

Parallel skeletonization algorithms in the cubic grid based on critical kernels

Gilles Bertrand, Michel Couprie

▶ **To cite this version:**

Gilles Bertrand, Michel Couprie. Parallel skeletonization algorithms in the cubic grid based on critical kernels. Punam Saha Gunilla Borgefors Gabriella Sanniti di Baja. Skeletonization: Theory, Methods and Applications, Elsevier, pp.181-210, 2017, Skeletonization: Theory, Methods and Applications, 9780081012925. hal-01613008

HAL Id: hal-01613008

https://enpc.hal.science/hal-01613008

Submitted on 9 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parallel skeletonization algorithms in the cubic grid based on critical kernels

Gilles Bertrand, Michel Couprie

Université Paris-Est, LIGM, Équipe A3SI, ESIEE Paris, France
e-mail: gilles.bertrand@esiee.fr, michel.couprie@esiee.fr

Abstract. We propose two compact and generic thinning schemes, both based on the critical kernels framework, which encompass an enormous variety of homotopic parallel 3D thinning algorithms—including asymmetric and symmetric algorithms that produce ultimate, curve, surface, and other kinds of skeletons. Our algorithms are fast and our approach is validated experimentally. We also propose an effective filtering strategy based on the notion of isthmus persistence that can be easily performed within our framework.

1 Introduction

Parallel thinning in discrete grids is a topic that has been studied since the pioneering years of digital image processing. The most essential requirement for a skeletonization method is topology preservation, hence the abundant literature devoted to the study of conditions under which a thinning method meets this requirement. Among these works, the notions of minimal non-simple set [1], P-simple point [2] and critical kernel [3] constitute major contributions to a systematic study of topology preservation in parallel thinning. Critical kernels provide, to our knowledge, the most general framework for the design of topologically sound thinning algorithms. In particular, we proved in [4] that the notions of minimal non-simple sets and P-simple points can be characterized in terms of critical kernels.

An important distinction must be made among parallel thinning algorithms, between symmetric and asymmetric algorithms. Symmetric algorithms [5–7], on one hand, are the ones that produce skeletons which are uniquely defined and invariant under 90 degrees rotations. Such skeletons may contain parts that are not as thin as possible, like two-voxel wide ribbons for example. On the other hand, asymmetric algorithms [8–17] produce thinner skeletons, but the price to pay for the improved thinness is the loss of 90 degrees rotations invariance. Both kinds of thinning algorithms are useful in applications. Choosing between a symmetric and an asymmetric algorithm is a matter of context.

Let us outline briefly and informally the framework that we will detail in section 3, derived from the notion of critical kernel. First of all, instead of considering single voxels, we consider cliques. A clique is a set of mutually adjacent voxels. In a voxel set X, certain cliques called critical cliques cannot be removed

from X without altering its topological characteristics. The critical cliques of X can be identified using local conditions. The main theorem of the critical kernels framework [3, 18] implies that any subset of the object X can be removed in parallel, provided that at least one voxel of every critical clique is preserved, and this guarantees topology preservation.

Since each clique can contain between one and eight voxels, there usually exists a huge number of combinations for the choice of the voxels that are kept during one step of thinning.

Consider for example the case of a critical clique made of two voxels a, b. A first possibility is to keep both of them, thus avoiding to make an arbitrary choice between a and b. Such a strategy applied to all the critical cliques, leads to symmetric thinning algorithms,

On the other hand, if we prefer thinner skeletons, we may use a systematic rule to decide which voxel among a and b will be removed. This leads to a so-called asymmetric thinning algorithm.

In addition, various constraints can easily be added in order to preserve some voxels having certain geometrical properties, to obtain curve or surface skeletons for example. The notions of 1-isthmus and 2-isthmus (section 5.2) will allow us to detect such voxels.

In this chapter, after providing the necessary background notions in section 2 and recalling the notion of critical clique in section 3, we discuss in section 4 a general strategy to design parallel thinning algorithms in this framework. Then, we propose an asymmetric (section 5) and a symmetric (section 6) generic thinning scheme. These schemes can be used to obtain different kinds of skeletons, for example ultimate, curve or surface skeletons, by choosing appropriate parameters.

It is well known that skeletons are highly sensitive to contour noise, that is, a small perturbation of the contour may provoke the appearing or the disappearing of an arbitrarily long skeleton branch. In section 8, we propose a natural extension of our thinning methods that copes with the robustness to noise issue. This extension is based on a notion of "isthmus persistence" [19], which has been introduced in the framework of arbitrary 3D complexes (objects that are not necessarily made of voxels). Isthmus persistence is natural in our framework, and could not be adapted to other approaches based, *e.g.*, on end voxel detection.

We also propose in section 9 a way to efficiently compute a hierarchy of nested filtered skeletons, corresponding to the different values of isthmus persistence.

We provide in section 7 some local characterizations that allow for the effective implementation of our methods. All the algorithms proposed in this chapter can be implemented to run in linear time complexity (section 10).

This chapter is essentially based on the contents of the papers [20, 21, 18]. However, it offers a new perspective on these works, thanks to a compact, synthetic and self-contained presentation.

2 Voxel Complexes and simple voxels

Let us first give some basic definitions for voxel complexes, see also [22, 23]. Let $\mathbb{Z}$ be the set of integers. We consider the families of sets $\mathbb{F}_0^1$, $\mathbb{F}_1^1$, such that $\mathbb{F}_0^1 = \{\{a\} \mid a \in \mathbb{Z}\}$, $\mathbb{F}_1^1 = \{\{a, a+1\} \mid a \in \mathbb{Z}\}$. A subset f of $\mathbb{Z}^n$, $n \geq 2$, that is the Cartesian product of exactly d elements of $\mathbb{F}_1^1$ and $(n-d)$ elements of $\mathbb{F}_0^1$ is called a *face* or an *d-face* of $\mathbb{Z}^n$, d is the *dimension of* f, and we write $dim(f) = d$.

In the illustrations of this chapter, unless explicitely stated, a 3-face (resp. 2-face, 1-face, 0-face) is depicted by a cube (resp. square, segment, dot), see *e.g.* figure 3.

A 3-face of $\mathbb{Z}^3$ is also called a *voxel*. A finite set that is composed solely of voxels is called a *(voxel) complex* (see figure 1). We denote by $\mathbb{V}^3$ the collection of all voxel complexes.

Notice that, by identifying each voxel with its center of mass, we get an equivalence between the data of a voxel complex and the one of a subset of $\mathbb{Z}^3$.

We say that two voxels x, y are *adjacent* if $x \cap y \neq \emptyset$. We write $\mathcal{N}(x)$ for the set of all voxels that are adjacent to a voxel x, $\mathcal{N}(x)$ is the *neighborhood of* x. Note that, for each voxel x, we have $x \in \mathcal{N}(x)$. We set $\mathcal{N}^*(x) = \mathcal{N}(x) \setminus \{x\}$.

Let $d \in \{0, 1, 2\}$. We say that two voxels x, y are *d-neighbors* if $x \cap y$ is a d-face. Thus, two distinct voxels x and y are adjacent if and only if they are d-neighbors for some $d \in \{0, 1, 2\}$. For example in figure 1, voxels b and c are 2-neighbors, voxels a and b are 1-neighbors but not 2-neighbors, voxels b and d are 0-neighbors but neither 2-neighbors nor 1-neighbors.

Let $X \in \mathbb{V}^3$. We say that X is *connected* if, for any $x, y \in X$, there exists a sequence $\langle x_0, ..., x_k \rangle$ of voxels in X such that $x_0 = x$, $x_k = y$, and x_i is adjacent to x_{i-1}, $i = 1, ..., k$.

Intuitively, a voxel x of a complex X is called a simple voxel if its removal from X "does not change the topology of X". This notion may be formalized with the help of the following recursive definition introduced in [18], see also [24, 25] for other recursive approaches for the notion of simple point.

Definition 1. Let $X \in \mathbb{V}^3$. We say that X is *reducible* if either:
i) X is composed of a single voxel; or
ii) there exists $x \in X$ such that $\mathcal{N}^*(x) \cap X$ is reducible and $X \setminus \{x\}$ is reducible.

Definition 2. Let $X \in \mathbb{V}^3$. A voxel $x \in X$ is *simple for* X if $\mathcal{N}^*(x) \cap X$ is reducible. If $x \in X$ is simple for X, we say that $X \setminus \{x\}$ is an *elementary thinning of* X.

Thus, a complex $X \in \mathbb{V}^3$ is reducible if and only if it is possible to reduce X to a single voxel by iteratively removing simple voxels. Observe that a reducible complex is necessarily non-empty and connected.

In figure 1 (left), the voxel a is simple for X ($\mathcal{N}^*(a) \cap X = \{b\}$ is made of a single voxel), the voxel d is not simple for X ($\mathcal{N}^*(d) \cap X = \{c, e, f, g\}$ is not connected), the voxel h is simple for X ($\mathcal{N}^*(h) \cap X = \{f, g\}$ is made of two voxels that are 2-neighbors and is reducible).

In [18], it was shown that the above definition of a simple voxel is equivalent to classical characterizations based on connectivity properties of the voxel's neighborhood [26–30]. An equivalence was also established with a definition based on the operation of collapse [31], this operation is a discrete analogue of a continuous deformation (a homotopy), see [24, 3, 30].

The notion of a simple voxel allows one to define thinnings of a complex, see an illustration figure 1 (right).

Let $X, Y \in \mathbb{V}^3$. We say that Y *is a thinning of* X or that X is *reducible to* Y, if there exists a sequence $\langle X_0, ..., X_k \rangle$ such that $X_0 = X$, $X_k = Y$, and X_i is an elementary thinning of X_{i-1}, $i = 1, ..., k$. Thus, a complex X is reducible if and only if it is reducible to a set made of a single voxel.

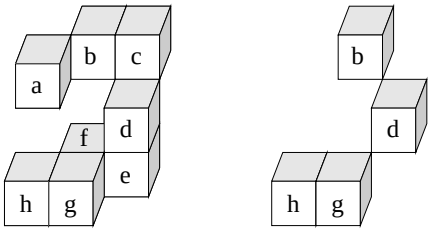


Fig. 1. Left: a complex X which is made of 8 voxels, Right: A complex $Y \subseteq X$, which is a thinning of X.

3 Critical cliques

Let X be a voxel complex. It is well known that, if we remove simultaneously (in parallel) simple voxels from X, we may "change the topology" of the original object X. For example, the two voxels f and g are simple for the object X depicted figure 1 (left). Nevertheless $X \setminus \{f, g\}$ has two connected components whereas X is connected.

In this section, we recall a framework for thinning in parallel discrete objects with the warranty that we do not alter the topology of these objects [3, 32, 18].

Let $d \in \{0, 1, 2, 3\}$ and let $C \in \mathbb{V}^3$. We say that C is a *d-clique* or a *clique* if $\cap\{x \in C\}$ is a d-face. If C is a d-clique, we say that d is the *rank of* C.

For example in figure 1, $\{d, e, f, g\}$ is a 0-clique, $\{f, g, e\}$ is a 1-clique, $\{b, c\}$ is a 2-clique, $\{a\}$ is a 3-clique.

If C is made of solely two distinct voxels x and y, we note that C is a d-clique if and only if x and y are d-neighbors, with $d \in \{0, 1, 2\}$.

Let $X \in \mathbb{V}^3$ and let $C \subseteq X$ be a clique. We say that C is *essential for* X if we have $C = D$ whenever D is a clique such that:
i) $C \subseteq D \subseteq X$; and
ii) $\cap\{x \in C\} = \cap\{x \in D\}$.

In other words, C is essential for X if it is maximal with respect to the inclusion, among all the cliques D in X such that ii) holds.

Observe that any complex C that is made of a single voxel is a clique (a 3-clique). Furthermore any such clique in a complex X is essential for X.

In figure 1 (left), $\{f, g\}$ is a 2-clique that is essential for X, $\{b, d\}$ is a 0-clique that is not essential for X, $\{b, c, d\}$ is a 0-clique essential for X, $\{e, f, g\}$ is a 1-clique essential for X.

Definition 3. Let $S \in \mathbb{V}^3$. The *$\mathcal{K}$-neighborhood of S*, written $\mathcal{K}(S)$, is the set made of all voxels that are adjacent to each voxel in S. We set $\mathcal{K}^*(S) = \mathcal{K}(S) \setminus S$.

We note that we have $\mathcal{K}(S) = \mathcal{N}(x)$ whenever S is made of a single voxel x. We also observe that we have $S \subseteq \mathcal{K}(S)$ whenever S is a clique.

Definition 4. Let $X \in \mathbb{V}^3$ and let C be a clique that is essential for X. We say that the clique C is *regular for X* if $\mathcal{K}^*(C) \cap X$ is reducible. We say that C is *critical for X* if C is not regular for X.

Notice that, if $C = \{x\}$ is a 3-clique in X, then C is regular for X if and only if x is simple for X. We can thus say that the notion of regular clique generalizes the one of simple voxel.

In figure 1 (left), the cliques $C_1 = \{b, c, d\}$, $C_2 = \{f, g\}$, and $C_3 = \{g, h\}$ are essential for X. We have $\mathcal{K}^*(C_1) \cap X = \emptyset$, $\mathcal{K}^*(C_2) \cap X = \{d, e, h\}$, and $\mathcal{K}^*(C_3) \cap X = \{f\}$. Thus, C_1 and C_2 are critical for X, while C_3 is regular for X.

The following result is a consequence of theorem 16 of [18] and a general theorem that holds for complexes of arbitrary dimensions [3].

Theorem 5. *Let $X \in \mathbb{V}^3$ and let $Y \subseteq X$. The complex Y is a thinning of X if any clique that is critical for X contains at least one voxel of Y.*

See an illustration in figure 1 where the complexes X and Y satisfy the condition of theorem 5. For example, the voxel d is a non-simple voxel for X, thus $\{d\}$ is a critical 3-clique for X, and d belongs to Y. Also, Y contains voxels in the critical cliques $C_1 = \{b, c, d\}$, $C_2 = \{f, g\}$, and the other ones.

The notion of critical kernel has been defined in the framework of simplicial or cubical complexes [3, 18]. However, it is possible to give a characterization of the critical kernel of a voxel complex X in terms of cliques. The *trace of a clique* C is the face $f = \cap\{x \in C\}$. The critical kernel of X is the set formed of the traces of the critical cliques of X (see an illustration in figure 3) and of all the faces included in these ones. Thus, theorem 5 can be restated in the following terms: The complex Y is a thinning of X if each face of the critical kernel of X is in at least one voxel of Y.

Notice that definitions of simple voxels and critical cliques, in this chapter, take advantage of special properties of faces in $\mathbb{Z}^d$, they cannot be transposed to other kinds of complexes (such as simplicial complexes).

4 Decreasing rank strategy

A parallel thinning algorithm consists of repeatidly aplying thinning steps, each of which is aimed at identifying and removing a subset of the current object. Equivalently, we have to define the subset of the object that will be kept until the end of the current step. In order to ensure topology preservation, when designing such an algorithm, we have to define a subset Y of a voxel complex X that is guaranteed to include at least one voxel of each clique that is critical for X. By theorem 5, this subset Y is a thinning of X.

There are many possible choices for defining such a subset Y. Among all the possible choices, we prefer the ones that provide sets Y having smaller sizes. Such a strategy will provide us with efficient thinning algorithms, in the sense where few thinning steps will be needed to achieve the final result.

Suppose that we examine critical cliques by decreasing rank (see figure 2 for the illustrations):
- A critical 3-clique is a clique composed of a unique, non-simple voxel (like voxel b). This is the simplest case, as no choice needs to be made: this voxel must be preserved.
- A critical 2-clique (like clique $\{a, b\}$) is necessarily composed of two voxels that intersect in a 2-face. If one of these two voxels forms a critical 3-clique (*i.e.*, is not simple, like voxel b), then this voxel has been detected previously and must be preserved. It is not necessary to keep the other one.
- A critical 1-clique (like clique $\{e, f, g, h\}$) may include critical 3-cliques or critical 2-cliques (like here, clique $\{e, g\}$). If it is the case, then at least one voxel of the included critical clique(s) has been chosen to be preserved, and no other voxel needs to be kept.

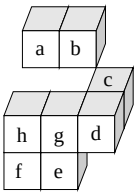


Fig. 2. A complex X.

Following this reasonning, which also applies to critical 0-cliques, we see that in order to reduce both the number of decisions and the number of kept voxels, we have to examine critical cliques by decreasing rank, and to keep track of the decisions that have been made for the cliques of higher ranks. This strategy will guide us for the design of all the algorithms proposed in this chapter.

5 Asymmetric thinning

We first introduce our 3D parallel asymmetric thinning scheme (see also [32, 4, 18]). Then, in 5.2 we show how to use this scheme in order to compute curve or surface skeletons, based on the notion of isthmus that we will introduce. We show in 5.3 some experimental results for curve thinning, which permits to compare our approach with respect to all previously published methods of the same kind.

5.1 Generic parallel asymmetric thinning scheme

The main features of the proposed scheme (algorithm `AsymThinningScheme`) are the following:
- Following the strategy presented in section 4, critical cliques are considered according to their decreasing ranks (step 3). Thus, each iteration includes four sub-iterations (steps 3–6). Voxels that have been previously selected are stored in a set Y (step 6). At a given sub-iteration, we consider only critical cliques included in $X \setminus Y$ (step 4).
- $Select$ is a function from $\mathbb{V}^3$ to V^3, the set of all voxels. More precisely, $Select$ associates, to each set S of voxels, a unique voxel x of S. We refer to such a function as a *selection function*. This function allows us to select a voxel in any given critical clique (step 5). A possible choice is to take for $Select(S)$, the first voxel of S in the lexicographic order of the voxels coordinates.
- In order to compute curve or surface skeletons, we have to keep other voxels than the ones that are necessary for the preservation of the topology of the object X. In the scheme, the set K corresponds to a set of features that we want to be preserved by a thinning algorithm (thus, we have $K \subseteq X$). This set K, called *constraint set*, is updated dynamically at step 8. The parameter $Skel_X$ is a function from X on $\{True, False\}$ that allows us to detect some *skeletal voxels* of X, *e.g.*, some voxels belonging to parts of X that are surfaces or curves. For example, if we want to obtain curve skeletons, a popular solution is to set $Skel_X(x) = True$ whenever x is a so-called *end voxel* of X: an end voxel is a voxel that has exactly one neighbor inside X. However, this solution is limited and does extend easily to the case of surface skeletons. Alternative choices for such a function will be introduced in section 5.2.

By construction, at each iteration, the complex Y at step 7 satisfies the condition of theorem 5. Thus, the result of the scheme is a thinning of the original complex X, whatever the choices of parameters K and $Skel_X$. Observe also that, except step 3, each step of the scheme may be computed in parallel.

Figure 3 provides an illustration of `AsymThinningScheme`. Let us consider the complex X depicted in (a). We suppose in this example that we do not keep any skeletal voxel, *i.e.*, the initial constraint set K is empty, and for any x in X, we set $Skel_X(x) = False$. The set of traces of the cliques that are critical for X is represented in (b). Thus, the set of the cliques that are critical for X is precisely composed of six 0-cliques, two 1-cliques, three 2-cliques, and one 3-clique. In (c) the different sub-iterations of the first iteration of the scheme (lines 3–5) are illustrated:

Algorithm 1: `AsymThinningScheme`$(X, K, Skel_X)$

Data: $X \in \mathbb{V}^3$, $K \subseteq X$, $Skel_X$ is a function from X on $\{True, False\}$
Result: X

```
repeat
    Y := K;
    for d ← 3 to 0 do
        B := set of all d-cliques that are critical for X and included in X \ Y;
        A := {Select(C) | C ∈ B};
        Y := Y ∪ A;
    X := Y;
    foreach voxel x ∈ X \ K such that Skel_X(x) = True do K := K ∪ {x};
until stability ;
```

- when $d = 3$, only one clique is considered, the dark grey voxel is selected whatever the selection function;
- when $d = 2$, all the three 2-cliques are considered since none of these cliques contains the above voxel. Voxels that could be selected by a selection function are depicted in medium grey;
- when $d = 1$, only one clique is considered, a voxel that could be selected is depicted in light grey;
- when $d = 0$, no clique is considered since each of the 0-cliques contains at least one voxel that has been previously selected.

After these sub-iterations, we obtain the complex depicted in (d). The figures (e) and (f) illustrate the second iteration, at the end of this iteration the complex is reduced to a single voxel.

Of course, the result of the scheme may depend on the choice of the selection function. This is the price to be paid if we try to obtain thin skeletons. For example, some arbitrary choices have to be made for reducing a two voxels wide ribbon to a simple curve.

In the sequel of the chapter, we take for $Select(S)$, the first voxel of S in the lexicographic order of the voxels coordinates.

Figure 4 shows another illustration, on bigger objects, of `AsymThinningScheme`. Here also, we give $\emptyset$ as parameter K, and for any x in X, we have $Skel_X(x) = False$ (no skeletal voxel). The result is called an ultimate asymmetric skeleton.

5.2 Isthmus-based asymmetric thinning

In this section, we show how to use our generic scheme `AsymThinningScheme` in order to get a procedure that computes either curve or surface skeletons. This thinning procedure preserves a constraint set K that is made of "isthmuses".

Intuitively, a voxel x of an object X is said to be a 1-isthmus (resp. a 2-isthmus) if the neighborhood of x corresponds - up to a thinning - to the one of a point belonging to a curve (resp. a surface) [18].

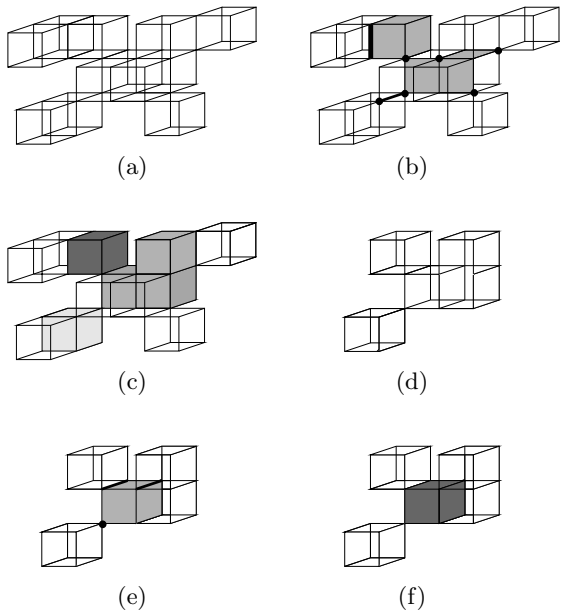


Fig. 3. (a): A complex X made of precisely 12 voxels. (b): The set of traces of the cliques that are critical for X (*i.e.*, the maximal faces of the critical kernel of X). (c): Voxels that have been selected by the algorithm. (d): The result Y of the first iteration. (e): The set of traces of the cliques that are critical for Y. (f): The result of the second iteration.

We say that $X \in \mathbb{V}^3$ is a 0-*surface* if X is precisely made of two voxels x and y such that $x \cap y = \emptyset$.

We say that $X \in \mathbb{V}^3$ is a 1-*surface* (or a *simple closed curve*) if:
i) X is connected; and
ii) For each $x \in X$, $\mathcal{N}^*(x) \cap X$ is a 0-surface.

Definition 6. Let $X \in \mathbb{V}^3$, let $x \in X$. We say that:
the voxel x is a 1-*isthmus for* X if $\mathcal{N}^*(x) \cap X$ is reducible to a 0-surface,
the voxel x is a 2-*isthmus for* X if $\mathcal{N}^*(x) \cap X$ is reducible to a 1-surface,
the voxel x is a 2^+-*isthmus for* X if x is a 1-isthmus or a 2-isthmus for X.

See figure 5 for an illustration of 1- and 2-isthmuses.

Notice that statement 1) of forthcoming theorem 8 (and the fact that 0- and 1-surfaces are not reducible) will imply that every 1- or 2-isthmus voxel is non-simple.

Our aim is to thin an object, while preserving a constraint set K, initially empty, that is made of voxels that are detected as k-isthmuses during the thinning process. We obtain curve skeletons with $k = 1$, and surface skeletons with $k = 2^+$. These two kinds of skeletons may be obtained by using

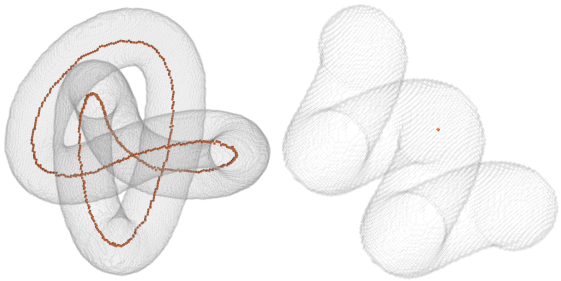


Fig. 4. Ultimate asymmetric skeletons obtained by using `AsymThinningScheme`. On the left, the object is a solid cylinder bent to form a knot. Its ultimate skeleton S is a discrete curve, *i.e.*, for any x in S, $\mathcal{N}^*(x) \cap S$ is made of exactly two voxels. On the right, the object is connected and without holes and cavities. Its ultimate skeleton is a single voxel.

`AsymThinningScheme`, with the function $Skel_X$ defined as follows:

$$Skel_X(x) = \begin{cases} True & \text{if } x \text{ is a } k\text{-isthmus,} \\ False & \text{otherwise.} \end{cases}$$

Observe that a voxel may belong to a k-isthmus at a given step of the algorithm, but not at further steps. This is why previously detected isthmuses are stored (see line 8 of `AsymThinningScheme`).

In figure 6, we show a curve skeleton and a surface skeleton obtained by our method from the same object.

5.3 Comparison with other parallel curve skeletonization methods

In order to validate our approach, we made some experiments to compare our isthmus-based curve thinning method using `AsymThinningScheme` with all other existing methods of the same kind. To make a fair comparison, we considered only parallel asymmetric thinning methods that produce curve skeletons of voxel objects, and that have no parameter, namely [8–17].

In figure 7 and figure 8, we show the results of all these methods and ours for a same shape. We notice in particular that some methods, for example [8] (a), are not sufficiently powerful to produce results that may be interpreted as curve skeletons. This illustrates the difficulty of designing a method that keeps enough voxels in order to preserve topology, and in the same time, deletes a sufficient number of voxels in order to produce thin curve skeletons. This difficulty is indeed high when these two opposite constraints are not made explicit in the algorithm. The strength of our approach lies in a complete separation of these constraints.

The example of figure 8 illustrates very well the sensitivity to contour noise of the methods. The original object is a discretized thickened spiral, and its curve

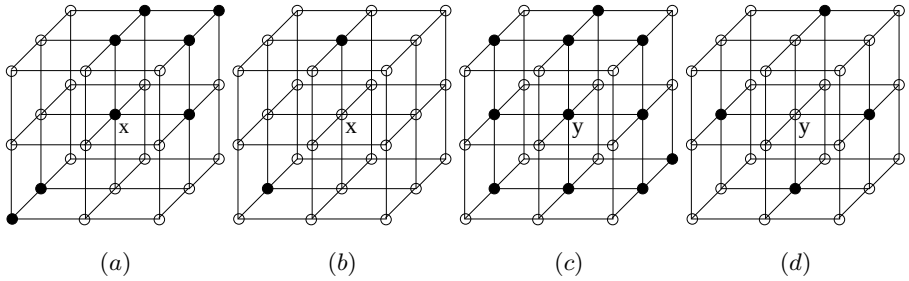


Fig. 5. In this figure, a voxel is represented by its central point. (a): A voxel x and the set $\mathcal{N}(x) \cap X$ (black points). (b): A set S which is a 0-surface, $\mathcal{N}^*(x) \cap X$ is reducible to S, thus x is a 1-isthmus for X. (c): A voxel y and the set $\mathcal{N}(y) \cap X$ (black points). (d): A set S which is a 1-surface, $\mathcal{N}^*(y) \cap X$ is reducible to S, thus y is a 2-isthmus for X.

skeleton should ideally be a simple curve. Any extra branch of the skeleton must undoubtedly be considered as spurious. To count the number of spurious branches for this example, we can simply count the number of end points and substract 2, as the ideal skeleton has exactly two extremities. In figure 8, we indicate for each skeleton the number of spurious branches. Notice that only [15] and `AsymThinningScheme` produce a skeleton free of spurious branches for this object.

Furthermore, in [21], we describe a quantitative study that shows that our method outperforms the other ones with respect to contour noise sensitivity.

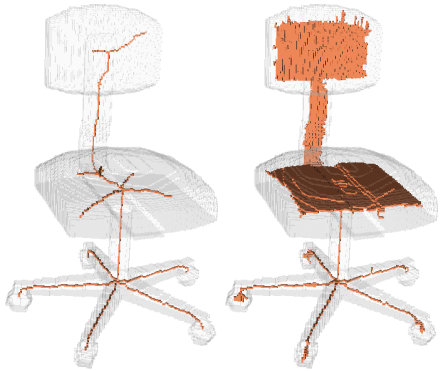


Fig. 6. Asymmetric skeletons obtained by using `AsymThinningScheme`. Left: curve skeleton. The function $Skel_X$ is based on 1-isthmuses. Right: surface skeleton. The function $Skel_X$ is based on 2^+-isthmuses.

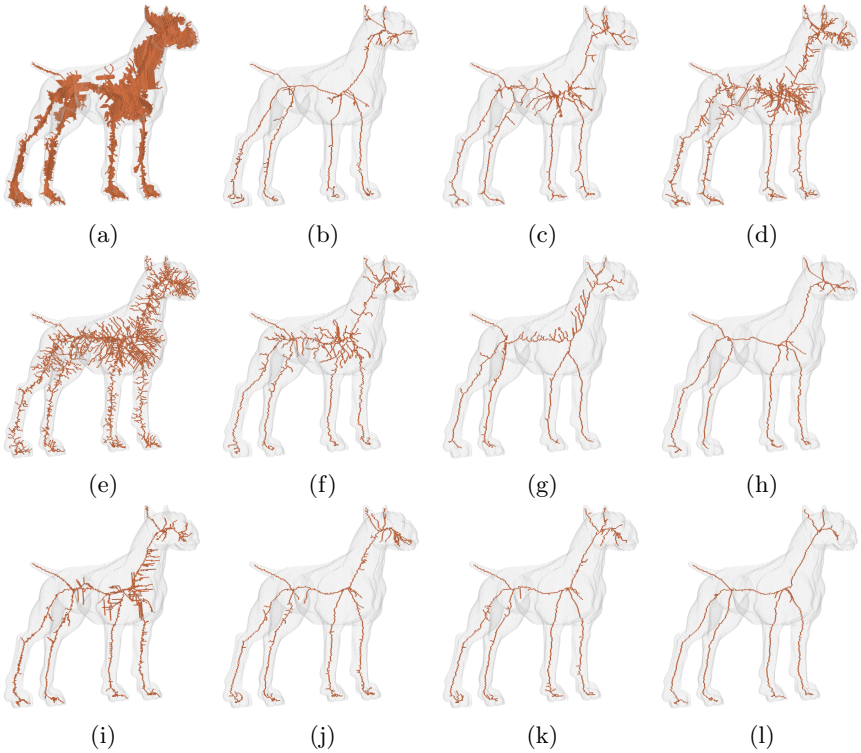
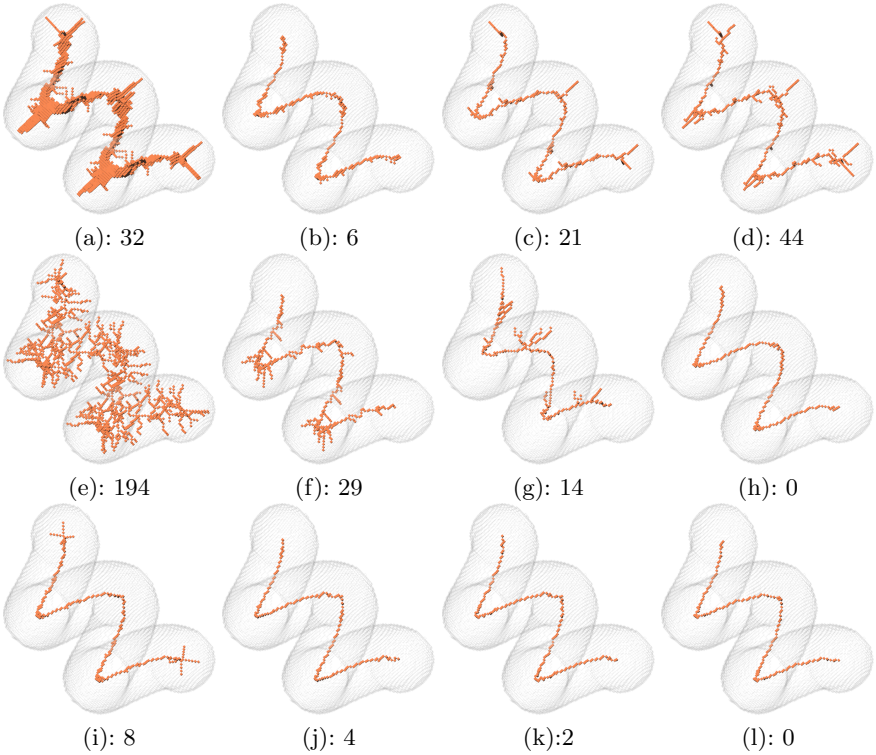


Fig. 7. Curve skeletons of a same object obtained through different methods: (a) [8], (b) [9], (c) [10], (d) [11], (e) [12], (f) [13], (g) [14], (h) [15], (i) [16], (j) [17], (k) [17], (l) `AsymThinningScheme`.



(a): 32 (b): 6 (c): 21 (d): 44

(e): 194 (f): 29 (g): 14 (h): 0

(i): 8 (j): 4 (k):2 (l): 0

Fig. 8. Idem figure 7, the number of spurious branches is given for each skeleton.

6 Symmetric thinning

As in the previous section, our goal here is to define a subset of a voxel complex X that is guaranteed to include at least one voxel of each clique that is critical for X. By theorem 5, this subset will be a thinning of X.

But now, we want our method to be independent of arbitrary choices, in particular of a choice of specific voxels in a given critical clique. This will ensure that the obtained skeletons are invariant with respect to 90 degrees rotations.

To obtain this result, we do not consider individual voxels anymore, but only cliques. We denote by $\mathcal{C}(X)$ the set of all cliques included in X.

6.1 Generic parallel symmetric thinning scheme

As in section 5.1, we first propose a generic algorithm (algorithm `SymThinningScheme`) that takes as input parameters a voxel complex X, a constraint set $K \subseteq X$, and a boolean function $Skel_X$, which serves to dynamically detect the cliques that must be preserved during the thinning.

Following the strategy presented in section 4, critical cliques are considered according to their decreasing ranks (step 3).

Algorithm 2: `SymThinningScheme`$(X, K, Skel_X)$

Data: $X \in \mathbb{V}^3$, $K \subseteq X$, $Skel_X$ a function from $\mathcal{C}(X)$ on $\{True, False\}$
Result: X

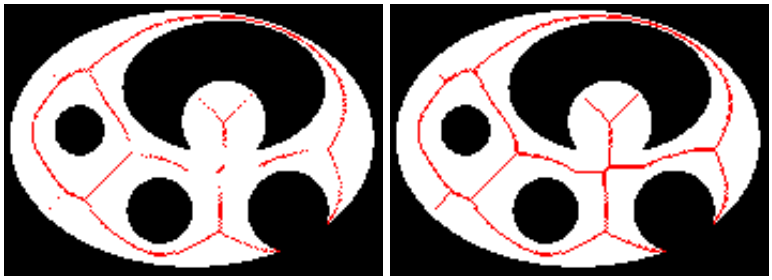
```
1 repeat
2 |  Y := K;
3 |  for d ← 3 to 0 do
4 |  |  A := union of all d-cliques that are critical for X and included in X \ Y;
5 |  |  Y := Y ∪ A;
6 |  X := Y;
7 |  foreach clique C ∈ C(X) such that Skel_X(C) = True do  K := K ∪ C;
8 until stability ;
```

Let us illustrate this scheme with a commonly used kind of constraint set, namely, the medial axis of X.

The notion of medial axis has been introduced by Blum in the 60s [33]. Let S be a subset of $\mathbb{R}^n$ or $\mathbb{Z}^n$, the medial axis of S consists of the centers of the n-dimensional balls that are included in S but that are not included in any other n-dimensional ball included in S. The medial axis of a voxel set X may be readily defined by identifying each voxel of X with its center.

In the continuous space $\mathbb{R}^n$, the medial axis of any set S is known to be homotopy-equivalent to S, under certain conditions of "smoothness" [34]. However, this is not the case in discrete grids, whatever the distance that is chosen to define the notion of ball. This fact is illustrated in 2D in figure 9 (a).

In order to ensure topology preservation, we use `SymThinningScheme` to thin the original object X with the only constraint of preserving the medial axis of X, see figure 9 (b). Notice that, by its very definition, the medial axis is invariant to 90 degrees rotations. Thus, using `SymThinningScheme`, we ensure that the skeleton finally obtained is invariant to 90 degrees rotations.



(a) (b)

Fig. 9. (a): A voxel set X and its medial axis $\mathrm{MA}(X)$, based on the so-called city block distance or 4-distance. (b): Result of $\texttt{SymThinningScheme}(X, \mathrm{MA}(X), Skel_X)$, with $Skel_X(C) = False$ for any $C \in \mathcal{C}(X)$.

6.2 Isthmus-based symmetric thinning

In section 5.2, we defined 1-isthmuses and 2-isthmuses as voxels satisfying a certain condition. In order to handle thick structures, as for example two-voxel thick ribbons that may exist in symmetric curve skeletons, we replace the notion of a voxel by the one of a clique. Intuitively, and generalizing definition 6, a critical clique C of an object X is said to be a 1-isthmus (resp. a 2-isthmus) if the neighborhood of C corresponds - up to a thinning - to the one of a point belonging to a curve (resp. a surface).

Definition 7. Let $X \in \mathbb{V}^3$ and let C be a clique which is essential for X. We say that :
the clique C is a 1-*isthmus for* X if $\mathcal{K}^*(C) \cap X$ is reducible to a 0-surface,
the clique C is a 2-*isthmus for* X if $\mathcal{K}^*(C) \cap X$ is reducible to a 1-surface,
the clique C is a 2^+-*isthmus for* X if C is a 1-isthmus or a 2-isthmus for X.

Recall that a clique C is critical for X if C is essential for X and if $\mathcal{K}^*(C) \cap X$ is not reducible to a single voxel (Def. 4). It will follow from statement 1) of forthcoming theorem 8 that a k-isthmus, $k \in \{1, 2, 2^+\}$, is necessarily critical. See figure 10 and figure 5 where several examples of 1- and 2-isthmuses are given.

Curve skeletons (with $k = 1$), and surface skeletons (with $k = 2^+$) may be obtained by using `SymThinningScheme`, with the function $Skel_X$ defined as

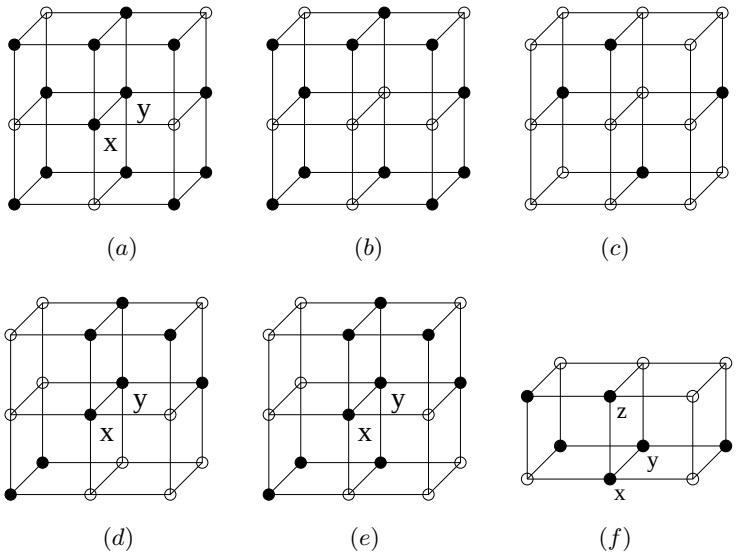


Fig. 10. In this figure, a voxel is represented by its central point. (a): A 2-clique $C = \{x, y\}$ and the set $\mathcal{K}(C) \cap X$ (black points). It can be seen that C is essential for X. (b): The set $D = \mathcal{K}^*(C) \cap X$. (c): A set E, D is reducible to E. The set E is a 1-surface, thus C is a 2-isthmus for X. (d): A 2-clique $C = \{x, y\}$ and the set $\mathcal{K}(C) \cap X$ (black points), C is a 1-isthmus for X. (e): A 2-clique $C = \{x, y\}$ and the set $\mathcal{K}(C) \cap X$, C is neither a 1-isthmus nor a 2-isthmus for X. In fact C is regular for X. (f): A 1-clique $C = \{x, y, z\}$ and the set $\mathcal{K}(C) \cap X$, C is a 1-isthmus for X.

follows, for any C in $\mathcal{C}(X)$:

$$Skel_X(C) = \begin{cases} True & \text{if } C \text{ is a } k\text{-isthmus,} \\ False & \text{otherwise.} \end{cases}$$

In figure 11 and figure 12, we illustrate the above algorithm for computing curve and surface skeletons respectively.

To our knowledge, the only other parallel and symmetric method for obtaining both types of skeletons was proposed in [6]. It is based on P-simple points, introduced by one of the authors [2]. The obtained skeletons are very close to the ones produced using the methods described in this chapter. However, in [6], the preservation of salient object features is achieved through the detection of end voxels. Using this strategy, it is not possible to define a notion of persistence (see section 8) that allows one to filter skeletons. Filtering is mandatory in almost all applications dealing with skeletons, as noise is usually affecting real data and skeletons are notoriously sensitive to noise (see figures 11, 12, 13). This problem will be studied and a solution will be proposed in the following section (see figure 14).

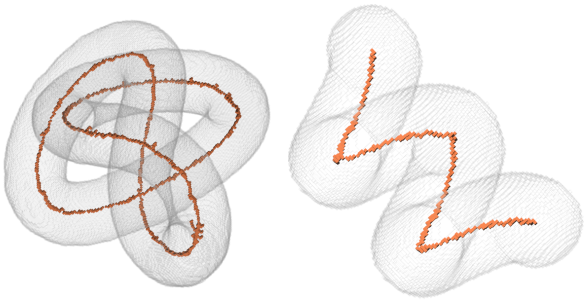


Fig. 11. Illustrations of isthmus-based curve thinning using `SymThinningScheme`.

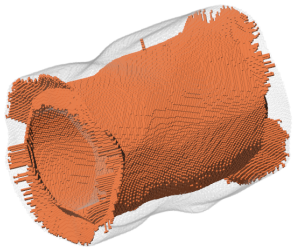


Fig. 12. Illustration of isthmus-based surface thinning using `SymThinningScheme`.

7 Characterization of critical cliques and k-isthmuses

A key point, in the implementation of the algorithms proposed in this chapter, is the detection of critical cliques and k-isthmuses.

In this section, we will see that it is possible to detect these cliques with efficient algorithms operating on four different kinds of neighborhoods. See also [18] where a set of masks is proposed for critical cliques.

First of all, recall that a clique that is either critical or a k-isthmus for a complex X in $\mathbb{V}^3$, is necessarily essential for X.

Up to $\pi/2$ rotations, the three configurations $\mathcal{C}_2$, $\mathcal{C}_1$, and $\mathcal{C}_0$ given in figure 15 may be used for the detection of a clique that is essential for a complex $X \in \mathbb{V}^3$ (in this figure a voxel is represented by a point). Let $C \subseteq X$. It may be seen that, up to $\pi/2$ rotations:
- C is a 2-clique that is essential for X if and only if $C = \mathcal{C}_2$.
- C is a 1-clique that is essential for X if and only if $C = \mathcal{C}_1 \cap X$ and there exist two voxels of C that are 1-neighbors.
- C is a 0-clique that is essential for X if and only if $C = \mathcal{C}_0 \cap X$ and there exist

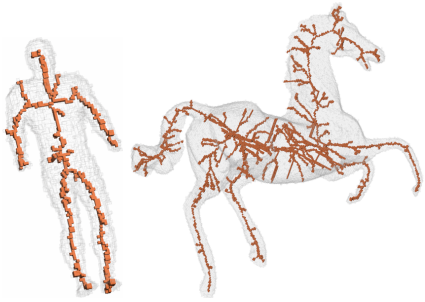


Fig. 13. Illustrations of isthmus-based curve thinning using `SymThinningScheme` with noisy shapes (a small amount of random noise has been added to the contours of the shapes).

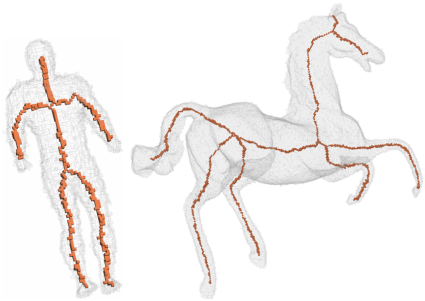


Fig. 14. Preview of the filtered skeletons obtained using the method of section 8, on the same noisy shapes as in figure 13.

two voxels of C that are 0-neighbors, or three voxels of C that are mutually 1-neighbors.

The $\mathcal{K}$-neighborhoods of the configurations $\mathcal{C}_2$, $\mathcal{C}_1$, and $\mathcal{C}_0$ are given figure 16. Observe that we have $\mathcal{K}_0 = \mathcal{C}_0$.

Recall that a set made of a single voxel x of an object X constitutes a clique (a 3-clique) which is essential for X. Therefore, there are precisely four different kinds of neighborhoods for an essential clique ($\mathcal{K}_0$, $\mathcal{K}_1$, $\mathcal{K}_2$, and $\mathcal{N}(x)$).

Thus, the neighborhoods involved in the very definitions of critical cliques and k-isthmuses (Def. 4 and Def. 7) are fully specified.

Nevertheless, we observe that, in these two definitions, we have to check if a given complex S is reducible to a certain complex T determined by specific properties.

For example, for checking whether an essential clique C is a 2-isthmus for X, we have to verify whether $S = \mathcal{K}^*(C) \cap X$ is reducible to a simple closed curve (a 1-surface).

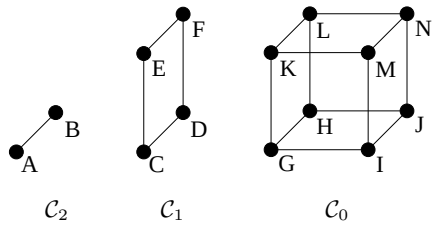


Fig. 15. Masks for 2-cliques ($\mathcal{C}_2$), 1-cliques ($\mathcal{C}_1$), and 0-cliques ($\mathcal{C}_0$). Here, a voxel is represented by its central point.

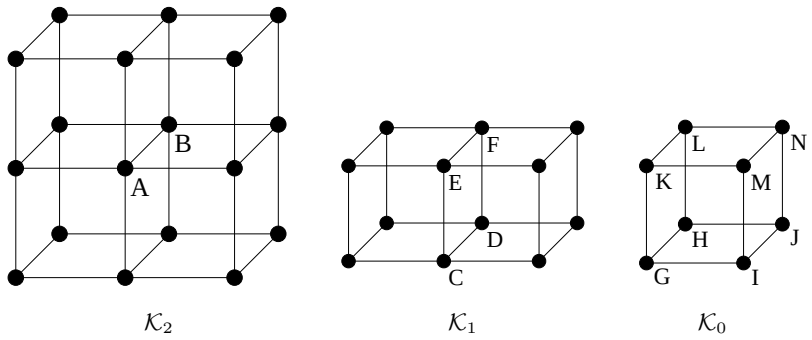


Fig. 16. $\mathcal{K}$-neighborhoods for 2-cliques ($\mathcal{K}_2$), 1-cliques ($\mathcal{K}_1$), and 0-cliques ($\mathcal{K}_0$). A voxel is represented by its central point.

Let $S \in \mathbb{V}^3$ be an arbitrary complex which is reducible to a complex T. In fact, there is the possibility that S is reducible to a complex R, with $T \subseteq R$, but that R is not reducible to T.

For example, such a situation occurs when S is a cuboid, T is made of a single voxel, and R is an object such as the so-called *dunce hat* [35], or *the house with two rooms* [36]. In [30], this kind of object is referred to as *a lump*.

As a consequence of this fact, for testing if an arbitrary complex S is reducible to T, it is a priori necessary to compute all the complexes R such that S is reducible to R, and $T \subseteq R$. In other words, the "reducibility problem" is, in general, a complex problem from the algorithmic point of view. See *e.g.* [37] for this issue.

In [30], it was shown that it is possible to find lumps in the neighborhood of a cubical element (a face) of $\mathbb{Z}^5$. The following result —which has been proved with the help of a computer program, as explained in [20]— shows that, in $\mathbb{V}^3$, there is not enough space for such objects to lie in the $\mathcal{K}$-neighborhood of a clique. Note that point 1) of theorem 8 corresponds to theorem 17 of [18].

Theorem 8 ([20])**.** *Let $C \in \mathbb{V}^3$ be a clique, let $S \subseteq \mathcal{K}^*(C)$, and let x be a voxel which is simple for S.*

1) If S is reducible, then $S \setminus \{x\}$ is reducible.
2) If S is reducible to a 0-surface, then $S \setminus \{x\}$ is reducible to a 0-surface.
3) If S is reducible to a 1-surface, then $S \setminus \{x\}$ is reducible to a 1-surface.

As a consequence of theorem 8, we can use a greedy algorihm for testing if an essential clique C is critical, a 1-isthmus, a 2-isthmus or not critical for X. We can arbitrarily select and remove a voxel which is simple for the set $S = \mathcal{K}^*(C) \cap X$. After repeating this operation until stability, we check if C is critical (S is not made of a single voxel), if C is a 1-isthmus (S is made of two voxels), or if C is a 2-isthmus (S is a simple closed curve).

Testing whether a voxel x is simple or not for an object $S \in \mathbb{V}^3$ can be done in constant time. This may be done by using previously proposed characterizations [30], or by using a pre-computed look-up table. From theorem 8, testing the status of a clique may be performed in linear time with respect to the size of its neighborhood (at most 26). Again, the results can be pre-computed and stored in look-up tables to speed up the test. Each test (simple, critical, 1-isthmus, 2-isthmus) can thus be performed in constant time.

8 Isthmus persistence and skeleton filtering

In this section and the following one, we focus on symmetric thinning. However, the persistence-based thinning method presented next is adaptable with only very minor changes to the asymmetric thinning algorithm presented in section 5.2.

Even in the continuous framework, the skeleton suffers from its sensitivity to small contour perturbations, in other words, it lacks stability. This fact, among others, explains why it is often necessary to add a filtering step (or pruning step) to any method that aims at computing the skeleton. Hence, there is a rich literature devoted to skeleton pruning, in which different criteria were proposed in order to discard "spurious" skeleton points or branches, see *e.g.* [20], section 7.

A particularly appealing approach to prevent the appearance of spurious skeleton branches (or surfaces) is based on the notion of isthmus persistence. Its advantages are the following:
- it applies both in 2D and 3D cases, and to curve as well as surface skeletons,
- it is governed by a single parameter,
- it is effective (although measuring this criterion is difficult, because the quality of a skeleton is always a tradeoff between fidelity and low complexity, see [38], and therefore its assessment depends very much on the application),
- it is quite simple to compute in the framework of a thinning procedure.

This notion first appeared, to our knowledge, in the context of cubical complexes, in the work of L. Liu *et al.* [19].

The idea is the following. When a voxel is detected as belonging to an isthmus for the first time during the thinning process, the number of thinning steps that have been performed at this moment is recorded and called the *birth date* of this voxel. Intuitively, it gives an information about the local thickness of the object

around this voxel (see figure 17 for an illustration in 2D). Then, later on during the thinning process, the same voxel may become a candidate for deletion, that is, it is no longer in an isthmus. At this time, we record the number of thinning steps that were performed, and we call this number the *death date* of the voxel. The difference *death date* − *birth date* is called the *persistence* of the voxel (see also [39]). A voxel that has a short persistence is very likely to be part of a spurious skeleton branch, whereas voxels with long persistences indicate the presence of robust skeleton parts (figure 17). In this approach, only isthmuses having a persistence greater than a given threshold will be kept.

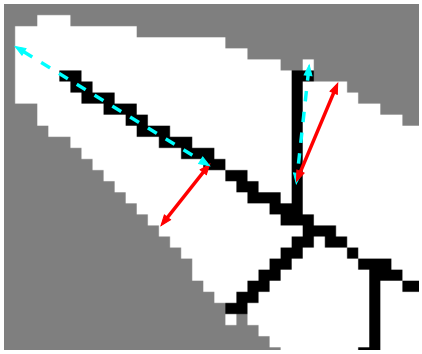


Fig. 17. The intervals depicted with a solid line correspond to the birth dates, the dotted lines to the death dates.

The works [19] and [39] present some asymmetric thinning procedures which take place in the framework of general cubical complexes, *i.e.*, complexes that are not necessarily made of voxels. With our definitions of 1D and 2D isthmuses, either single-voxel or multiple-voxels cliques, it becomes easy to implement this stragegy to detect robust skeleton elements in the context of symmetric or asymmetric parallel thinning of objects made of voxels.

In the following algorithm, k stands for the type of the considered isthmuses (1 or 2^+), and p is a parameter that sets the persistence threshold. The function b associates to certain voxels their birth date, and K is a constraint set that is dynamically updated by adding those voxels whose persistence is greater than the threshold p (lines 9–10).

In line 8, the birth date $b(x)$ of each new isthmus voxel x is recorded. Notice also that the voxels in P (line 9) are deletable. The test $b(x) > 0$ means that such a voxel x was an isthmus voxel formerly, and its death date is i. These voxels are kept in X, and added to the constraint set K (line 10) because of their long persistence (at least p).

In figures 18 and 19 we show outcomes of algorithm `PersistenceSymThinning`, with $k = 1$ and $k = 2^+$, respectively, for different values of parameter p.

Algorithm 3: `PersistenceSymThinning`$(X,\ k,\ p)$

Data: $X \in \mathbb{V}^3$, $k \in \{1, 2^+\}$, $p \in \mathbb{N} \cup \{+\infty\}$
Result: X

```
1  i := 0; K := ∅; foreach x ∈ X do b(x) := 0;
2  repeat
3      i := i + 1; Y := K; Z := ∅;
4      for d ← 3 to 0 do
5          A := union of all d-cliques that are critical for X and included in X \ Y;
6          B := union of all d-cliques that are k-isthmuses for X and included in
           X \ Y;
7          Y := Y ∪ A; Z := Z ∪ B;
8      foreach x ∈ Z such that b(x) = 0 do b(x) := i;
9      P := {x ∈ X \ Y | b(x) > 0 and i − b(x) ⩾ p};
10     K := K ∪ P; X := Y ∪ P;
11 until stability ;
```

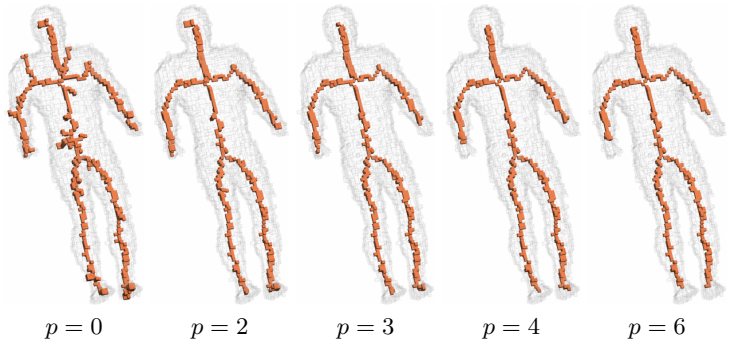
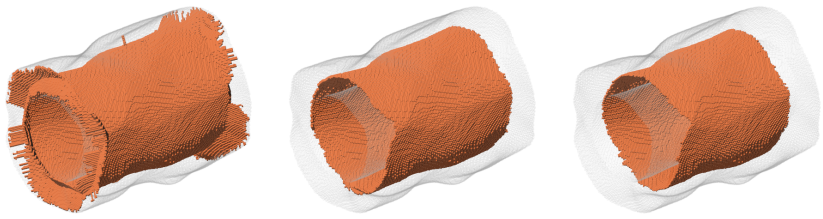


Fig. 18. Outcomes of algorithm `PersistenceSymThinning`, with $k = 1$, for different values of parameter p.

9 Hierarchies of skeletons

With the preceding examples, we saw that —intuitively— the greater the persistence, the smaller the filtered skeleton. By varying the value of the persistence parameter, we may compute a whole family of nested homotopic skeletons, to the cost of applying the previous algorithm once for each parameter value ("naive" approach). Once computed, such a family can be efficiently stored as a function, *i.e.*, a grayscale image, obtained by stacking all the filtered skeletons. Thresholding this function (named *skeleton stack function*, or *SSF* in the sequel) at a given value would give back the filtered skeleton corresponding to this value of the parameter.

More specifically, given a voxel complex X, we want to compute a function Ψ from X to $\mathbb{N} \cup \{+\infty\}$. Let Ψ_k denote the threshold of Ψ at level k, *i.e.*, $\Psi_k = \{x \in X \mid \Psi(x) \geqslant k\}$. The function Ψ must meet the two following requirements:



$p = 0$ $p = 10$ $p = 20$

Fig. 19. Outcome of algorithm `PersistenceSymThinning`, with $k = 2^+$, for different values of parameter p.

i) Ψ_0 is a symmetric skeleton of X (unfiltered), and
ii) for any k and any ℓ in $\mathbb{N}$ such that $k > \ell$, Ψ_k is a symmetric thinning of Ψ_ℓ.

It may be interesting to compute a SSF instead of a single filtered skeleton in cases, which are frequent, where there is no trivial way to fix *a priori* a single parameter value for a set of images. In such cases, the precomputed SSF may allow a user to interactively visualize the effect of each parameter value on the filtering. Alternatively, a post-processing algorithm may be applied, in certain applications, to automatically choose a threshold of the SSF that satisfies a given criterion (for example, a chosen number of skeleton branches).

Now, we are facing the problem of efficiently computing an SSF. We show in the rest of this section that the cost of this computation can be dramatically reduced with respect to the one of the naive method evoked at the beginning of this section.

A first idea consists of storing, during the thinning process, the function Π that associates its persistence to each voxel that is detected as an isthmus at any step. The algorithm `Persistence` (algorithm 4), derived from algorithm `PersistenceSymThinning`, plays this role. This algorithm computes the birth and death dates of all non-permanent isthmus voxels, and stores the differences (lines 8–12). This corresponds to the case where the parameter p is set to infinity in algorithm `PersistenceSymThinning`.

What happens if we threshold the map Π at a given level π? We could expect to get a filtered skeleton as computed by algorithm `PersistenceSymThinning` with parameter $p = \pi$, but this is not always the case: see figure 20 for a counter-example. We also see with this example that the topological characteristics are not always preserved by this procedure.

In order to compute a skeleton stack based on the persistence map, we introduce algorithm `PersistenceSymThinning` (algorithm 5). This algorithm first computes a persistence function Π using algorithm `Persistence` (line 1). Then, it computes the lowest function Ψ that is above Π and satisfies the requirements i) and ii) of an SSF. To do this, the algorithm performs a symmetric parallel

Algorithm 4: `Persistence`$(X,\ k)$

Data: $X \in \mathbb{V}^3$, $k \in \{1, 2^+\}$
Result: Π : a function from X into $\mathbb{N} \cup \{+\infty\}$

```
1  i := 0; foreach x ∈ X do Π(x) := 0;
2  repeat
3  |  i := i + 1;
4  |  Y := ∅; Z := ∅; for d ← 3 to 0 do
5  |  |  A := union of all d-cliques that are critical for X and included in X \ Y;
6  |  |  B := union of all d-cliques that are k-isthmuses for X and included in
   |  |  X \ Y;
7  |  |  Y := Y ∪ A; Z := Z ∪ B;
8  |  foreach x ∈ Z such that Π(x) = 0 do
9  |  |  Π(x) := i; // birth
11 |  foreach x ∈ X \ Y such that Π(x) ≠ 0 do
12 |  |  Π(x) := i − Π(x); // death
14 |  X := Y;
15 until stability ;
16 foreach x ∈ X do Π(x) := +∞;
```

thinning of X guided by Π (that is, voxels are, loosely speaking, considered in increasing order of the value $\Pi(x)$, see lines 6 and 8–14. Notice however that, when a voxel x is deleted at some iteration, each undeleted neighbor y of x will be considered for possible deletion at a future iteration even if $\Pi(y) < \Pi(x)$, see lines 15 and 17–18). For each deleted voxel x, the value of a variable ψ is stored as $\Psi(x)$ (line 16). The value of ψ increases monotonically during the thinning, as slowly as possible, but at each iteration of the main loop line 7 ensures that it will not be less than λ, which is the persistence value $\Pi(x)$ of the voxels x that are to be considered for deletion at that iteration.

We illustrate this algorithm in 2D, as it is much easier to visualize a 2D map than a 3D one. Figure 21 displays an object X, its SSF computed by the above algorithm, and two thresholds of this map. The first threshold, at the value 1, is indeed identical to the result of algorithm `IsthmusSymmetricThinning` with $k = 1$ (curve skeleton). The second threshold, at the value 3, provides a filtered skeleton that captures the main structure of the object.

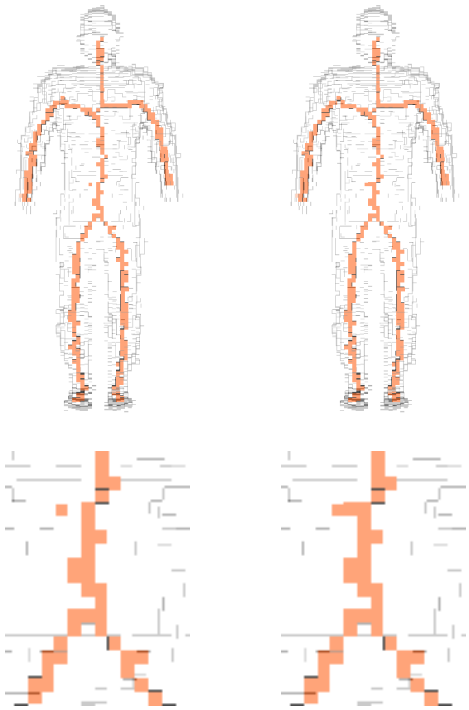


Fig. 20. Left: a threshold of the map Π, computed by algorithm `Persistence`, at value 3. Right: outcome of algorithm `PersistenceSymThinning` with $p = 3$. Bottom row: detail (zoomed).

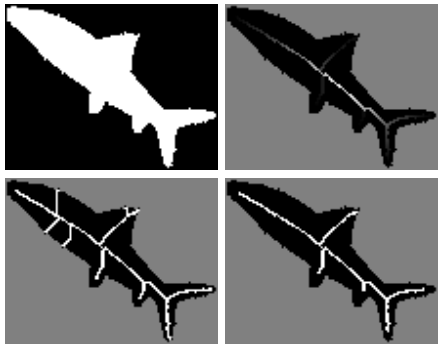


Fig. 21. First row: an object X (left), and its SSF Ψ (right). The darker the pixel, the lowest the value. Notice that the brightest value corresponds to the value $+\infty$, which is given to pixels that belong to any filtered skeleton. Second row: thresholds of the map Ψ at values 1 (left) and 3 (right).

Algorithm 5: `PersistenceSSF`$(X,\, k)$

Data: $X \in \mathbb{V}^3$, $k \in \{1, 2^+\}$
Result: Ψ : a function from X into $\mathbb{N} \cup \{+\infty\}$

```
1  Π := Persistence(X, k);
2  ψ := −∞;
3  foreach x ∈ X do Ψ(x) := +∞;
4  Q := {(x, Π(x)) | x ∈ X};
5  repeat
6  |  λ := min{π | (., π) ∈ Q};
7  |  if λ > ψ then ψ := λ;
8  |  P := {x ∈ X | (x, λ) ∈ Q};
9  |  Q := Q \ {(x, λ) | x ∈ P};
10 |  Y := X \ P;
11 |  for d ← 3 to 0 do
12 |  |  A := set of all voxels belonging to any d-clique which is critical for X
   |  |  and included in X \ Y;
13 |  |  Y := Y ∪ A;
14 |  V := X \ Y; X := Y;
15 |  foreach x ∈ V do
16 |  |  Ψ(x) := ψ;
17 |  |  foreach y ∈ N*(x) ∩ X do
18 |  |  |  Q := Q ∪ {(y, Π(y))};
19 until Q = ∅ ;
```

10 Complexity

In this section, we discuss the time complexity of the five algorithms presented in the chapter.

Algorithm 1 (`AsymThinningScheme`), Algorithm 2 (`SymThinningScheme`):
These algorithms may be computed in $O(t.n)$ time on a sequential computer, where t is the number of steps to thin the object (which corresponds, intuitively, to its thickness), and in $O(t)$ on an ideal parallel computer.

Alternatively, a breadth-first strategy may be employed in order to achieve a linear ($O(n)$) processing time for these algorithms with a sequential computer. It consists of maintaining a list of the voxels whose status changes during an iteration, and to use it to restrict the work to be done, in the next iteration, to their neighborhoods.

Algorithm 3 (`PersistenceSymThinning`):
On an ideal parallel computer, the instructions in lines 5–7 may be executed in constant time. Thus, the overall complexity of the algorithm is in $O(t)$, where t represents the number of steps of the repeat loop needed to reach stability (intuitively, this corresponds to the thickness of the object X). On a sequential computer, a direct implementation leads to an algorithm in $O(t \times n)$, where n is the number of voxels in the boolean array representation of X. However, the use of a breadth-first strategy leads to a linear-time ($O(n)$) algorithm.

Algorithm 4 (`Persistence`):
The complexity analysis is the same as for Algorithm 3 (`PersistenceSymThinning`).

Algorithm 5 (`PersistenceSSF`):
To obtain the best complexity for this algorithm, a breadth-first strategy must be employed to compute the sets Y, as in Algorithm 3 (`PersistenceSymThinning`). Also, an adapted priority queue data structure must be used to represent the set Q. As the priorities are small integers (step indices), which may be sorted in linear time by counting sort, Q can be managed to perform insertion and min extraction operations in constant time (see [40]). Furthermore, observe that the sets V (lines 14–15) at different iterations of the repeat loop are disjoint, and that for each voxel x there are at most 26 neighboring voxels y (line 17). Overall, the algorithm may be implemented to run in $O(n)$ complexity.

11 Conclusion

We presented two generic parallel thinning schemes acting in the cubic 3D grid. The first scheme is asymmetric, with the aim of producing "thin" skeletons. The second one is symmetric and produces skeletons that are uniquely defined and invariant to 90 degrees rotations.

We showed how these schemes can be used to produce ultimate, curve, or surface skeletons of 3D objects made of voxels, based in particular on the notions of 1- and 2-isthmus. However, it should be noticed that any static constraint set, and any dynamic skeletal voxel detection rule (for example, an application-dependent one) can be used. In any case, topology preservation is ensured by the properties of critical kernels.

We performed some experiments in order to compare our asymmetric curve skeletonization method with all methods of the same class found in the literature. The results show clearly that our method outperforms the other ones with respect to robustness to noise. All our algorithms can be implemented to run in linear time.

Furthermore, we showed that an effective filtering can be easily performed within our framework, thanks to the notion of persistence. In this approach, the filtering is done dynamically, with very little added cost, and is governed by a unique parameter. Persistence is closely linked to the notion of isthmus, and we stress that this kind of filtering cannot be adapted to the other methods considered in our experiments.

References

1. Ronse, C.: Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images. Discrete Applied Mathematics **21**(1) (1988) 67–79
2. Bertrand, G.: On P-simple points. Comptes Rendus de l'Académie des Sciences, Série Math. **I**(321) (1995) 1077–1084
3. Bertrand, G.: On critical kernels. Comptes Rendus de l'Académie des Sciences, Série Math. **I**(345) (2007) 363–367
4. Bertrand, G., Couprie, M.: On parallel thinning algorithms: Minimal non-simple sets, P-simple points and critical kernels. Journal of Mathematical Imaging and Vision **35**(1) (2009) 23–35
5. Manzanera, A., Bernard, T., Prêteux, F., Longuet, B.: n-dimensional skeletonization: a unified mathematical framework. Journal of Electronic Imaging **11**(1) (2002) 25–37
6. Lohou, C., Bertrand, G.: Two symmetrical thinning algorithms for 3D binary images. Pattern Recognition **40** (2007) 2301–2314
7. Palágyi, K.: A 3D fully parallel surface-thinning algorithm. Theoretical Computer Science **406**(1-2) (2008) 119–135
8. Tsao, Y., Fu, K.: A parallel thinning algorithm for 3D pictures. Computer Graphics and Image Processing **17**(4) (1981) 315–331
9. Tsao, Y., Fu, K.: A 3D parallel skeletonwise thinning algorithm pictures. In: Proceedings PRIP 82: IEEE Computer Society Conference on Pattern Recognition and Image Processing, I.E.E.E. (1982) 678–683
10. Palágyi, K., Kuba, A.: A 3d 6-subiteration thinning algorithm for extracting medial lines. Pattern Recognition Letters **19**(7) (1998) 613–627
11. Ma, C.M., Wan, S.Y.: Parallel thinning algorithms on 3D (18,6) binary images. Computer Vision and Image Understanding **80** (2000) 364–378
12. Ma, C.M., Wan, S.Y., Lee, J.D.: Three-dimensional topology preserving reduction on the 4-subfields. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(12) (2002) 1594–1605
13. Ma, C., Wan, S., Chang, H.: Extracting medial curves on 3D images. Pattern Recognition Letters **23**(8) (2002) 895–904
14. Lohou, C., Bertrand, G.: A 3D 12-subiteration thinning algorithm based on P-simple points. Discrete Applied Mathematics **139** (2004) 171–195
15. Lohou, C., Bertrand, G.: A 3D 6-subiteration curve thinning algorithm based on P-simple points. Discrete Applied Mathematics **151** (2005) 198–228

16. Németh, G., Kardos, P., Palágyi, K.: Topology preserving 2-subfield 3D thinning algorithms. In: Proceedings Signal Processing, Pattern Recognition and Applications (SPPRA 2010), ACTA Press (2010) 311–316
17. Németh, G., Kardos, P., Palágyi, K.: Topology preserving 3D thinning algorithms using four and eight subfields. In Campilho, A., Kamel, M., eds.: Image Analysis and Recognition. Volume 6111 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 316–325
18. Bertrand, G., Couprie, M.: Powerful Parallel and Symmetric 3D Thinning Schemes Based on Critical Kernels. Journal of Mathematical Imaging and Vision **48**(1) (2014) 134–148
19. Liu, L., Chambers, E.W., Letscher, D., Ju, T.: A simple and robust thinning algorithm on cell complexes. Computer Graphics Forum **29**(7) (2010) 2253–2260
20. Bertrand, G., Couprie, M.: Isthmus based parallel and symmetric 3D thinning algorithms. Graphical Models **80** (2015) 1–15
21. Couprie, M., Bertrand, G.: Asymmetric parallel 3D thinning scheme and algorithms based on isthmuses. Pattern Recognition Letters **76** (2016) 22–31 Special Issue on Skeletonization and its Application.
22. Kovalevsky, V.: Finite topology as applied to image analysis. Computer Vision, Graphics and Image Processing **46** (1989) 141–161
23. Kong, T.Y., Rosenfeld, A.: Digital topology: introduction and survey. Computer Vision, Graphics and Image Processing **48** (1989) 357–393
24. Kong, T.Y.: Topology-preserving deletion of 1's from 2-, 3- and 4-dimensional binary images. In: Proceedings Discrete Geometry for Computer Imagery. Volume 1347 of Lecture Notes in Computer Science., Springer (1997) 3–18
25. Bertrand, G.: New notions for discrete topology. In: Proceedings Discrete Geometry for Computer Imagery. Volume 1568 of Lecture Notes in Computer Science., Springer (1999) 218–228
26. Bertrand, G., Malandain, G.: A new characterization of three-dimensional simple points. Pattern Recognition Letters **15**(2) (1994) 169–175
27. Bertrand, G.: Simple points, topological numbers and geodesic neighborhoods in cubic grids. Pattern Recognition Letters **15** (1994) 1003–1011
28. Saha, P., Chaudhuri, B., Chanda, B., Dutta Majumder, D.: Topology preservation in 3D digital space. Pattern Recognition **27** (1994) 295–300
29. Kong, T.Y.: On topology preservation in 2-D and 3-D thinning. International Journal on Pattern Recognition and Artificial Intelligence **9** (1995) 813–844
30. Couprie, M., Bertrand, G.: New characterizations of simple points in 2D, 3D and 4D discrete spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(4) (2009) 637–648
31. Whitehead, J.: Simplicial spaces, nuclei and m-groups. Proceedings of the London Mathematical Society **45**(2) (1939) 243–327
32. Bertrand, G., Couprie, M.: Two-dimensional thinning algorithms based on critical kernels. Journal of Mathematical Imaging and Vision **31**(1) (2008) 35–56
33. Blum, H.: A transformation for extracting new descriptors of shape. In Wathen-Dunn, W., ed.: Models for the perception of speech and visual form. MIT Press (1967) 382–380
34. Lieutier, A.: Any open bounded subset of R^{n} has the same homotopy type as its medial axis. Computer-Aided Design **36**(11) (2004) 1029–1046
35. Zeeman, E.: On the dunce hat. Topology **2** (1964) 341–358
36. Bing, R.: Some aspects of the topology of 3-manifolds related to the Poincaré conjecture. Lectures on modern mathematics **II** (1964) 93–128

37. Malgouyres, R., Francés, A.: Deciding whether a simplicial 3-complex collapses to a 1-complex is NP-complete. In: Proceedings Discrete Geometry for Computer Imagery. Volume 4992 of Lecture Notes in Computer Science., Springer (2008) 177–188
38. Shaked, D., Bruckstein, A.M.: Pruning medial axes. Computer Vision and Image Understanding **69**(2) (1998) 156–169
39. Chaussard, J.: Topological tools for discrete shape analysis. Ph.D. dissertation, Université Paris-Est (2010)
40. Thorup, M.: Equivalence between priority queues and sorting. In: Proceedings 43rd Symposium on Foundations of Computer Science. FOCS '02, Washington, DC, USA, IEEE Computer Society (2002) 125–134