operator Supports for Fully Parallel Thinning Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):828–833, 1993.

[Hat02] Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

[Hil69] C. J. Hilditch. Linear Skeletons From Square Cupboards. *Machine Intelligence*, 403, 1969.

[Hil83] C. J. Hilditch. Comparison of thinning algorithms on a parallel processor. *Image and Vision Computing*, 1(3):115–132, 1983.

[Hir96] Tomio Hirata. A unified linear-time algorithm for computing distance maps. *Information Processing Letters*, 58(3):129–133, 1996.

[HP76] Steven L. Horowitz and Theodosios Pavlidis. Picture Segmentation by a Tree Traversal Algorithm. *Journal of the ACM*, 23(2):368–388, April 1976.

[HR08] Wim H. Hesselink and Jos B. T. M. Roerdink. Euclidean Skeletons of Digital Image and Volume Data in Linear Time by the Integer Medial Axis Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2204–2217, 2008.

[HSCP87] Christopher M. Holt, Alan Stewart, Maurice Clint, and Ronald H. Perrott. An improved parallel thinning algorithm. *Communications of the ACM*, 30(2):156–160, 1987.

[Hul09] Jérôme Hulin. *Axe Médian Discret : Propriétés Arithmétiques et Algorithmes*. PhD thesis, Laboratoire d'Informatique Fondamentale de Marseille, 2009.

[HW07] Brian Hopkins and Robin Wilson. The truth about Königsberg. In Robert E Bradley and C Edward Sandifer, editors, *Leonhard Euler: Life, Work and Legacy*, volume 5 of *Studies in the History and Philosophy of Mathematics*, pages 409–420. Elsevier, 2007.

[Izu74] M. I. Izutsdkiver. Algorithm for the initial processing of an ensemble of symbols in the recognition process. *Automation and Remote Control*, 35(8):1292–1298, 1974.

[JAB+10] Rachid Jennane, Gabriel Aufort, Claude Benhamou, Murat Ceylan, Yüksel Özbay, and Osman Ucan. A New Method for 3D Thinning of Hybrid Shaped Porous Media Using Artificial Intelligence. Application to Trabecular Bone. *Journal of Medical Systems*, pages 1–14, 2010.

[JBC07] Tao Ju, Matthew L. Baker, and Wah Chiu. Computing a family of skeletons of volumetric models for shape description. *Computer Aided Design*, 39(5):352–360, May 2007.

[JC92] Ben-Kwei Jang and Roland T. Chin. One-Pass Parallel Thinning: Analysis, Properties, and Quantitative Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1129–1140, 1992.

[JC93] Ben-Kwei Jang and Roland T. Chin. Reconstructable Parallel Thinning. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:1145–1181, 1993.

[JCB10] Marcin Janaszewski, Michel Couprie, and Laurent Babout. Hole filling in 3D volumetric objects. *Pattern Recognition*, 43(10):3548–3559, 2010.

[KG04] T. Yung Kong and Chyi-Jou Gau. Minimal Non-simple Sets in 4-Dimensional Binary Images with (8,80)-Adjacency. In *IWCIA 2004: Proceedings of the 10th International Workshop on Combinatorial Image Analysis*, volume 3322 of *Lecture Notes in Computer Science*, pages 318–333, 2004.

[KK88] K. Kedem and D. Keret. A fast algorithm for skeletonizing lines by midline technique. In *ICSC 1988: Proceedings of the 1st International Computer Science Conference*, pages 731–735, 1988.

[Kle06] Gisela Klette. Voxels and Junctions in 3D skeletons. In *IWCIA 2006: Proceedings of the 11th International Workshop on Combinatorial Image Analysis*, pages 34–44, 2006.

[KLR90] T. Yung Kong, Richard A. Litherland, and Azriel Rosenfeld. *Problems in the topology of binary digital images*, pages 376–385. Elsevier, 1990.

[KNP09] Péter Kardos, Gábor Németh, and Kálmán Palágyi. An Order-Independent Sequential Thinning Algorithm. In *IWCIA 2009: Proceedings of the 13th International Workshop on Combinatorial Image Analysis*, volume 5852 of *Lecture Notes in Computer Science*, pages 162–175. Springer Verlag, 2009.

[Kon89] T. Yung Kong. A digital fundamental group. *Computers and Graphics*, 13(2):159–166, 1989.

[Kon93] T. Y. Kong. Problem of determining whether a parallel reduction operator for n-dimensional binary images always preserves topology. In R.A. Melter and A. Y. Wu, editors, *SPIE Proceedings of Conference on Vision Geometry II*, volume 2060 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 69–77, 1993.

[Kon95] T. Yung Kong. On Topology Preservation in 2-D and 3-D Thinning. *International Journal of Pattern Recognition and Artificial Intelligence*, 9:813–844, 1995.

[Kon97] T. Yung Kong. Topology-Preserving Deletion of 1's from 2-, 3- and 4-Dimensional Binary Images. In *DGCI 1997 : Proceedings of the 7th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 1347 of *Lectures Notes in Computer Science*, pages 3–18, London, UK, 1997. Springer-Verlag.

[KR89] T. Yung Kong and Azriel Rosenfeld. Digital topology: introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48(3):357–393, 1989.

[KRR92] T. Yung Kong, A. W. Roscoe, and Azriel Rosenfeld. Concepts of digital topology. *Topology and its Applications*, 46(3):219–262, 1992.

[KSKB95] Ron Kimmel, Doron Shaked, Nahum Kiryati, and Alfred M. Bruckstein. Skeletonization via Distance Maps and Level Sets. *Computer Vision and Image Understanding*, 62:382–391, 1995.

[LB04] Christophe Lohou and Gilles Bertrand. A 3D 12-subiteration thinning algorithm based on P-simple points. *Discrete Applied Mathematics*, 139(1-3):171–195, 2004.

[LB05] Christophe Lohou and Gilles Bertrand. A 3D 6-subiteration curve thinning algorithm based on P-simple points. *Discrete Applied Mathematics*, 151(1-3):198–228, 2005.

[LC94] Thomas C. M. Lee and Richard Cowan. A Stochastic Tesselation of Digital Space. In *ISMM 1994: Proceedings of the 2nd International Symposium on Mathematical Morphology*, pages 218–224, Fontainebleau, 1994. Kluwer.

[LCLJ10] Lu Lui, Erin Wolf Chambers, David Letscher, and Tao Ju. A simple and robust thinning algorithm on cell complexes. In *Computer Graphics Forum (Proceedings of Pacific Graphics 2010)*, 2010.

[Lie03] André Lieutier. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. In *Proceedings of the 8th ACM Symposium on Solid Modeling Applications*, pages 65–75. Academic Press, 2003.

[Lis47] Johann Benedikt Listing. *Vorstudien zur Topologie*. Vandenhoeck und Ruprecht, Göttingen, 1847.

[LKC94] Ta-Chih Lee, Rangasami L. Kashyap, and Chong-Nam Chu. Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, November 1994.

[LLS92] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning Methodologies-A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:869–885, 1992.

[Loho1] Christophe Lohou. *Contribution à l'analyse topologique des images : Étude d'algorithmes de squelettisation pour images 2D et 3D selon une approche topologie digitale ou topologie discrète*. PhD thesis, Université de Marne-la-Vallée, France, 2001.

[Loho8] Christophe Lohou. Detection of the non-topology preservation of Ma's 3D surface-thinning algorithm, by the use of P-simple points. *Pattern Recognition Letters*, 29(6):822–827, 2008.

[Loh10] Christophe Lohou. Detection of the non-topology preservation of Ma and Sonka's algorithm, by the use of P-simple points. *Computer Vision and Image Understanding*, 114(3):384–399, 2010.

[LV01] Jacques-Olivier Lachaud and Anne Vialard. Discrete Deformable Boundaries for the Segmentation of Multidimensional Images. In Carlo Arcelli, Luigi Cordella, and Gabriella di Baja, editors, *Visual Form 2001 : Proceedings of the 4th International Workshop on Visual Form*, volume 2059 of *Lecture Notes in Computer Science*, pages 542–551. Springer Berlin / Heidelberg, 2001.

[LW86] H. E. Lü and Patrick Shen-Pei Wang. A comment on "A fast parallel algorithm for thinning digital patterns". *Communications of the ACM*, 29(3):239–242, 1986.

[Ma94] Cherng-Min Ma. On topology preservation in 3D thinning. *CVGIP: Image Understanding*, 59(3):328–339, 1994.

[Ma95] Cherng-Min Ma. A 3D fully parallel thinning algorithm for generating medial faces. *Pattern Recognition Letters*, 16(1):83–87, 1995.

[Ma00] Cherng-Min Ma. Parallel Thinning Algorithms on 3D (18, 6) Binary Images. *Computer Vision and Image Understanding*, 80(3):364–378, December 2000.

[Mal00] Rémy Malgouyres. Homotopy in two-dimensional digital images. *Theoretical Computer Science*, 230(1-2):221–233, 2000.

[Mal01] Rémy Malgouyres. Computing the Fundamental Group in Digital Spaces. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(7):1053–1074, 2001.

[Mar60] Andreï Andreyevich Markov. Insolubility of the problem of homeomorphy. In *Proceedings of the International Congress of Mathematics of 1958*, pages 300–306. Cambridge University Press, 1960.

[Mat88] Georges Matheron. *Examples of topological properties of skeletons*, volume 2, pages 217–238. Academic Press, 1988.

[Mau96] Charles Richard Francis Maunder. *Algebraic Topology*. Dover Publications, 1996.

[MB92] Grégoire Malandain and Gilles Bertrand. Fast characterization of 3D simple points. In *ICPR 1992: Proceedings of the 11th International Conference on Pattern Recognition*, volume 3, pages 232–235, The Hague, The Netherlands, 1992.

[MBA93] Grégoire Malandain, Gilles Bertrand, and Nicholas Ayache. Topological segmentation of discrete surfaces. *International Journal of Computer Vision*, 10(2):183–197, 1993.

[MBPL99] Antoine Manzanera, Thierry M. Bernard, Françoise J. Prêteux, and Bernard Longuet. Medial faces from a Concise 3D thinning algorithm. *ICCV 1999: Proceedings of the 7th IEEE International Conference on Computer Vision*, 1:337, 1999.

[MDC90] Jayanta Mukherjee, Partha Pratim Das, and Biswanath N. Chatterji. On connectivity issues of ESPTA. *Pattern Recognition Letters*, 11(9):643–648, September 1990.

[Men96] Yann Menguy. *Optimisation quadratique et géométrique de problèmes de dosimétrie inverse*. PhD thesis, Université Joseph-Fourier - Grenoble I, 1996.

[Mey79] Fernand Meyer. *Cytologie quantitative et morphologie mathématique*. PhD thesis, École des Mines de Paris, France, 1979.

[MF08] Rémy Malgouyres and Angel R. Francés. Determining whether a simplicial 3-complex collapses to a 1-complex is NP-complete. In *DGCI 2008 : Proceedings of the 14th IAPR International Conference on*

Discrete Geometry for Computer Imagery, volume 4992 of *Lecture Notes in Computer Science*, pages 177–188, Lyon, 2008. Springer-Verlag.

[MFV98] Grégoire Malandain and Sara Fernández-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16(5):317–327, 1998.

[MN01] Pavel Mrazek and Mirko Navara. Consistent Positive Directional Splitting of Anisotropic Diffusion. In *Computer Vision Winter Workshop 2001*, 2001.

[Mor81] David George Morgenthaler. Three-dimensional simple points: serial erosion, parallel thinning and skeletonization. Technical Report TR-1005, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, 1981.

[MQR03] Calvin R. Jr. Maurer, Rensheng Qi, and Vijay V. Raghavan. A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[MRH00] Arnold Meijster, Jos B. T. M. Roerdink, and Wim H. Hesselink. *A General Algorithm for Computing Distance Transforms in Linear Time*, pages 331–340. Kluwer, 2000.

[MS96] Cherng-Min Ma and Milan Sonka. A fully parallel 3D thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64(3):420–433, 1996.

[MU74] Ivaturi S. N. Murthy and Jayaram K. Udupa. A Search Algorithm for Skeletonization of Thick Patterns. *Computer Graphics and Image Processing*, 3(3):246–259, 1974.

[MWL02] Cherng-Min Ma, Shu-Yen Wan, and Jiann-Der Lee. Three-dimensional topology preserving reduction on the 4-subfields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1594–1605, December 2002.

[NKP10a] Gábor Németh, Péter Kardos, and Kálmán Palágyi. Topology preserving 2-subfield 3D thinning algorithms. In *SPPRA 2010: Proceedings of the 7th IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, pages 310–316, Innsbruck, 2010. IASTED.

[NKP10b] Gábor Németh, Péter Kardos, and Kálmán Palágyi. Topology Preserving 3D Thinning Algorithms Using Four and Eight Subfields. In Aurélio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, volume 6111 of *Lecture Notes in Computer Science*, pages 316–325. Springer Berlin / Heidelberg, 2010.

[NP09] Gábor Németh and Kálmán Palágyi. Parallel Thinning Algorithms Based on Ronse's Sufficient Conditions for Topology Preservation. *Progress in Combinatorial Image Analysis*, pages 183–194, 2009.

[NS84] Nabil Jean Naccache and Rajjan Shinghal. SPTA: A Proposed Algorithm for Thinning Binary Patterns. *IEEE Transactions on Systems, Man and Cybernetics*, 14:409–418, 1984.

[NSK+97] Martin Näf, Gábor Székely, Ron Kikinis, Martha Elizabeth Shenton, and Olaf Kübler. 3d Voronoi skeletons and their usage for the characterization and recognition of 3D organ shape. *Computer Vision and Image Understanding*, 66(2):147–161, 1997.

[OK95] Robert L. Ogniewicz and Olaf Kübler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28(33):343–359, 1995.

[OP03] Stanley J. Osher and Nikos Paragios. *Geometric Level Set Methods in Imaging,Vision,and Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[OS88] Stanley J. Osher and James A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[Pal02] Kálmán Palágyi. A 3-subiteration 3D thinning algorithm for extracting medial surfaces. *Pattern Recognition Letters*, 23(6):663–675, 2002.

[Pal07a] Kálmán Palágyi. A 3-subiteration surface-thinning algorithm. In *CAIP 2007: Proceedings of the 12th International Conference on Computer Analysis of Images and Patterns*, CAIP'07, pages 628–635, Berlin, Heidelberg, 2007. Springer-Verlag.

[Pal07b] Kálmán Palágyi. A Subiteration-Based Surface-Thinning Algorithm with a Period of Three. In Fred Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 294–303. Springer Berlin / Heidelberg, 2007.

[Palo8] Kálmán Palágyi. A 3D fully parallel surface-thinning algorithm. *Theoretical Computer Science*, 406(1-2):119–135, 2008.

[Pav8o] Theodosios Pavlidis. A thinning algorithm for discrete binary images. *Computer Graphics and Image Processing*, 13(2):142–157, 1980.

[Pav81] Theodosios Pavlidis. A Flexible Parallel Thinning Algorithm. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 162–167, 1981.

[Pav82a] Theodosios Pavlidis. *Algorithms for Graphics and Image Processing*. Springer, 1982.

[Pav82b] Theodosios Pavlidis. An Asynchronous Thinning Algorithm. *Computer Graphics and Image Processing*, 20(2):133–157, 1982.

[PCB08] Nicolas Passat, Michel Couprie, and Gilles Bertrand. Minimal Simple Pairs in the 3-D Cubic Grid. *Journal of Mathematical Imaging and Vision*, 32(3):239–249, 2008.

[PK98] Kálmán Palágyi and Attila Kuba. A 3D 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, 19(7):613–627, 1998.

[PK99a] Kálmán Palágyi and Attila Kuba. A Parallel 3D 12-Subiteration Thinning Algorithm. *Graphical Models and Image Processing*, 61(4):199–221, July 1999.

[PK99b] Kálmán Palágyi and Attila Kuba. Directional 3D Thinning Using 8 Subiterations. In Gilles Bertrand, Michel Couprie, and Laurent Perroton, editors, *Discrete Geometry for Computer Imagery*, volume 1568 of *Lecture Notes in Computer Science*, pages 325–336. Springer Berlin / Heidelberg, 1999.

[PL90] Theodosios Pavlidis and Yuh-Tay Liow. Integrating Region Growing and Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, 1990.

[Plo09] Erwan Plougonven. *Lien entre la microstructure des matériaux poreux et leur perméabilité : mise en évidence des paramètres géométriques et topologiques influant sur les propriétés de transport par analyses d'images microtomographiques*. PhD thesis, Université Sciences et Technologies - Bordeaux I, 2009.

[PM90] Pietro Perona and Jitendra Malik. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[Poi95] Henri Poincaré. Analysis Situs. *Journal de l'École Polytechnique, 2ème série, cahier 1*, pages 1–121, 1895.

[PTHS06] Kálmán Palágyi, Juerg Tschirren, Eric A. Hoffman, and Milan Sonka. Quantitative analysis of pulmonary airway tree structure. *Computers in Biology and Medicine*, 36:974–996, 2006.

[Pud98] Chris Pudney. Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images. *Computer Vision and Image Understanding*, 72(3):404–413, 1998.

[RCC90] Serge Riazanoff, Bernard Cervelle, and Jean Chorowicz. Parametrisable Skeletonization of Binary and Multilevel Images. *Pattern Recognition Letters*, 11:25–33, 1990.

[Ron86] Christian Ronse. A topological characterization of thinning. *Theoretical Computer Science*, 43(1):31–41, 1986.

[Ron88] Christian Ronse. Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images. *Discrete Applied Mathematics*, 21(1):67–79, 1988.

[Ros70] Azriel Rosenfeld. Connectivity in Digital Pictures. *Journal of the ACM*, 17(1):146–160, 1970.

[Ros73] Azriel Rosenfeld. Arcs and Curves in Digital Pictures. *Journal of the ACM*, 20(1):81–87, 1973.

[Ros75] Azriel Rosenfeld. A characterization of parallel thinning algorithms. *Information and Control*, 29(3):286–291, 1975.

[Ros79] Azriel Rosenfeld. Digital topology. *The American Mathematical Monthly*, 86(8):621–630, 1979.

[Ros81] Azriel Rosenfeld. Three-dimensional digital topology. *Information and Control*, 50(2):119–127, 1981.

[RS02] Vincent Ranwez and Pierre Soille. Order independent homotopic thinning for binary and grey tone anchored skeletons. *Pattern Recognition Letters*, 23(6):687–702, 2002.

[RT02a] Éric Rémy and Édouard Thiel. Medial axis for chamfer distances: computing look-up tables and neighbourhoods in 2D or 3D. *Pattern Recognition Letters*, 23(6):649–661, April 2002.

[RT02b] Martin Rumpf and Alexandru Telea. A continuous skeletonization method based on level sets. In *VisSym 2002: Proceedings of the Symposium on Data Visualisation*, pages 151—-ff, Aire-la-Ville, Switzerland,

Switzerland, 2002. Eurographics Association.

[RT03] Éric Rémy and Édouard Thiel. Look-Up Tables for Medial Axis on Squared Euclidean Distance Transform. In *DGCI 2003 : Proceedings of the 11th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 2886 of *Lectures Notes in Computer Science*, pages 224–235, 2003.

[RT05] Éric Rémy and Édouard Thiel. Exact Medial Axis with Euclidean Distance. *Image and Vision Computing*, 23(2):167–175, 2005.

[RT08] Dennie Reniers and Alexandru Telea. Segmenting simplified surface skeletons. In *DGCI 2008 : Proceedings of the 14th IAPR International Conference on Discrete Geometry for Computer Imagery*, Lecture Notes in Computer Science, pages 262–274, Lyon, 2008.

[Rut66] Denis Rutovitz. Pattern Recognition. *Journal of the Royal Statistical Society*, 129(4):504–530, 1966.

[SAAY06] Marie Samozino, Marc Alexa, Pierre Alliez, and Mariette Yvinec. Reconstruction with Voronoi Centered Radial Basis Functions. In *SGP 2006: Proceedings of the 4th Symposium on Geometry Processing (SGP'06)*, pages 51–60, 2006.

[SBTZ99] Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, and Steven W. Zucker. The Hamilton-Jacobi skeleton. In *ICCV 1999: Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 828–834, 1999.

[SC94] Punam Kumar Saha and Bidyut Baran Chaudhuri. Detection of 3-D Simple Points for Topology Preserving Transformations with Application to Thinning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1028–1032, 1994.

[SCCM94] Punam Kumar Saha, Bidyut Baran Chaudhuri, Bhabatosh Chanda, and D. Dutta Majumder. Topology preservation in 3D digital space. *Pattern Recognition*, 27(2):295–300, 1994.

[SCD97] Punam Kumar Saha, Bidyut Baran Chaudhuri, and Dwijesh Dutta Majumder. A new shape preserving parallel thinning algorithm for 3D digital images. *Pattern Recognition*, 30(12):1939–1955, December 1997.

[SCdo6] André Vital Saúde, Michel Couprie, and Roberto de Alencar Lotufo. Exact Euclidean Medial Axis in Higher Resolution. In *DGCI 2006 : Proceedings of the 13th IAPR International Conference on Discrete Geometry for Computer Imagery*, volume 4245 of *Lecture Notes in Computer Science*, pages 605–616, 2006.

[Sch74] Adrian E. Scheidegger. *The Physics of Flow Through Porous Media*. University of Toronto Press, Toronto, 1974.

[SCTK98] Daniel Sharvit, Jacky Chan, Hüseyin Tek, and Benjamin B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, 1998.

[Ser82] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[SM93] Michel Schmitt and Juliette Mattioli. *Morphologie mathématique*. Collection logique mathématiques informatique. Masson, 1993.

[Soi99] Pierre Soille. *Morphological Image Analysis*. Springer-Verlag, 1999.

[Sos89] Juan Humberto Sossa Azuela. An Improved Parallel Algorithm for Thinning Digital Patterns. *Pattern Recognition Letters*, 10:77–80, 1989.

[SR71] Renato Stefanelli and Azriel Rosenfeld. Some Parallel Thinning Algorithms for Digital Pictures. *Journal of the ACM*, 18(2):255–264, 1971.

[SS01] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[ST96] Gabriella Sanniti Di Baja and Édouard Thiel. Skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14(1):47–57, 1996.

[Sti80] John Stillwell. *Classical Topology and Combinatorial Group Theory*. Springer, 1980.

[Str05] Robin Strand. A classification of centres of maximal balls in Z^3. In *SCIA 2005: Proceedings of the 14th Scandinavian Conference on Image Analysis*, volume 3540 of *Lecture Notes in Computer Science*, pages 1057–1065, Joensuu, 2005. Springer.

[SU79] S. N. Srihari and Jayaram K. Udupa. Understanding the bin of parts. In *Proceedings of the International Conference on Cybernetics and Society*, pages 44–49, 1979.

[SW94] Frank Y. Shih and Wai-Tak Wong. Fully Parallel Thinning With Tolerance To Boundary Noise. *Pattern Recognition*, 27(12):1677–1695, 1994.

[Tam78] H. Tamura. A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint. In *ICPR 1978: Proceedings of the 4th International Conference on Pattern Recognition*, pages 715–719, 1978.

[Tan09] Martin Tancer. d-collapsibility is NP-complete for $d \geqslant 4$. *Electronic Notes in Discrete Mathematics*, 34:53–57, 2009.

[TF81a] Y. F. Tsao and K. S. Fu. A parallel thinning algorithm for 3-D pictures. *Computer Graphics and Image Processing*, 17(4):315–331, 1981.

[TF81b] Y. F. Tsao and K. S. Fu. Parallel Thinning Operations for Digital Binary Images. In *Pattern Recognition and Image Processing*, pages 150–155, 1981.

[TM01] Jun-ichiro Toriwaki and Kensaku Mori. Distance Transformation and Skeletonization of 3D Pictures and Their Applications to Medical Images. In *Digital and Image Geometry*, Lecture Notes in Computer Science, pages 412–428, London, UK, 2001. Springer-Verlag.

[TTDP09] Olena Tankyevych, Hugues Talbot, Petr Dokládal, and Nicolas Passat. Direction-adaptive grey-level morphology. application to 3D vascular brain imaging. In *ICIP 2009: Proceedings of the 16th IEEE International Conference on Image Processing*, pages 2237–2240, Piscataway, NJ, USA, 2009. IEEE Press.

[TV92] Hugues Talbot and Luc M. Vincent. Euclidean skeletons and conditional bisectors. In Petros Maragos, editor, *VCIP 1992: Proceedings of the International Conference on Visual Communications and Image Processing*, volume 1818, pages 862–876. SPIE, 1992.

[VCAB07] Gérard L. Vignoles, Olivia Coindreau, Azita Ahmadi, and Dominique Bernard. Assessment of geometrical and transport properties of a fibrous C/C composite preform as digitized by X-ray computerized microtomography: Part II. Heat and gas transport properties. *Journal of Materials Research*, 22(6):1537–1550, 2007.

[Vin91] Luc M. Vincent. Efficient computation of various types of skeletons. In Murray H Loew, editor, *Proceedings of SPIE : Medical Imaging V: Image Processing*, volume 1445, pages 297–311. SPIE, 1991.

[WB07] Tao Wang and Anup Basu. A note on 'A fully parallel 3D thinning algorithm and its applications'. *Pattern Recognition Letters*, 28(4):501–506, 2007.

[Wel91] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, 1991.

[WHF86] Patrick Shen-Pei Wang, L. W. Hui, and T. Fleming. Further improved fast parallel thinning algorithm for digital patterns. *Computer Vision, Image Processing and Communications - Systems and Applications*, pages 37–40, 1986.

[WL93] Zhenyu Wu and Richard Leahy. An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[WT92] Rei-Yao Wu and Wen-Hsiang Tsai. A new one-pass parallel thinning algorithm for binary images. *Pattern Recognition Letters*, 13(10):715–723, 1992.

[XTP03] Wenjie Xie, Robert P. Thompson, and Renato Perucchio. A topology-preserving parallel 3D thinning algorithm for extracting the curve skeleton. *Pattern Recognition*, 36(7):1529–1544, July 2003.

[YTF75] Shigeki Yokoi, Jun-Ichiro Toriwaki, and Teruo Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing*, 4(1):63–73, 1975.

[Zah71] Charles T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.

[ZCMO96] Hong-Kai Zhao, Tony F. Chan, Barry Merriman, and Stanley J. Osher. A Variational Level Set Approach to Multiphase Motion, 1996.

[ZS84] T. Y. Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS