January 2004

Making 3D Binary Digital Images Well-Composed

Marcelo Siqueira
University of Pennsylvania

Longin Jan Latecki
Temple University

Jean Gallier
University of Pennsylvania, jean@cis.upenn.edu

Making 3D Binary Digital Images Well-Composed

Abstract

A 3D binary digital image is said to be *well-composed* if and only if the set of points in the faces shared by the voxels of foreground and background points of the image is a surface in R^3. Well-composed images enjoy important topological and geometric properties; in particular, there is only one type of connected component in any well-composed image, as 6-, 14-, 18-, and 26-connected components are equal. This implies that several algorithms used in computer vision, computer graphics, and image processing become simpler. For example, thinning algorithms do not suffer from the irreducible thickness problem if the image is well-composed. In this report, we introduce a new randomized algorithm for making 3D binary digital images that are not well-composed into well-composed ones, analyze its time complexity, and present experimental evidence of its effectiveness when faced with practical medical imaging data. We also introduce a new approach to extract simplicial surfaces from 3D binary images, which is based on our algorithm for making 3D binary digital images well-composed. We show that the extraction of simplicial surfaces from well-composed images using the Marching Cubes (MC) algorithm or some octree-based algorithms can be simplified, as only six out of the fourteen canonical configurations of cube-isosurface intersection can occur. Moreover, the continuous analog of the digital boundary of the input well-composed image and the MC surface are guaranteed to be topologically equivalent.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-04-22.

Making 3D Binary Digital Images Well-Composed

Marcelo Siqueira
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
marcelos@seas.upenn.edu

Longin Jan Latecki
Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
latecki@temple.edu

Jean Gallier
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
jean@cis.upenn.edu

Abstract

A 3D binary digital image is said to be *well-composed* if and only if the set of points in the faces shared by the voxels of foreground and background points of the image is a surface in $\mathbb{R}^3$. Well-composed images enjoy important topological and geometric properties; in particular, there is only one type of connected component in any well-composed image, as 6-, 14-, 18-, and 26-connected components are equal. This implies that several algorithms used in computer vision, computer graphics, and image processing become simpler. For example, thinning algorithms do not suffer from the irreducible thickness problem if the image is well-composed. In this report, we introduce a new randomized algorithm for making 3D binary digital images that are not well-composed into well-composed ones, analyze its time complexity, and present experimental evidence of its effectiveness when faced with practical medical imaging data. We also introduce a new approach to extract simplicial surfaces from 3D binary images, which is based on our algorithm for making 3D binary digital images well-composed. We show that the extraction of simplicial surfaces from well-composed images using the Marching Cubes (MC) algorithm or some octree-based algorithms can be simplified, as only six out of the fourteen canonical configurations of cube-isosurface intersection can occur. Moreover, the continuous analog of the digital boundary of the input well-composed image and the MC surface are guaranteed to be topologically equivalent.

1 Introduction

A special class of 3D digital images, called 3D well-composed images, has been defined in [1]. A well-composed image enjoys very useful topological and geometric properties, such as the fact that there is

only one connectedness relation on points of the image. This implies that several basic algorithms in computer vision, computer graphics, and image processing become simpler when dealing with well-composed images. For instance, thinning algorithms do not suffer from the irreducible thickness problem if the image is well-composed [2]. Thinning well-composed gray-scale images leads not only to theoretically better understood algorithms but also to practically relevant thinning and segmentation algorithms [3]. The property of well-composedness is used as an important tool for proving theoretical results in digital topology [4]. Here, we show that the extraction of isosurfaces from well-composed images using the Marching Cubes (MC) algorithm [5] or some octree-based algorithms [6, 7] can be simplified. On the other hand, if a digital image lacks the property of being well-composed, then the digitization process that gave rise to it is not topology-preserving, as the results in [8] show that if the resolution of the digitization process is fine enough to ensure preservation of topology, then the image must be well-composed. This fact has motivated the development of algorithms for "repairing" 2D and 3D binary digital images that are not well-composed [9, 4].

The idea behind a repairing algorithm is to change the color assigned to some points of the image, so that the resulting image is well-composed. There is no guarantee that the resulting well-composed image is the same as the one that would be obtained by a topology-preserving digitization process. However, if the repairing algorithm changes the color assigned to a few points of the image only, then the image and its well-composed counterpart will still be very similar, which may be entirely satisfactory for several image-based applications that can benefit from using well-composed images. Here, we introduce a new repairing algorithm for making 3D *binary* digital images well-composed. Our algorithm changes the color assigned to background points of the input image only, so that they become foreground points in the resulting well-composed image. The resulting image has the same resolution as the input one. To our best knowledge, this is the first repairing algorithm for 3D binary digital images with this property. Rosenfeld, Kong and Nakamura [4] introduced an image operator, called *simple deformation*, that can be used to making 2D binary digital images well-composed. This operator can be extended to deal with 3D images in a straightforward manner [10], but the resulting well-composed images are 9 and 27 times bigger than the input one in the 2D and 3D cases, respectively.

We also introduce a new approach to extract simplicial surfaces from 3D binary digital images, which is based on our repairing algorithm. By "extracting" a simplicial surface, we mean to consider the image as a sampling of a continuous scalar field on the points of the image, and then generate a simplicial surface that approximates an isosurface of this field defined by a given isovalue. Extraction of simplicial surfaces from imaging data is the first step of a very popular, two-step scheme to visualize imaging data: We extract a simplicial surface that approximates an isosurface corresponding to the boundary between the interior and the exterior of some subset of points of the image, and then we use fast algorithms for rendering triangles to display the approximating simplicial surface. The underlying idea of our new approach to extract simplicial surfaces from 3D binary digital images is to first make the image well-composed, and then apply some existing algorithm for extracting uniform or adaptive simplicial surfaces from the resulting well-composed image. We show that, if our approach is used, then the task

of extracting simplicial surfaces becomes simpler. Furthermore, well-known problems of some popular algorithms for extracting simplicial surfaces from 3D digital images cannot occur. So, our approach offers some practical benefits.

The remaining of this technical report is organized in the following way: Section 2 introduces basic concepts of digital topology that are needed for the description of our algorithm; Section 3 presents basic concepts from piecewise-linear topology and also some lemmas, which are used to define simplicial surfaces and to prove an important result in Section 8. Section 4 formally defines 3D well-composed images and presents some of their most important properties; Section 5 describes our algorithm for making 3D binary digital images well-composed; Section 6 presents the computational complexity of our algorithm and provides a probabilistic argument for its effectiveness; Section 7 reports on the use of our algorithm on several medical digital images; Section 8 introduced our new approach to extract simplicial surfaces from 3D binary digital images; and Section 9 summarizes our results and discusses future work.

2 Digital Topology and Digital Images

This section introduces some basic concepts from "digital topology", which is the field that studies the topological properties of digital images. The material is this section can be found in a book by Herman [11] and in a paper by Latecki [1].

Let X be any set of points in $\mathbb{R}^3$. Then, the *Voronoi Neighborhood* $\mathcal{V}_X(p)$ in $\mathbb{R}^3$ of any element p of X is defined as

$$\mathcal{V}_X(p) = \{x \in \mathbb{R}^3 \mid d(x,p) \leq d(x,r), \forall r \in X\},$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance function between two points in $\mathbb{R}^3$. In other words, the Voronoi neighborhood $\mathcal{V}_X(p)$ of p in X consists of all those points in $\mathbb{R}^3$ which are at least as close to p as to any other point r of X.

As usual in digital topology, we interpret $\mathbb{Z}^3$ as the set of points with integer coordinates in $\mathbb{R}^3$, and we identify each point p of $\mathbb{Z}^3$ with its Voronoi neighborhood $\mathcal{V}_{\mathbb{Z}^3}(p)$ in $\mathbb{Z}^3$. Note that $\mathcal{V}_{\mathbb{Z}^3}(p)$ is the unit upright cube in $\mathbb{R}^3$ centered at p and whose faces are parallel to the coordinate planes. Since this correspondence between points in $\mathbb{Z}^3$ and their Voronoi neighborhoods in $\mathbb{R}^3$ plays an important role in this work, we define it formally as a function $CA : \mathbb{Z}^3 \rightarrow \mathcal{P}(\mathbb{R}^3)$ such that, for every $p \in \mathbb{Z}^3$, $CA(p) = \mathcal{V}_{\mathbb{Z}^3}(p)$, where $\mathcal{P}(\mathbb{R}^3)$ denotes the power set of $\mathbb{R}^3$. We refer to $CA(p)$ as the *continuous analog* or *voxel* of p. We can extend the definition of continuous analog of a point to a set of points as follows: The *continuous analog* $CA(X)$ *of a set* $X \subset \mathbb{Z}^3$ is defined as $CA(X) = \bigcup\{CA(p) \mid p \in X\}$. In this case, CA can be viewed as a function $CA : \mathcal{P}(\mathbb{Z}^3) \rightarrow \mathcal{P}(\mathbb{R}^3)$, where $\mathcal{P}(\mathbb{Z}^3)$ is the power set of $\mathbb{Z}^3$. Figure 1 illustrates the continuous analog $CA(X)$ of a set $X = \{a, b, c, d\}$ consisting of four points, a, b, c, and d, of $\mathbb{Z}^3$.

Let A be any set and ρ be a binary relation on A, i.e., ρ is a subset of $A^2 = A \times A$, the set of all ordered pairs of elements of A. If $(p, q) \in \rho$ then we say that p *is ρ-adjacent to* q. Furthermore, if ρ is a *symmetric* relation, i.e., if for any $p, q \in A$, we have that $(p, q) \in \rho$ if and only if $(q, p) \in \rho$, then we also say that p and q are ρ-adjacent. For $A = \mathbb{Z}^3$, we define three symmetric binary relations on $\mathbb{Z}^3$, α_3, δ_3 and ω_3, as follows: For any two points $p = (p_1, p_2, p_3)$ and $q = (q_1, q_2, q_3)$ of $\mathbb{Z}^3$,

$$(p, q) \in \alpha_3 \text{ if and only if } p \neq q \text{ and, for } 1 \leq i \leq 3,\ |p_i - q_i| \leq 1,$$

$$(p, q) \in \delta_3 \text{ if and only if } (p, q) \in \alpha_3 \text{ and } \sum_{i=1}^{3} |p_i - q_i| \leq 2,$$

and

$$(p, q) \in \omega_3 \text{ if and only if } \sum_{i=1}^{3} |p_i - q_i| = 1.$$

Note that $\omega_3 \subseteq \delta_3 \subseteq \alpha_3$. For instance, points a and b of $\mathbb{Z}^3$ shown in Figure 1 are ω_3-, δ_3-, and α_3-adjacent, points b and c are δ_3- and α_3-adjacent but not ω_3-adjacent, and points c and d are α_3-adjacent but not δ_3- nor ω_3-adjacent. The symmetric binary relations ω_3, δ_3 and α_3 are also denoted as 6, 18, and 26 adjacencies, respectively. For any two distinct points p and q in $\mathbb{Z}^3$, if $(p, q) \in \omega_3$ then we say that $CA(p)$ and $CA(q)$ are *face-adjacent*; if $(p, q) \notin \omega_3$ and $(p, q) \in \delta_3$ then we say that $CA(p)$ and $CA(q)$ are *edge-adjacent*; and if $(p, q) \notin \delta_3$ and $(p, q) \in \alpha_3$ then we say that $CA(p)$ and $CA(q)$ are *corner-adjacent*. So, for the points a, b, c, and d of $\mathbb{Z}^3$ in Figure 1, we have that $CA(a)$ and $CA(b)$ are face-adjacent, $CA(b)$ and $CA(c)$ are edge-adjacent, and $CA(c)$ and $CA(d)$ are corner-adjacent.

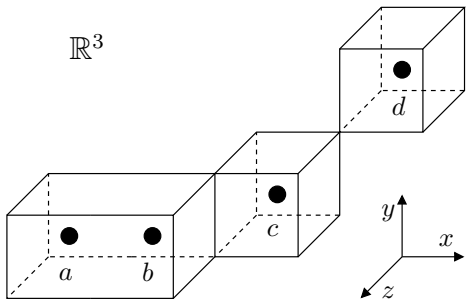

Figure 1: Continuous analog of the set $X = \{a, b, c, d\} \subset \mathbb{Z}^3$.

Let A be any set. For any p and q in A, the sequence $\langle x^{(0)}, \ldots, x^{(k)} \rangle$ of elements of A is said to be a *ρ-path in A connecting p to q* if $x^{(0)} = p$, $x^{(k)} = q$ and, for $1 \leq i \leq k$, $x^{(i-1)}$ is ρ-adjacent to $x^{(i)}$. We call k the *length* of this path. In particular, there are ρ-paths of length zero, e.g., $\langle x \rangle$. We refer to them as *trivial paths*. If there is a ρ-path in A connecting p to q then we say that p *is ρ-connected in A to q*. For instance, if $X = \{a, b, c, d\} \subset \mathbb{Z}^3$ is the set of points in Figure 1, then points a and b are ω_3-, δ_3- and α_3-connected in X, points a and c are δ_3- and α_3-connected in X, but not ω_3-connected, and points a and d are α_3-connected in X, but not ω_3- nor δ_3-connected in X. We say that a subset B of A is a *ρ-connected subset* if, for any p and q in B, we have that p is ρ-connected in B to q. It is easy to see that ρ-connectedness in A is a reflexive and transitive relation on A. If it is also symmetric,

then it is an equivalence relation on A. In this case, the set A can be partitioned into disjoint maximal ρ-connected subsets, called ρ-connected components of A, which are nonempty ρ-connected subsets of A that are not proper subsets of any other ρ-connected subset of A. Note that if ρ is a symmetric relation then ρ-connectedness in A is also a symmetric relation on A, and therefore an equivalence relation on A.

A *digital space* is a pair (A, ρ), where A is an arbitrary nonempty set and ρ is a symmetric binary relation on A such that A is ρ-connected. For our purposes, the set A is always a subset of $\mathbb{Z}^3$ and the symmetric binary relation ρ is either ω_3, δ_3, or α_3. The elements of A are called *spels* (short for spatial elements) and elements of ρ are called *surfels* (short for surface elements). Note that if $A = \mathbb{Z}^3$, then a spel is a point, while a continuous analog of a spel is the voxel in $\mathbb{R}^3$ centered at the point. Any nonempty subset of ρ is called a *digital surface* in (A, ρ). Although ρ is a symmetric binary relation, the fact that a surfel (p, q) is in the digital surface does not imply that (q, p) must be. Actually, we will restrict our attention to *antisymmetric digital surfaces* S in (A, ρ), meaning that if $(p, q) \in S$ then $(q, p) \notin S$. Note that an antisymmetric digital surface is also an anti-reflexive relation; that is, for every spel p, we have that $(p, p) \notin S$.

Let (A, ρ) be a digital space, and let M and N be subsets of A. Then, the *digital boundary in* (A, ρ) *between* M *and* N is defined as $bd_\rho(M, N) = \{(p, q) \in A \mid (p, q) \in \rho,\ p \in M,\ \text{and}\ q \in N\}$. Note that, if $bd_\rho(M, N)$ is not empty, then $bd_\rho(M, N)$ is a digital surface in (A, ρ). Furthermore, if $bd_\rho(M, N)$ is nonempty and M and N are disjoint sets, then $bd_\rho(M, N)$ is clearly an antisymmetric surface. For instance, consider the digital space $(\mathbb{Z}^3, \alpha_3)$, and let X be the singleton set $\{a\} \subseteq \mathbb{Z}^3$ such that $a = (0, 0, 0)$. Then, the boundary $bd_{\omega_3}(X, X^c)$ in $(\mathbb{Z}^3, \alpha_3)$ between X and X^c, where $X^c = \mathbb{Z}^3 \setminus X$, is the antisymmetric digital surface consisting of the surfels (a, b), (a, c), (a, d), (a, e), (a, f), and (a, g), where b, c, d, e, f and g are points of $\mathbb{Z}^3$ with $b = (-1, 0, 0)$, $c = (1, 0, 0)$, $d = (0, -1, 0)$, $e = (0, 1, 0)$, $f = (0, 0, -1)$, and $g = (0, 0, 1)$.

We can also extend the definition of continuous analog to apply to sets of surfels in $(\mathbb{Z}^3, \rho)$ as follows: Let $CA : \mathbb{Z}^3 \times \mathbb{Z}^3 \to \mathcal{P}(\mathbb{R}^3)$ be a function such that $CA((p, q)) = CA(p) \cap CA(q)$, for $p, q \in \mathbb{Z}^3$, and let $CA : \mathcal{P}(\mathbb{Z}^3 \times \mathbb{Z}^3) \to \mathcal{P}(\mathbb{R}^3)$ be a function such that $CA(B) = \bigcup\{CA(b) \mid b \in B\}$, where $B \subseteq \mathbb{Z}^3 \times \mathbb{Z}^3$ and $\mathcal{P}(\mathbb{Z}^3 \times \mathbb{Z}^3)$ is the power set of $\mathbb{Z}^3 \times \mathbb{Z}^3$. For instance, if B is the digital boundary $bd_{\omega_3}(X, X^c)$ in the previous example, then $CA(B)$ is exactly the topological boundary of $CA(a)$ in $\mathbb{R}^3$, i.e., the set of points in the faces of the voxel $CA(a)$. Here, we are only interested in digital boundaries $bd_{\omega_3}(M, N)$ in $(\mathbb{Z}^3, \alpha_3)$ between a set M and its complement set $N = M^c$ when either M or M^c is a nonempty *digital set* of $\mathbb{Z}^3$, i.e., a nonempty and finite subset of $\mathbb{Z}^3$.

By definition, the continuous analog $CA(bd_{\omega_3}(M, M^c))$ of the digital boundary $bd_{\omega_3}(M, M^c)$ in $(\mathbb{Z}^3, \omega_3)$ between M and M^c is the set of points of the faces of the voxels in $CA(M)$ that are shared by a voxel in $CA(M)$ and a voxel in $CA(M^c)$, or equivalently, not in $CA(M)$. We denote $CA(bd_{\omega_3}(M, M^c))$ by $bdCA(M)$. Note that $bdCA(M)$ is just the topological boundary of $CA(M)$ in $\mathbb{R}^3$. Since, for any two points p and q of $\mathbb{Z}^3$, we have that $(p, q) \in bd_{\omega_3}(M, M^c)$ if and only if $(q, p) \in bd_{\omega_3}(M^c, M)$, it follows that $bd_{\omega_3}(M, M^c) \neq bd_{\omega_3}(M^c, M)$, but $bdCA(M) = bdCA(M^c)$.

A *3D (gray-scale) digital image* $\mathcal{I} : D \to V$ is a function from a subset D of points in $\mathbb{Z}^3$ to a subset V of $\mathbb{R}$. We refer to D as *grid* and to the elements in V as *colors*. A *3D binary digital image* is a 3D digital image $\mathcal{I} : D \to V$ in which the set of colors V is the binary set $\{0, 1\}$. We shall denote a binary image $\mathcal{I}$ by the pair (D, X), where D is the grid of $\mathcal{I}$ and X is the set $\{p \in D \mid \mathcal{I}(p) = 1\}$. The sets X and X^c, where $X^c = D \setminus X = \{p \in D \mid \mathcal{I}(p) = 0\}$ is the complement set of X with respect to D, are commonly referred to as *foreground* and *background* points of (D, X), respectively. Since every background point is assigned the color 0 and every foreground point is assigned the color 1, we sometimes denote X by X_1 and X^c by X_0. A binary image (D, X) is typically obtained from a gray-scale digital image $\mathcal{I}' : D \to V'$ as follows: We choose a real number $\alpha \in \mathbb{R}$ and then, for every $p \in D$, we let $\mathcal{I}(p) = 0$ if $\mathcal{I}'(p) \leq \alpha$ and $\mathcal{I}(p) = 1$ otherwise; that is, we define $X = \{p \in D \mid \mathcal{I}'(p) > \alpha\}$ and $X^c = \{p \in D \mid \mathcal{I}'(p) \leq \alpha\}$. This method of generating (D, X) from $\mathcal{I}$ is commonly referred to as "thresholding" in the computer vision and image processing literature.

3 Simplicial Surfaces

This section introduces some basic concepts of piecewise-linear topology and provides a formal definition of simplicial surface. The concepts presented in this section are very useful for proving an important lemma in Section 8, and they can be found in [12].

Let X be a subset of $\mathbb{R}^m$. We say that X is *convex* if for any two points $q, r \in X$, the point $p = (1 - \lambda)q + \lambda r$, with $0 \leq \lambda \leq 1$ and $\lambda \in \mathbb{R}$, is also a point of X. The empty set is trivially convex, every singleton set is convex, and the entire space $\mathbb{R}^m$ is convex. Given any nonempty subset X of $\mathbb{R}^m$, there is a smallest set containing X denoted by $conv(X)$ and called the *convex hull of* X. If X is any subset of $n > 0$ points $x_1, \ldots, x_n$ of $\mathbb{R}^m$ then the set of all *convex combinations* $\sum_{i=1}^{n} \lambda_i x_i$, where $\sum_{i=1}^{n} \lambda_i = 1$ and $\lambda_i \geq 0$, is the convex hull $conv(X)$ of the set X. Given a collection of $n + 1$ affinely independent points in $\mathbb{R}^m$, $V = \{v_0, v_1, \ldots, v_n\}$, the *n-simplex (or simplex)* $\sigma = [v_0, v_1, \ldots, v_n]$ *spanned by* V is the convex hull of V, $conv(V)$. That is, the set of all convex combinations $\sum_{i=0}^{n} \lambda_i v_i$, where $\sum_{i=0}^{n} \lambda_i = 1$ and $\lambda_i \geq 0$, with $0 \leq i \leq n$. The *dimension* of σ, denoted by $dim(\sigma)$, is n. In $\mathbb{R}^m$, the largest number of affinely independent points is $m + 1$, and we have simplices of dimension $0, 1, \ldots, m$. Figure 2 shows examples of the four simplices in $\mathbb{R}^3$. A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

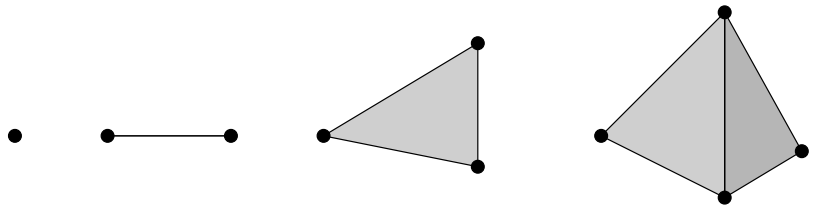

Figure 2: Examples of simplices of dimension 0, 1, 2, and 3 in $\mathbb{R}^3$.

Note that the convex hull of any nonempty subset $V' \subseteq V$ is again a simplex. The simplex spanned by V', say τ, is called a *face* of σ and the inclusion relation is denoted as $\tau \prec \sigma$. If $dim(\tau) = d$ then τ is called an *d-face* of σ. If d is 0 then τ is called a *Vertex*. If d is 1 then τ is called an *edge*. If $n = m$ and $dim(\tau) = m - 1$ then τ is called a *facet* of σ. If $\tau = \sigma$ then τ is an *improper* face, and all others are *proper* faces of σ. The *boundary* $bd(\sigma)$ of a simplex σ is the union of its proper faces. The relative interior of a simplex σ, denoted by $int(\sigma)$, arises by removing its boundary. The number of d-faces of σ is equal to the number of ways we can choose $d + 1$ from $n + 1$ points, which is $\binom{n+1}{d+1}$. Thus, the total number of faces of σ is

$$\sum_{d=0}^{n} \binom{n+1}{d+1} = 2^{n+1} - 1.$$

A *simplicial complex* $\mathcal{K}$ is a finite collection of simplices in $\mathbb{R}^m$ satisfying the following two conditions:

1. If a simplex is in $\mathcal{K}$, then all its faces are in $\mathcal{K}$, i.e., if $\sigma \in \mathcal{K}$ and $\tau \prec \sigma$ then $\tau \in \mathcal{K}$;
2. If $\sigma, \tau \in \mathcal{K}$ are simplices such that $\sigma \cap \tau \neq \emptyset$, then $\sigma \cap \tau$ is a face of each of σ and τ, i.e., $\sigma \cap \tau \prec \sigma$ and $\sigma \cap \tau \prec \tau$.

Figure 3 shows three sets of simplices in $\mathbb{R}^2$. The set on the left is not a simplicial complex because it is missing an edge and a vertex. The set in the middle contains two simplices that intersect each other but the intersection is not a face of either one, and therefore it cannot be a simplicial complex. The set on the right is a simplicial complex.

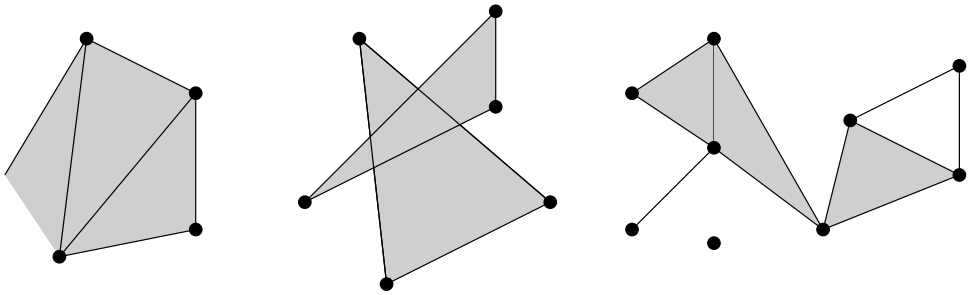

Figure 3: Collections of simplices in $\mathbb{R}^2$. (a) and (b) are not simplicial complexes, but (c) is.

Let $\mathcal{K}$ be a simplicial complex in $\mathbb{R}^m$. The *dimension* $dim(\mathcal{K})$ of $\mathcal{K}$ is the largest dimension of a face in $\mathcal{K}$, i.e., $dim(\mathcal{K}) = \max\{dim(\sigma) \mid \sigma \in \mathcal{K}\}$. Here, we will refer to an n-dimensional simplicial complex as simply an n-complex. The set consisting of the union of all points in the simplices of $\mathcal{K}$ is called the *underlying space* of $\mathcal{K}$, or the *polyhedron* of $\mathcal{K}$, or the *geometric realization* of $\mathcal{K}$, and it is denoted by $|\mathcal{K}|$. Since $\mathcal{K}$ is a finite collection of simplices, its underlying space $|\mathcal{K}|$ is a compact set. A *sub-complex* of a simplicial complex in $\mathcal{K}$ is a subset of $\mathcal{K}$ that is itself a simplicial complex. An important example of a sub-complex is the d-skeleton $\mathcal{K}^{\leq d}$ of a simplicial complex $\mathcal{K}$. It consists of all simplices of $\mathcal{K}$ of

dimension at most d, i.e., $\mathcal{K}^{\leq d} = \{\sigma \in \mathcal{K} \mid dim(\sigma) \leq d\}$. Further we define $\mathcal{K}^d = \{\sigma \in \mathcal{K} \mid dim(\sigma) = d\}$ to be the subset of simplices in $\mathcal{K}$ of dimension d. In particular, $\mathcal{K}^0$ is the set of vertices of $\mathcal{K}$. Note that $\mathcal{K}^d$ is not a simplicial complex. We say that a simplicial complex K' *subdivides* $\mathcal{K}$ if $|\mathcal{K}'| = |\mathcal{K}|$ and if every simplex of $\mathcal{K}'$ is a subset (not necessarily proper) of a simplex of $\mathcal{K}$.

Let τ be a simplex in $\mathcal{K}$. The *star* of τ in $\mathcal{K}$, denoted by $st(\tau, \mathcal{K})$, is the set of all simplices in $\mathcal{K}$ that contain τ. That is,

$$st(\tau, \mathcal{K}) = \{\sigma \in \mathcal{K} \mid \tau \prec \sigma\}.$$

The *link* of τ in $\mathcal{K}$, denoted by $lk(\tau, \mathcal{K})$, is the set of all faces in $st(\tau)$ that do not intersect τ. That is,

$$lk(\tau, \mathcal{K}) = \{\sigma \in st(\tau, \mathcal{K}) \mid \sigma \cap \tau = \emptyset\}.$$

Let $\mathcal{K}$ be the simplicial complex in Figure 4(a). The star $st([v], \mathcal{K})$ of $[v]$ consists of $[v]$ itself, 1-simplices $[r, v]$, $[s, v]$, $[p, v]$, $[t, v]$, $[x, v]$, $[t, v]$, and 2-simplices $[r, s, v]$, $[s, t, v]$, $[t, x, v]$, and $[x, z, v]$, as illustrated by Figure 4(b). The link $lk([v], \mathcal{K})$ of $[v]$ consists of the 0-simplices $[r]$, $[s]$, $[p]$, $[x]$, $[z]$, $[t]$, and 1-simplices $[r, s]$, $[s, p]$, $[z, x]$, and $[x, t]$, as illustrated by Figure 4(c).

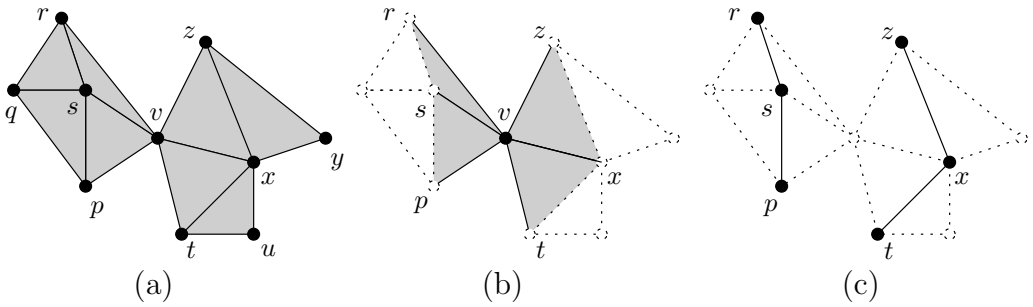

Figure 4: (a) A simplicial complex. (b) Star of the vertex v in (a). (c) Link of vertex v in (a).

Let $\mathcal{K}$ be a simplicial complex in $\mathbb{R}^m$, and let $\mathcal{L}$ be a simplicial complex in $\mathbb{R}^n$. A map $f : \mathcal{K}^0 \to \mathcal{L}^0$ is a *simplicial map* if whenever $[v_0, v_1, \ldots, v_i]$ is an i-simplex of $\mathcal{K}$, then $[f(v_0), f(v_1), \ldots, f(v_i)]$ is an i-simplex of $\mathcal{L}$. A simplicial map is a *simplicial isomorphism* if it is a bijective map on the set of vertices, and its inverse is also a simplicial map. If there is a simplicial isomorphism from $\mathcal{K}$ to $\mathcal{L}$ then we say that $\mathcal{K}$ and $\mathcal{L}$ are *simplicially isomorphic*. For instance, if $\mathcal{K}$ is a tetrahedron, then any map $f : \mathcal{K}^0 \to \mathcal{K}^0$ defines a simplicial map from $\mathcal{K}$ to $\mathcal{K}$, as any collection of two or three vertices of $\mathcal{K}$ are the vertices of a simplex of $\mathcal{K}$. The following lemma from [12] establishes an important relationship between the existence of a simplicial map from a simplicial complex to another and the underlying spaces of the two simplicial complexes:

Lemma 3.1. *Let $\mathcal{K}$ be a simplicial complex in $\mathbb{R}^m$, and let $\mathcal{L}$ be a simplicial complex in $\mathbb{R}^n$. Then, the following are true:*

(a) If $\mathcal{K}$ and $\mathcal{L}$ are simplicially isomorphic, then $|\mathcal{K}|$ and $|\mathcal{L}|$ are homeomorphic.

(b) If $\mathcal{K}$ and $\mathcal{L}$ have simplicially isomorphic subdivisions, then $|\mathcal{K}|$ and $|\mathcal{L}|$ are homeomorphic.

A 2-complex $\mathcal{K}$ is called a *simplicial surface* if $\mathcal{K}$ is a 2-complex such that each 1-simplex of $\mathcal{K}$ is the face of precisely two simplices, and the underlying space of the link of each 0-simplex of $\mathcal{K}$ is homeomorphic to the 1-sphere, $S^1 = \{x \in \mathbb{R}^2 | \|x\| = 1\}$. For instance, the simplicial complex consisting of all proper faces of a tetrahedron is a simplicial surface. However, the simplicial complex consisting of all proper faces of the two tetrahedra illustrated by Figure 5 is not a simplicial surface, as the link of $[v]$ is not homeomorphic to S^1. The underlying space of a simplicial surface is called the *underlying surface* of the simplicial surface. Recall that a subset $X \subset \mathbb{R}^m$ is called a topological surface, or just surface for short, if each point $p \in X$ has a neighborhood that is homeomorphic to the open unit disk $D = \{x \in \mathbb{R}^2 \mid \|x\| < 1\}$. The following lemma from [12] states that the underlying surface of a simplicial surface is a topological surface:

Lemma 3.2. *Let $\mathcal{K}$ be a simplicial complex in $\mathbb{R}^m$. Then $|\mathcal{K}|$ is a topological surface if and only if $\mathcal{K}$ is a simplicial surface.*

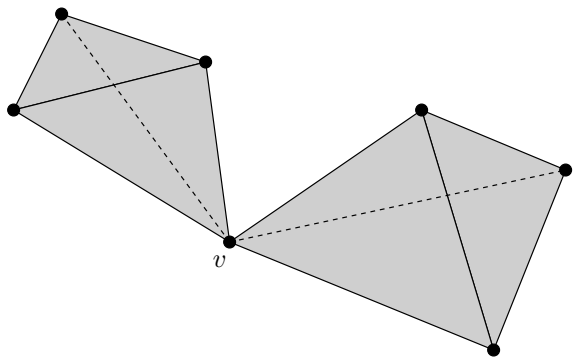

Figure 5: The 2-complex consisting of the proper faces of the two tetrahedra is not a simplicial surface.

4 Well-Composedness

Let $(\mathbb{Z}^3, X)$ be a 3D binary digital image, and assume that either X or its complement $X^c = \mathbb{Z}^3 \setminus X$ is a digital set. Now, consider the following definition of a 3D well-composed, binary digital image in [1]:

Definition 4.1. *A 3D binary digital image $(\mathbb{Z}^3, X)$ is* well-composed *if $bdCA(X)$ is a surface in $\mathbb{R}^3$.*

From the above definition, it is not easy to think of any property that the points in X or X^c must satisfy in order to have that $(\mathbb{Z}^3, X)$ is well-composed. However, well-composedness is indeed equivalent to two simple and local conditions involving voxels of $CA(X)$ and $CA(X^c)$ that contain a face in $bdCA(X)$. Let Y be a any subset of four points of $\mathbb{Z}^3$. We say that Y is an instance of the *critical configuration (C1) in* $(\mathbb{Z}^3, X)$ if the voxels of the four points of Y share an edge, two of them are in $CA(X)$ and the other two are in $CA(X^c)$, and the two voxels in $CA(X)$ (resp. $CA(X^c)$) are edge-adjacent. Figure 6(a)

illustrates the voxels of the points of Y when Y is an instance of the critical configuration (C1). Now, let Y be a any subset of eight points of $\mathbb{Z}^3$. We say that Y is an instance of the *critical configuration (C2) in* $(\mathbb{Z}^3, X)$ if the voxels of the eight points of Y share a vertex, two of them are in $CA(X)$ (resp. $CA(X^c)$) and the other six are in $CA(X^c)$ (resp. $CA(X)$), and the two voxels in $CA(X)$ (resp. $CA(X^c)$) are corner-adjacent. Figure 6(b) illustrates the voxels of the points of Y when Y is an instance of the critical configuration (C2).

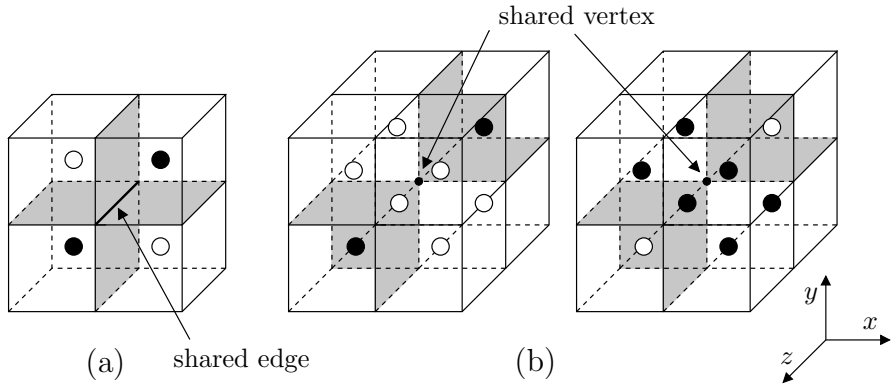

Figure 6: (a) Example of critical configuration (C1) in $(\mathbb{Z}^3, X)$. (b) Example of critical configuration (C2) in $(\mathbb{Z}^3, X)$. Points in X are represented by black circles, while points in X^c are represented by white circles.

The following theorem proved by Latecki in [1] establishes an important equivalence between the well-composedness property of a 3D binary digital image and the nonexistence of instances of critical configurations (C1) and (C2) in the image:

Theorem 4.1. *A 3D binary digital image $(\mathbb{Z}^3, X)$ is well-composed if and only if the critical configurations (C1) and (C2) with respect to X do not occur in $\mathbb{Z}^3$.*

Note that it is a straightforward task to determine if a binary image is well-composed based on the results from Theorem 4.1, as we just have to verify if there is any instance of (C1) and (C2) in $(\mathbb{Z}^3, X)$. Note also that $(\mathbb{Z}^3, X)$ is well-composed if and only if $(\mathbb{Z}^3, X^c)$ is well-composed, as $bdCA(X) = bdCA(X^c)$. Theorem 4.1 also implies that there is only one kind of connectedness in well-composed images, i.e., if $(\mathbb{Z}^3, X)$ is well-composed then any α_3-connected component of X or X^c is also a δ_3-connected component, which in turn is also a ω_3-connected component. This is due to the fact that, if a δ_3-connected component of X is not ω_3-connected, then there must exist an instance of (C1) in $(\mathbb{Z}^3, X)$. Similarly, if a α_3-connected component of X is not ω_3-connected, then there must exist an instance of (C2) in $(\mathbb{Z}^3, X)$. Finally, it has also been shown in [1] that, if $(\mathbb{Z}^3, X)$ is well-composed, then every connected component of $bdCA(X)$ is a *simple closed surface* or *Jordan surface*, i.e., a connected surface in $\mathbb{R}^3$. Furthermore, each connected component of $bdCA(X)$ separates the points of a ω_3-component Y of $\mathbb{Z}^3$, such that $X \subseteq Y$, from the points of Y^c, which is also a ω_3-component. The digital boundary $bd_{\omega_3}(Y, Y^c)$ in $(\mathbb{Z}^3, \alpha_3)$ is called a Jordan (digital) surface by Herman [11], as it captures the notion of a

Jordan surface in $\mathbb{R}^3$.

5 Our Repairing Algorithm

We now describe our new algorithm for repairing 3D binary digital images. Our algorithm takes as input a 3D binary digital image (D, X) such that D is assumed to be a finite rectilinear grid of points of $\mathbb{Z}^3$, $D = [d_{11}, d_{12}] \times [d_{21}, d_{22}] \times [d_{31}, d_{32}]$, where $d_{i1}, d_{i2} \in \mathbb{Z}$ and $d_{i1} < d_{i2}$, for $1 \leq i \leq 3$. The output is a 3D well-composed, binary digital image (D, X') such that $X = X'$ if (D, X) is already well-composed, and $X \subset X'$ otherwise. In other words, if (D, X) is not well-composed, then the foreground X'_1 of (D, X') is a proper superset of the foreground X_1 of (D, X), or equivalently, the background X'_0 of (D, X') is a proper subset of the background X_0 of (D, X). The set X' is computed by finding a subset P of X_0 such that the image (D, X'), with $X' = X \cup P$, is well-composed. So, the set P can be viewed as the subset of the set X_0 of background points of (D, X) whose assigned colors are changed from 0 to 1 to produce (D, X'). Observe that such a set P always exists, as (D, X') is well-composed if we let $P = X_0$. However, we ideally would like to find a smallest or *optimal* set P, with $P \subseteq X_0$, such that (D, X') is well-composed.

Since D is a finite set, the background X_0 of (D, X) is also finite. Hence, there is a very simple algorithm for finding a smallest P: Enumerate and test all subsets P of X_0 in increasing order of cardinality until a well-composed image $(D, X' = X \cup P)$ is found. Recall that we can determine if $(D, X' = X \cup P)$ is well-composed by checking the existence of an instance of (C1) or (C2) in $(\mathbb{Z}^3, X')$. But, because X_0 has $2^{|X_0|}$ subsets, where X_0 is the cardinality of X_0, and X_0 has typically one or two million points in practical applications, this algorithm is impractical. By realizing that critical configurations are *local* configurations of points of D, we devised a better alternative to this naïve solution. Our algorithm is not guaranteed to find a smallest P such that $(D, X' = X \cup P)$ is well-composed, but our experiments in Section 7 show evidences that it is very effective on practical data, and Section 6 provides a formal proof for its effectiveness.

Our algorithm starts by letting $P = \emptyset$, and then it iteratively inserts points from $X_0 \setminus P$ into P one at a time. Every point inserted into P eliminates at least one instance of a critical configuration in $(D, X \cup P)$. However, the insertion of a point may also give rise to *new* critical configurations in D, which will trigger the insertion of at least one more point from $X_0 \setminus P$ into P. Before we give details of our repairing algorithm, we describe which points from $X_0 \setminus P$ the algorithm chooses to insert into P in order to eliminate *one* instance of (C1) or (C2) from $(D, X \cup P)$.

First, consider an instance Y of (C1) in $(D, X \cup P)$, and refer to Figure 6(a). The set Y has four points, two of them, say a and b, are in $X_0 \setminus P$ and two of them are in $X \cup P$. We notice that the insertion of either a or b into P eliminates Y from $(D, X \cup P)$. On the other hand, points a and b are the only ones from $X_0 \setminus P$ that can be inserted into P in order to eliminate Y from $(D, X \cup P)$. Now, consider

an instance Y of (C2) in $(D, X \cup P)$, and refer to Figure 6(b). The set Y has eight points, two of them are in $X_0 \setminus P$ (resp. $X \cup P$) and the other six are in $X \cup P$ (resp. $X_0 \setminus P$). Consider the case in which Y has exactly two points in $X_0 \setminus P$, say a and b, as the right cube shown by Figure 6(b) We notice that the insertion of either a or b into P eliminates Y from $(D, X \cup P)$. Furthermore, points a and b are the only ones from $X_0 \setminus P$ that can be inserted into P in order to eliminate Y from $(D, X \cup P)$. Finally, consider the case in which the critical configuration Y has six points in $X_0 \setminus P$. We first notice that the insertion of any of these points into P eliminates Y from $(D, X \cup P)$. However, this insertion always gives rise to an instance of (C1) in $(D, X \cup P)$. This situation is illustrated in Figure 7. On the other hand, only the insertion of one of the six points in $Y \cap (X_0 \setminus P)$ can eliminate Y from $(D, X \cup P)$.

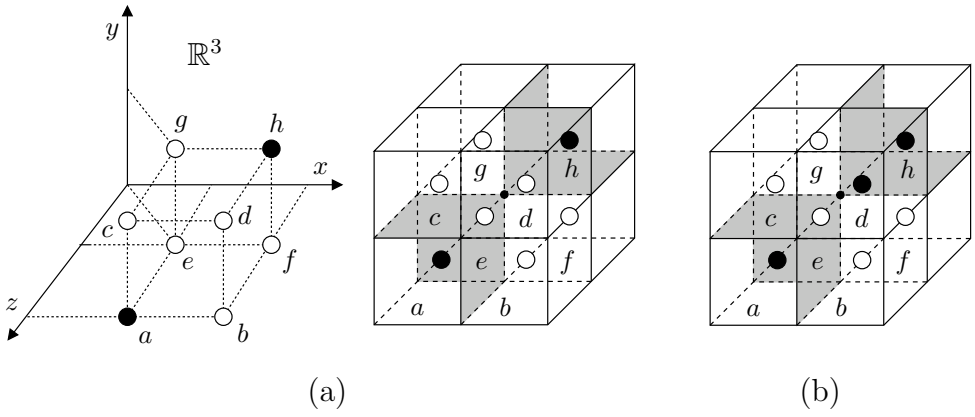

Figure 7: (a) $Y = \{a, b, c, d, e, f, g, h\}$ is an instance of (C2) in $(D, X \cup P)$. The six points in $Y \cap (X_0 \setminus P)$ are $\{b, c, d, e, f, g\}$. (b) We can eliminate Y from $(D, X \cup P)$ by inserting d into P. However, this elimination gives rise to an instance of (C1) in $(D, X \cup P)$, where P is now a set containing the point d.

Our algorithm computes the set $P \subseteq X_0$, such that $(D, X' = X \cup P)$ is well-composed, in two steps. First, the algorithm loops over all points in D to find all instances of (C1) and (C2) in (D, X). Next, the algorithm eliminates all critical configurations found in the first step and the ones that may eventually arise during this elimination process. We now describe the first step. Assume that P is initially empty and recall that $D = [d_{11}, d_{12}] \times [d_{21}, d_{22}] \times [d_{31}, d_{32}]$, where $d_{i1}, d_{i2} \in \mathbb{Z}$ and $d_{i1} < d_{i2}$, for $1 \leq i \leq 3$. So, for each $j \in \{d_{11}, \ldots, d_{12}\}$, $k \in \{d_{21}, \ldots, d_{22}\}$, and $l \in \{d_{31}, \ldots, d_{32}\}$, we consider the point $r \in D$ with integer coordinates $r = (j, k, l)$, and we define the sets $\mathcal{N}_x(r) = \{(j, y, z) \mid y \in \{k, k+1\} \text{ and } z \in \{l, l+1\}\}$, $\mathcal{N}_y(r) = \{(x, k, z) \mid x = \{j, j+1\} \text{ and } z \in \{l, l+1\}\}$, $\mathcal{N}_z(r) = \{(x, y, l) \mid x = \{j, j+1\} \text{ and } y \in \{k, k+1\}\}$, and $\mathcal{N}(r) = [j, j+1] \times [k, k+1] \times [l, l+1]$ of points of $\mathbb{Z}^3$. Note that each instance of (C1) in $(D, X \cup P)$ is one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$, for some $r \in D$. Likewise, each instance of (C2) in $(D, X \cup P)$ is the set $\mathcal{N}(r)$, for some $r \in D$. So, in order to find all instances of (C1) and (C2) in (D, X), the algorithm considers the sets $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$, for every $r \in D$. If any of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration, the algorithm inserts r into a list Q, which is empty in the beginning of the first step.

If Q is empty in the end of the first step then the input image (D, X) is already well-composed, and

the algorithm outputs (D, X) and terminates. Otherwise, the second step takes place. The algorithm removes one point r from Q at a time, and then eliminates *all* instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$, and the instance of (C2) in $\mathcal{N}(r)$, if any. By examining Figure 6, we can see that if $\mathcal{N}(r)$ is an instance of (C2) in $(D, X \cup P)$, then none of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$ and $\mathcal{N}_z(r)$ is an instance of (C1) in $(D, X \cup P)$. Conversely, if at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$ and $\mathcal{N}_z(r)$ is an instance of (C1) in $(D, X \cup P)$, then $\mathcal{N}(r)$ is not an instance of (C2) in $(D, X \cup P)$. So, for each $r \in Q$, there are three mutually exclusive cases: None of the sets $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration in $(D, X \cup P)$, or at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) in $(D, X \cup P)$, or $\mathcal{N}(r)$ is an instance of (C2) in $(D, X \cup P)$.

Consider the case in which at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) in $(D, X \cup P)$, i.e., assume that the set $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains one, two or three instances of (C1) in $(D, X \cup P)$. We saw before that an instance Y of (C1) in $(D, X \cup P)$ can be eliminated from $(D, X \cup P)$ if and only if one of the two points of $Y \cap (X_0 \setminus P)$ is inserted into P. So, if $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly one instance Y of (C1) in $(D, X \cup P)$, then our algorithm chooses one out of two points to insert into P in order to eliminate Y from $(D, X \cup P)$. If $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly two instances of (C1) in $(D, X \cup P)$, say Y_1 and Y_2, then note that the set $Y_1 \cup Y_2$ has exactly three points of $X_0 \setminus P$, not four, as any two instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ share two points, one of which is a point of $X_0 \setminus P$ and the other is a point of $X \cup P$. Figure 8 illustrates all possible cases in which the set $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ has exactly two instances of (C1) in $(D, X \cup P)$. Note that, for each of the six cases in Figure 8, our repairing algorithm eliminates the critical configurations Y_1 and Y_2 from $(D, X \cup P)$ by inserting into P either the point of $X_0 \setminus P$ shared by Y_1 and Y_2 or the other two points of Y_1 and Y_2 from the set $(Y_1 \cup Y_2) \cap (X_0 \setminus P)$.

Finally, if $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains three instances of (C1) in $(D, X \cup P)$, then each of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) and $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ has either three or four points of $X_0 \setminus P$, as illustrated by Figure 9. If $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly three points of $X_0 \setminus P$, then our algorithm inserts two of these three points into P to eliminate all instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$. Otherwise, our algorithm inserts either the common point of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ in $X_0 \setminus P$ or the other three points of $(\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)) \cap (X_0 \setminus P)$ into P. In either case, all instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ are eliminated from $(D, X \cup P)$. Now, consider the case in which $\mathcal{N}(r)$ is an instance of (C2) in $(D, X \cup P)$. As shown in Figure 6(b), the set $\mathcal{N}(r)$ contains either two or six points of $X_0 \setminus P$. In either case, our repairing algorithm inserts only one of these points into P to eliminate the instance of (C2) in $\mathcal{N}(r)$ from $(D, X \cup P)$. Recall that, if the set $\mathcal{N}(r)$ contains six points of $X_0 \setminus P$, then the insertion of any of these six points into P always gives rise to a new critical configuration in $(D, X \cup P)$.

Our repairing algorithm chooses one or more points to insert into P to eliminate all instances of (C1) in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ or the instance of (C2) in $\mathcal{N}(r)$ according to three rules. These rules are mutually exclusive. Before we describe the rules, we define three sets of subsets of points of $(X_0 \setminus P) \cap \mathcal{N}(r)$

used in our description of the rules: $B(r)$, $B_1(r)$, and $B_2(r)$. We define $B(r)$ as the set of all subsets of points of $X_0 \setminus P$ that our algorithm can insert into P in order to eliminate all instances of (C1) or (C2) in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$. So,

- If $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly one instance Y of (C1) in $(D, X \cup P)$, then $B(r) = \{\{a\}, \{b\}\}$, where a and b are the two points of $Y \cap (X_0 \setminus P)$.
- If $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly two instances of (C1) in $(D, X \cup P)$, say Y_1 and Y_2, then $B(r) = \{\{a\}, \{b, c\}\}$, where a is the common point of Y_1 and Y_2 and $X_0 \setminus P$, and b and c are the two other points of $X_0 \setminus P$ in $Y_1 \cup Y_2$ (see Figure 8).
- If $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains three instances of (C1) in $(D, X \cup P)$, then we have two situations: If the common point a of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is in $X_0 \setminus P$, then $B(r) = \{\{a\}, \{b, c, d\}\}$, where b, c, and d are the other points of $(\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)) \cap (X_0 \setminus P)$ (see Figure 9, case (1)). Otherwise, $B(r) = \{\{b, c\}, \{b, d\}, \{c, d\}\}$, where b, c, and d are the points of the set $(\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)) \cap (X_0 \setminus P)$ (see Figure 9, case (2)).
- Finally, if $\mathcal{N}(r)$ contains an instance of (C2) in $(D, X \cup P)$, then we also have two situations: If $\mathcal{N}(r)$ has exactly two points a and b of $X_0 \setminus P$, then $B(r) = \{\{a\}, \{b\}\}$. Otherwise, $B(r) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}\}$, where a, b, c, d, e, and f are the six points of $\mathcal{N}(r) \cap (X_0 \setminus P)$.

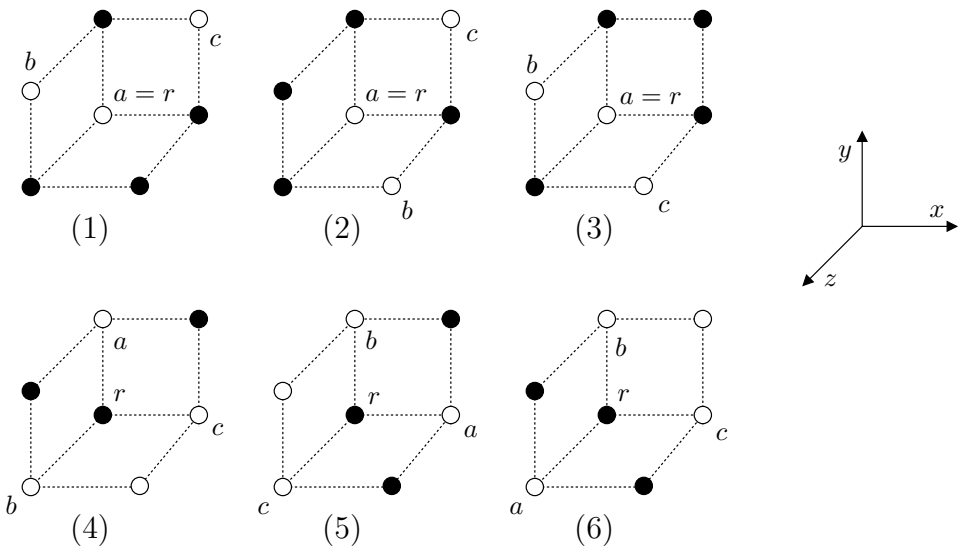

Figure 8: All possible cases in which the set $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains exactly two instances of (C1) in $(D, X \cup P)$. Points in $X \cup P$ are represented by black circles, while points in $X_0 \setminus P$ are represented by white circles. Our repairing algorithm inserts either point a or points b and c into P to eliminate the two instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ from $(D, X \cup P)$.

We also define the partition $B_1(r)$ and $B_2(r)$ of $B(r)$ such that $S \in B_1(r)$ if and only if $S \in B(r)$ and the insertion of all elements of S into P does not give rise to any critical configuration in $(D, X \cup P)$,

and $S \in B_2(r)$ if and only if $S \in B(r)$ and the insertion of all elements of S into P gives rise to at least one critical configuration in $(D, X \cup P)$. Now, for each $r \in D$, if one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration, our repairing algorithm chooses a set S from $B(r)$ according to the following three rules:

(1) If $B_1(r)$ is a singleton set, then let the set S be the single element of $B_1(r)$ and insert the elements of S into P.

(2) If $B_1(r)$ has two or more elements, then select a set S from $B_1(r)$ at random and insert the elements of S into P.

(3) If $B_1(r)$ is the empty set, or equivalently, if $B_2(r) = B(r)$, then select a set S from $B_2(r)$ at random and insert the elements of S into P. Furthermore, insert into list Q all points $p \in D$ such that at least one of the sets $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$ is a critical configuration created by inserting the points of S into P.

Note that any point $p \in D$ inserted into Q by rule (3) must be contained in $\mathcal{N}(r)$, as $S \subset \mathcal{N}(r)$. So, the sets $\mathcal{N}_x(p), \mathcal{N}_y(p), \mathcal{N}_z(p)$, and $\mathcal{N}(p)$ are all contained in the grid $[k-1, k+2] \times [j-1, j+2] \times [l-1, l+2] \subset \mathbb{Z}^3$, where (j, k, l) are the integer coordinates of point r. So, our algorithm can find all points $p \in D$ to insert into Q by looking for new critical configurations created by the insertion of the elements of S into P inside the grid $[k-1, k+2] \times [j-1, j+2] \times [l-1, l+2]$.

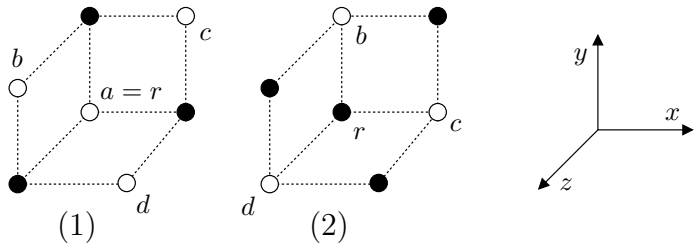

Figure 9: The two cases in which the set $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains three instances of (C1) in $(D, X \cup P)$. Points in $X \cup P$ are represented by black circles, while points in $X_0 \setminus P$ are represented by white circles. In case (1), our algorithm inserts either a or b and c into P to eliminate the three instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ from $(D, X \cup P)$. In case (2), either b and c, or b and d, or c and d are inserted into P.

Since critical configurations may share points in $X_0 \setminus P$, the insertion of the points of S into P may actually eliminate more than the critical configurations found in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$. So, by the time the algorithm picks the next point r from Q, it is possible that none of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration, even though at least one of them was. Otherwise, point r would not have been inserted into Q. In our experiments with medical digital images (see Section 7), we noticed that this situation occurred very often.

Recall that, in the beginning of the second step, the list Q contains the point $r \in D$ if and only if at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) or $\mathcal{N}(r)$ is an instance of (C2) in the input image (D, X). Each iteration of the second step removes one point r from Q, and eliminates all instances of (C1) in $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ and the instance of (C2) in $\mathcal{N}(r)$, if any. This elimination process may give rise to new critical configurations in $(D, X \cup P)$. Each instance of a new critical configuration is a set $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, or $\mathcal{N}(p)$, for some $p \in D$. Rule (3) inserts p into Q whenever one of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$ and $\mathcal{N}_z(p)$ is an instance of (C1) or $\mathcal{N}(p)$ is an instance of (C2) in $(D, X \cup P)$ after the insertion of the elements of S into P. So, at any given time, the set $\bigcup_{r \in Q}\{\mathcal{N}_x(r), \mathcal{N}_y(r), \mathcal{N}_z(r), \mathcal{N}(r)\}$ is a superset of the set of critical configurations in $(D, X \cup P)$. Hence, if Q is empty, the binary image $(D, X' = X \cup P)$ must be well-composed. Our algorithm ends the second step when Q becomes empty.

We now must show that Q eventually becomes empty. Note that this termination condition implies the correctness of our algorithm, as $(D, X' = X \cup P)$ is well-composed if Q is empty. In fact, the list Q eventually becomes empty, and this fact follows from two simple observations. First, the set D is a finite set, which implies that the set X_0 of background points of (D, X) is also finite. Second, the algorithm inserts at least one point from $X_0 \setminus P$ into P in each iteration of the second step, and it never removes points from P. Since there are finitely many points in X_0, our algorithm cannot insert points into P indefinitely. So, after a finite number of iterations, no more points are inserted into Q. Since the algorithm removes one point from Q in each iteration, the list Q will eventually become empty, and therefore our algorithm always terminates and produces a well-composed image. In principle, the set P obtained by our algorithm may be the entire set X_0 of background points. However, as we show in the next section, it is highly unlikely that our repairing algorithm will obtain a set P such that $P = X_0$.

Our repairing algorithm is used by an approach to extract simplicial surfaces from 3D digital images described in Section 8. In this approach, we deal with binary images (D, X) in which the "boundary" points of D are either all background points or all foreground points of (D, X). By "boundary" points of D, we mean the subset of points $p = (p_1, p_2, p_3) \in D$ such that at least one of the following is true: $p_1 \in \{d_{11}, d_{12}\}$, $p_2 \in \{d_{21}, d_{22}\}$, and $p_3 \in \{d_{31}, d_{32}\}$. Note that if this property holds, then (D, X) does not contain any instance of the critical configuration (C1) involving only boundary points of D, and no instance of the critical configuration (C2) involving a boundary point of D. Since our repairing algorithm eliminates critical configurations by selecting points in the critical configuration, if all boundary points of D are background points, then they will also be in the background of the well-composed image (D, X') generated by our repairing algorithm on input (D, X). This observation is very important to prove some results in Section 8.

6 Complexity and Effectiveness

The first step of our repairing algorithm enumerates the sets $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$, for each $r \in D$, and determines if they are instances of a critical configuration in D with respect to X. So, the

time complexity of the first step of our algorithm is $\mathcal{O}(|D|)$, where $|D|$ is the cardinality of D. The second step is also a loop whose number of iterations is the number n of points inserted into Q during the entire execution of the algorithm. Since each point r inserted into Q corresponds to at least one instance of a critical configuration in $(D, X \cup P)$, the number n is bounded by the number of critical configurations in $(D, X \cup P)$ during the entire execution of our algorithm, which in turn is no larger than $|D|$. So, the time complexity of the second step is also $\mathcal{O}(|D|)$, which implies that our algorithm is linear in the number of points of D.

Our algorithm is not guaranteed to obtain a smallest set P such that $(D, X' \cup P)$ is well-composed. Besides, the set P computed by our algorithm can in principle be much larger than the smallest possible size. Since P is a subset of X_0 and any instance of a critical configuration must contain at least two points of $X_0 \setminus P$, the size of the set P computed by *any* repairing algorithm that only adds points from $X_0 \setminus P$ to P is at most $|X_0| - 1$, where $|X_0|$ is the cardinality of X_0. Since the points inserted into P by our algorithm in each iteration of the second step eliminates at least one critical configuration of $(D, X \cup P)$, the size of the set P computed by our algorithm is also bounded above by $m + t$, where m is the number of critical configurations in (D, X) and t is the number of new critical configurations generated by our algorithm in order to produce the resulting well-composed image $(D, X' = X \cup P)$. So, we can have an idea of how effective our repairing algorithm is by providing an upper bound for the size of P in terms of m and t. On deriving this upper bound, we view m as an intrinsic feature of the input image and t as an intrinsic feature of our algorithm. Hence, we express the number t of new critical configurations in terms of m. Since our algorithm is randomized, we actually derive an expression for the *expected value* of t in terms of m.

From the description of our algorithm in Section 5, a new critical configuration is created if and only if rule (3) is applied to eliminate an existing critical configuration. We also know that any critical configuration in (D, X) or any new critical configuration created by our repairing algorithm is one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$, for some $r \in D$. Since $\mathcal{N}_x(r_1) \neq \mathcal{N}_x(r_2)$, $\mathcal{N}_y(r_1) \neq \mathcal{N}_y(r_2)$, $\mathcal{N}_z(r_1) \neq \mathcal{N}_z(r_2)$, and $\mathcal{N}(r_1) \neq \mathcal{N}(r_2)$, for any two distinct points r_1 and r_2, we can express t as $\sum_{r \in D} C(r)$, where $C(r)$ is the number of new critical configurations resulting from the elimination of *all* instances of (C1) in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, or the instance of (C2) in $\mathcal{N}(r)$, if any. If we think of $C(r)$ as a random variable, then the expected value $E[t]$ of t is $E[t] = E[\sum_{r \in D} C(r)] = \sum_{r \in D} E[C(r)]$, where $E[C(r)]$ is the expected value of $C(r)$. By definition, the expected value $E[C(r)]$ of $C(r)$ is given by $E[C(r)] = \sum_{i=1}^{K} i \cdot P[C(r) = i]$, where $P[C(r) = i]$ is the probability that $C(r)$ will have value i, and K is the largest value of $C(r)$. So, if we can obtain an expression for $P[C(r) = i]$, for each $1 \leq i \leq K$, we can provide an expression for $E[C(r)]$ and $E[t]$.

Suppose that rule (3) is used to eliminate an instance of (C1) in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, or the instance of (C2) in $\mathcal{N}(r)$, for some $r \in D$. Let S be the set of points in $B(r)$ selected by rule (3) and whose elements are inserted into P. We know that every point $q \in S$ is a point of $\mathcal{N}(r)$, and that every new critical configuration created by the insertion of q into P is inside the set $G(r) = [j - 1, j + 2] \times [k -$

$1, k+2] \times [l-1, l+2]$ of points of $\mathbb{Z}^3$, where (j, k, l) are the integer coordinates of point r. Note that $G(r)$ is a $4 \times 4 \times 4$ grid of points of $\mathbb{Z}^3$ whose interior points are exactly the points of $\mathcal{N}(r)$. The previous observations imply that we can compute $P[C(r) = i]$ by restricting our attention to the set $G(r)$, which contains exactly 64 points.

Let A be the set of all distinct binary images $(G(r), Y)$. Since each point in $G(r)$ is assigned a color from the binary set $\{0, 1\}$, the cardinality $|A|$ is 2^{64}. We are interested in the subset A' of A such that $(G(r), Y) \in A'$ if and only if $(G(r), Y) \in A$ and at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) in $(G(r), Y)$ or $\mathcal{N}(r)$ is an instance of (C2) in $(G(r), Y)$. There are exactly 84 ways of assigning a color from the set $\{0, 1\}$ to the points in $[j, j+1] \times [k, k+1] \times [l, l+1]$ such that at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) in $(G(r), Y)$ or $\mathcal{N}(r)$ is an instance of (C2) in $(G(r), Y)$. This implies that the cardinality $|A'|$ of the set A' is $|A'| = 2^{56} \cdot 84$. If we assume that each of the $2^{56} \cdot 84$ binary images $(G(r), Y)$ of A' are equally likely to occur in any subset $G(r)$ of D, then we can define the probability $P((G(r), Y)) = 1/|A'| = 1/(2^{56} \cdot 84)$ that the points of $G(r) \subset D$ of (D, X) will be assigned the colors in $(G(r), Y) \in A'$, given that at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, and $\mathcal{N}_z(r)$ is an instance of (C1) in (D, X) or $\mathcal{N}(r)$ is an instance of (C2) in (D, X).

Let $(G(r), Y)$ be any binary image of A'. Since $(G(r), Y) \in A'$, either the set $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ contains one, two or three instances of (C1) or the set $\mathcal{N}(r)$ is an instance of (C2). Recall that our repairing algorithm eliminates *all* critical configurations of $(G(r), Y)$ in either $\mathcal{N}_x(r) \cup \mathcal{N}_y(r) \cup \mathcal{N}_z(r)$ or $\mathcal{N}(r)$ by selecting a set S of $B(r)$ and inserting all elements of S into P. Recall also that rule (3) is used to select S if and only if, for any $S \in B(r)$, the insertion of the elements of S into P gives rise to at least one critical configuration in $(D, X \cup P)$. So, if we assume that each $S \in B(r)$ is equally likely to be selected by rule (3), then we can define the probability $P(S)$ that S will be chosen from B by rule (3) as $P(S) = 0$ if $B_2(r) \neq B(r)$ and $P(S) = 1/|B_2(r)|$ otherwise, where $|B_2(r)|$ is the cardinality of the set $B_2(r)$. Since the set $B(r)$ is not empty whenever one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration in $(D, X \cup P)$, the probability $P(S)$ is well-defined. Now, we can define the probability that the insertion of the elements of S into P will give rise to exactly i critical configurations by $\sum_{S \in B(r)} P(S) \cdot N(S, Y, i)]$, where $N(S, Y, i)$ is 1 if the insertion of the elements of S into P gives rise to exactly i critical configurations, and 0 otherwise.

Since any set $S \in B(r)$ contains at most 3 elements, and instances of (C1) and (C2) are mutually exclusive in the same set $\mathcal{N}(p)$, for any $p \in D$, and each point q in S can give rise to at most 12 instances of (C1) and 8 instances of (C2), we have that the largest value K of $C(r)$ is no larger than 36. Furthermore, since the set $G(r)$ has only 64 points, we can easily build a computer program to obtain $N(S, Y, i)$ for each binary image $(G(r), Y)$ in A', for each set S in $B(r)$, and for each i, with $1 \leq i \leq K = 36$. Now, note that the probability $P[C(r) = i]$ that, for a given $r \in D$, the random variable $C(r)$ will have value i is precisely the sum of the probabilities $P((G(r), Y))[\sum_{S \in B(r)} P(S) \cdot N(s, Y, i)]$ that the elimination of all instances of (C1) in $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, or the instance of (C2) in $\mathcal{N}(r)$, if any, will give rise to exactly i new critical configurations given that at least one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a

critical configuration. Thus, we can express the probability $P[C(r) = i]$ that $C(r)$ will have value i, for some $r \in D$, as

$$P[C(r) = i] = \sum_{(G(r),Y) \in A} P((G(r), Y))[\sum_{S \in B(r)} P(S) \cdot N(s, Y, i)].$$

We built a computer program to compute $N(S, Y, i)$[1], and then we used this program to obtain the value of $P[C(r) = i]$, for each i such that $1 \leq i \leq 36$. Using these values, we have found $P[C(r) = 1] = 0.0791921$, $P[C(r) = 2] = 0.0316537$, $P[C(r) = 3] = 0.0187593$, $P[C(r) = 4] = 0.0112136$, $P[C(r) = 5] = 0.00472093$, $P[C(r) = 6] = 0.00173523$, $P[C(r) = 7] = 0.00070917$, $P[C(r) = 8] = 0.000186012$, $P[C(r) = 9] = 2.32515 \cdot 10^{-5}$, and $P[C(r) = k]$ is approximately zero for $10 \leq k \leq 36$, $k \in \mathbb{Z}$. So, $E[C(r)]$ of $C(r)$ is 0.28529. From the fact that $E[C(r)] = \sum_{i=1}^{36} i \cdot P[C(r) = i] = 0.28529$, we can derive an upper bound for the expected value $E[t] = \sum_{r \in D} E[C(r)]$ of t in terms of m, as formally stated by the following theorem:

Theorem 6.1. *Let (D, X) be a 3D binary digital image. If there are m instances of critical configurations in (D, X), then the expected value $E[t]$ of the number t of new critical configurations created by the repairing algorithm in Section 5 on input (D, X) is less than $m/2$.*

Proof. Let R be the set of points $r \in D$ such that one of $\mathcal{N}_x(r)$, $\mathcal{N}_y(r)$, $\mathcal{N}_z(r)$, and $\mathcal{N}(r)$ is an instance of a critical configuration in (D, X). Clearly, the number of points in R is less than or equal to m. For any $r \in D$, we have that $E[C(r)] = 0.28529$. So, since $C(r) = 0$ for $r \in D \setminus R$, if there are m critical configurations in (D, X), then the expected number of critical configurations resulting from the elimination of m existing critical configurations in (D, X) is $E[\sum_{r \in D} C(r)] = E[\sum_{r \in R} C(r) + \sum_{r \in D \setminus R} C(r)] = E[\sum_{r \in R} C(r)] = \sum_{r \in R} E[C(r)] = 0.28529 \cdot |R| \leq 0.28529 \cdot m$, where $|R|$ is the cardinality of R. Since the algorithm iterates until all critical configurations are eliminated from the binary image $(D, X \cup P)$, where P is the set of background points converted into foreground points at any given time, we have that the expected number $E[t] = \sum_{r \in D} E[C(r)]$ of new critical configurations created during the entire execution of the algorithm is bound above by $\sum_{k=1}^{\infty} (0.28529)^k \cdot m < \sum_{k=1}^{\infty} (1/3)^k \cdot m = m/2$. □

Recall that the size of the set P obtained by our repairing algorithm is bounded above by $m + t$. So, the upper bound for the expected value of t in Theorem 6.1 provides a theoretical argument for the good performance of our repairing algorithm in the experiment described in Section 7, while the experiment itself provides a practical evidence.

7 Experimental Results

As we mentioned in the introduction, if the digitization process that gives rise to a given 3D binary digital image is topology-preserving then the image must be well-composed. So, if an image is not well-composed

[1] http://www.seas.upenn.edu/∼marcelos/probab.cpp

then the digitization process that produced the image is not topology-preserving. The purpose of any repairing algorithm is to "restore" the well-composedness property of the input image (D, X) whenever (D, X) is not well-composed. However, without additional information about the topology that (D, X) would have if it had been generated by a topology-preserving process, the repairing algorithm generates a well-composed image that is at best a good guess for the "correct" well-composed image. From the point of view of image-based applications, a good guess is more likely to be a well-composed image that does not differ too much from (D, X). So, in this context, if (D, X') is the well-composed image, then the size of the set $(X - (X \cap X')) \cup (X' - (X \cap X'))$ is the main indication of the usefulness and effectiveness of a repairing algorithm. Note that the size of $(X - (X \cap X')) \cup (X' - (X \cap X'))$ is precisely the number of identical points in both images that are not assigned the same color. Since $X \subseteq X'$ in the image generated by our repairing algorithm, the set $(X - (X \cap X')) \cup (X' - (X \cap X'))$ is exactly the set P such that $X' = X \cup P$.

We have shown an upper bound for the expected value of the number t of new critical configurations generated by our repairing algorithm on an image with m critical configurations (see Section 6). Since the size of the set P generated by our repairing algorithm is bounded above by $m + t$, this bound provides in theory an idea of the effectiveness of our algorithm. Here, we report on an experiment that applies our algorithm to several medical digital images encountered in practice. Our goal is to also show empirical evidences of the effectiveness of our repairing algorithm. We used six magnetic resonance (MR) images in our experiment. All of them are binary images with $129^3 = 2146689$ points obtained from re-sampling and thresholding segmented grey-scale images. Three of them were obtained from the segmentation of a normal brain image into white matter, grey matter, and cerebrospinal fluid (CSF). The normal brain image was produced by an on-line 3D MR image simulator [13], which can be found in `www.bic.mni.mcgill.ca/brainweb`. Two other images are real lung images corresponding to the inspiration and expiration stages of lung motion [2]. The last image is a male thorax image from the dataset of the Visible Human Project in `www.nlm.nih.gov/research/visible`. This image was segmented using a fuzzy segmentation method [14].

We ran our repairing algorithm on each image 10 times. In each execution of the algorithm, we collected the size of the set P obtained by the algorithm and the number of critical configurations created by our algorithm in order to generate $(D, X' = X \cup P)$. Finally, we computed the average size of P and the average number of new critical configurations for each image. Table 1 shows the results of our experiment. Note that, for each image, the average size of P is at most $0,32\%$ of the size of the image. Furthermore, in all cases, the average size of P is smaller than the total number of critical configurations, which implies that the optimal size of P is also smaller than the number of critical configurations, and therefore the average size of P and its optimal size are close. Note also that the average number of critical configurations is no larger than $0.2 \cdot m$, where m is the number of critical configurations in the input image. The results in Table 1 shows that our algorithm generated well-composed images that are very similar to the input ones. Hence, this algorithm can be very useful as a preprocessing step

[2] We are very grateful to Tessa Sundaram for providing us with the segmented lung images used in this experiment.

of algorithms that benefit from dealing with well-composed images, such as isosurface extraction and thinning algorithms.

Table 1: Results from the application of the repairing algorithm in Section 5 to six distinct MR images with 129^3 points each. The first column identifies the images; the second and third columns contain the number of instances of the critical configuration (C1) and critical configuration (C2) in the input image, respectively; the fourth column shows the average size of the set P; and the fifth column shows the average number of new critical configurations generated by our algorithm in order to compute P. The average size of P and the average number of critical configurations were computed by executing our repairing algorithm on each image 10 times.

Image	**# C. Conf. 1)**	**# C. Conf. 2)**	**Avg. |P|**	**Avg. # New C. C.**
Brain (White Matter)	2538	418	1833.5	249.9
Brain (Grey Matter)	9688	1316	6834.2	2115.9
Brain (CSF)	8574	676	5303.3	787.1
Lung (Inspiration)	1908	128	1213.7	288.4
Lung (Expiration)	3284	237	2068.7	483.2
Thorax	1962	84	1123.7	143.8

8 A New Approach to Extract Simplicial Surfaces

3D digital images are used in several important applications, such as medical imaging, fluid dynamics simulation, and videoconferencing. In such applications, it is often needed to visualize the image. A very popular, two-step scheme to visualize 3D digital images consists of extracting simplicial surfaces from the image, and then using fast algorithms for rendering triangles to display the surface. The marching cubes (MC) algorithm has been the most widely used algorithm for extracting simplicial surfaces from 3D digital images [5]. This algorithm has two well-known problems: (1) the collection of triangles and their vertices and edges produced by the MC algorithm may not be a simplicial surface, and even if it is (2) the simplicial surface typically contains an excessive number of triangles to accurately represent its geometry.

Lachaud [15] proposed an elegant solution to problem (1), which is based on the digital topology of the connected components of the image. His work establishes a relationship between the topology of the isosurface being approximated and the topology of the digital boundary of the sets of image points separated by this surface. This relationship is used is to guarantee that the output of the MC algorithm is always a simplicial surface. Other solutions to problem (1) have been proposed (e.g., see [16, 17, 18, 19, 20, 21, 22]), but none of them explore the relationship given by Lachaud in [15] between the isosurface being approximated and the topology of the digital boundary of the sets of image points separated by

this surface.

We introduce a new approach for extracting simplicial surfaces. Our approach provides a new solution for problem (1) and makes some algorithms that address problem (2) simpler. Like Lachaud's solution [15], ours also relies on the digital topology of the connected components of the image. The idea behind our solution is to make the input image well-composed, and then use the topology of the face boundaries of the image to determine the topology of the simplicial surface. The main advantage of our solution lies in the fact that the MC algorithm always generates a simplicial surface with no need for adopting any mechanism (e.g., a look-up table) to guarantee that the output of the algorithm is a simplicial surface. This is also the case with some octree-based algorithms that deal with problem (2) [6, 7]. Before we give details of our approach, we review the MC algorithm and two other algorithms from [6, 7].

8.1 The MC Algorithm and Octree-based Algorithms

The MC algorithm is the most widely used algorithm for extracting simplicial surfaces from 3D imaging data. This algorithm takes as input a 3D digital image $\mathcal{I} : D \rightarrow V$ and a scalar $\alpha \in \mathbb{R}$, and then generates a simplicial surface that separates the points $p \in D$ with $\mathcal{I}(p) \leq \alpha$ from the points $q \in D$ with $\mathcal{I}(q) > \alpha$. This simplicial surface can be viewed as an approximation for an isosurface $f^{-1}(\alpha)$ in $\mathbb{R}^3$ defined by a continuous scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ such that $\mathcal{I}(p) = f(p)$, for every $p \in D$. Note that, in this context, the digital image $\mathcal{I}$ is assumed to be a sampling of a continuous scalar field at the points in D. However, the MC algorithm does not have any further knowledge about f, except for the values of f at the points of D.

The domain D of $\mathcal{I}$ is assumed to be a finite rectilinear grid of points of $\mathbb{Z}^3$, i.e., $D = [d_{11}, d_{12}] \times [d_{21}, d_{22}] \times [d_{31}, d_{32}]$, where $d_{i1}, d_{i2} \in \mathbb{Z}$ and $d_{i1} < d_{i2}$, for $1 \leq i \leq 3$. We also assume that the values of $\mathcal{I}$ at points of the "boundary" of D are either all less than or equal to α or all greater than α, i.e., either $\mathcal{I}(p) \leq \alpha$ for all points $p = (p_1, p_2, p_3) \in D$ such that $p_1 \in \{d_{11}, d_{12}\}$ or $p_2 \in \{d_{21}, d_{22}\}$ or $p_3 \in \{d_{31}, d_{32}\}$, or $\mathcal{I}(p) > \alpha$ for all points $p = (p_1, p_2, p_3) \in D$ such that $p_1 \in \{d_{11}, d_{12}\}$ or $p_2 \in \{d_{21}, d_{22}\}$ or $p_3 \in \{d_{31}, d_{32}\}$.

The MC algorithm perceives D as a set of integer points in $\mathbb{R}^3$, and then constructs a cube subdivision of the convex hull of D, $conv(D)$, as follows: For each $i \in \{0, \ldots, d_{12} - d_{11} - 1\}$, $j \in \{0, \ldots, d_{22} - d_{21} - 1\}$, and $k \in \{0, \ldots, d_{32} - d_{31} - 1\}$, the algorithm builds a cube whose vertices are the eight points of D with coordinates $(d_{11} + i, d_{21} + j, d_{31} + k)$, $(d_{11} + i + 1, d_{21} + j, d_{31} + k)$, $(d_{11} + i, d_{21} + j + 1, d_{31} + k)$, $(d_{11} + i + 1, d_{21} + j + 1, d_{31} + k)$, $(d_{11} + i, d_{21} + j, d_{31} + k + 1)$, $(d_{11} + i + 1, d_{21} + j, d_{31} + k + 1)$, $(d_{11} + i, d_{21} + j + 1, d_{31} + k + 1)$, and $(d_{11} + i + 1, d_{21} + j + 1, d_{31} + k + 1)$. The next step is the computation of the vertices of the simplicial surface. This computation starts by a search for the subdivision cubes that intersect $f^{-1}(\alpha)$. A cube is assumed to intersect $f^{-1}(\alpha)$ if and only if the values of f at its vertices are not all greater than α or not all smaller than or equal to α. If the values of f at the vertices of

the cube are all smaller (resp. all greater) than α then the cube is assumed to be inside (resp. outside) $f^{-1}(\alpha)$.

For each cube that intersects $f^{-1}(\alpha)$, the algorithm finds its intersecting edges. An edge $[q, s]$ of the cube is said to be an *intersecting edge* if and only if either $f(q) \leq \alpha$ and $f(s) > \alpha$ or $f(q) > \alpha$ and $f(s) \leq \alpha$, or equivalently, if and only if either $\mathcal{I}(q) \leq \alpha$ and $\mathcal{I}(s) > \alpha$ or $\mathcal{I}(q) > \alpha$ and $\mathcal{I}(s) \leq \alpha$. Since f is assumed to be a continuous function, every intersecting edge must contain a point p such that $f(p) = \alpha$. A vertex of the simplicial surface is then computed in the interior of each intersecting edge. The position of this vertex is an approximation for the point p in the interior of the edge such that $f(p) = \alpha$. Note that a cube intersects $f^{-1}(\alpha)$ if and only if one of it contains an intersecting edge. Since either $\mathcal{I}(p) \leq \alpha$ for all points $p = (p_1, p_2, p_3) \in D$ such that $p_1 \in \{d_{11}, d_{12}\}$ or $p_2 \in \{d_{21}, d_{22}\}$ or $p_3 \in \{d_{31}, d_{32}\}$, or $\mathcal{I}(p) > \alpha$ for all points $p = (p_1, p_2, p_3) \in D$ such that $p_1 \in \{d_{11}, d_{12}\}$ or $p_2 \in \{d_{21}, d_{22}\}$ or $p_3 \in \{d_{31}, d_{32}\}$, all intersecting edges of the cube subdivision of the convex hull $conv(D)$ of D are in the interior of $conv(D)$, or equivalently, all intersecting edges are shared by exactly four cubes of the cube subdivision of $conv(D)$.

Finally, the MC algorithm connects the vertices of the simplicial surface along the faces of the cubes to produce one or more 3D polygons. Each 3D polygon is later triangulated to give rise to the faces (triangles) of the simplicial surface inside the cubes, as shown in Figure 10. The way the MC algorithm connects the vertices of the simplicial surface along the faces of a cube to produce 3D polygons is based on the values of f at the eight vertices of the cube. Since there are eight vertices in each cube and each vertex can be classified as being inside or outside $f^{-1}(\alpha)$ (i.e., either $f(p) \leq \alpha$ or $f(p) > \alpha$), for every point p in D), there are $2^8 = 256$ distinct configurations for each cube of the subdivision. However, due to complementary and rotational symmetry, we can show that these 256 configurations are equivalent to the 14 canonical configurations illustrated in Figure 10 [22]. Complementary symmetric cases are those where the vertices classified as inside $f^{-1}(\alpha)$ in the canonical cases are vertices classified as being outside $f^{-1}(\alpha)$, and vice-versa.

Note that configurations 1, 2, 5, 8, 9, and 11 admit only one way of connecting the vertices of the simplicial surface to produce a 3D polygon along the faces of the cube, as every face of the cube in those configurations contains either none or exactly two intersecting edges. Unlike, configurations 3, 4, 6, 7, 10, 12 and 13 admit more than one way of connecting the vertices of the simplicial surface to produce a 3D polygon along the faces of the cube, as a face of the cube in those configurations may contain more than one intersecting edge. Furthermore, configurations 3, 4, 6, 7, 10, 12 and 13 produce more than one 3D polygon. So, the MC algorithm has more than one choice for connecting vertices of the simplicial surface to form 3D polygons in configurations 3, 4, 6, 7, 10, 12 and 13. These configurations are often referred to as *ambiguous* configurations.

Each distinct choice of connection in the ambiguous configurations affects the resulting topology of the simplicial surface. Furthermore, if each choice is not made in a consistent way, the resulting topology may be "invalid", i.e., the resulting collection of triangles and their edges and vertices does not define a

simplicial surface, according to the definition of simplicial surface in Section 3. For instance, Figure 11 illustrates this situation using two instances of configuration 10, each of which produces two 3D polygons by connecting the vertices in the interior of the intersecting edges in a distinct way. As a result, the triangle edges in the face shared by the two cubes in Figure 11 are incident to only one triangle, which violates our definition of simplicial surface. Note that this is the case regardless of the way we triangulate the 3D polygons.

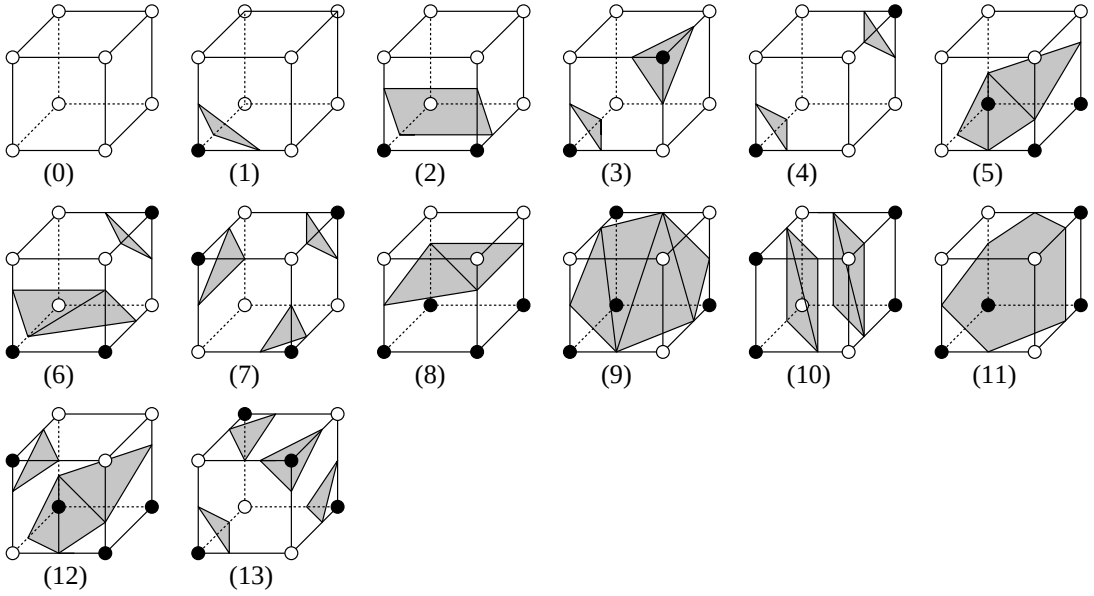

Figure 10: The canonical configurations of the MC algorithm and examples of how the 3D polygons can be simplicial. Vertices of the cube represented by circles with the same color are in the same side of $f^{-1}(\alpha)$.

Since the MC algorithm has been created, there have been developed several solutions for coping with the problem of guaranteeing that the output of the MC algorithm is always a simplicial surface [16, 17, 18, 19, 20, 15, 21, 22]. A simple solution is to use a look-up table with an entry for each of the 256 distinct configurations of a subdivision cube. As we mentioned before, only the configurations corresponding to configurations 3, 4, 6, 7, 10, 12 and 13 in Figure 11 admit more than one way of connecting the vertices of the simplicial surface. For the entries corresponding to these configurations, we store exactly one of the possible ways of connecting the vertices of the simplicial surface. The choice for a particular way may be arbitrary [19] or be based on further information about the input image, such as the topology of the digital boundary of the sets of image points separated by the underlying surface of the resulting simplicial surface [15]. In either case, the MC algorithm is guaranteed to generate a simplicial surface if, for any given subdivision cube that intersects the isosurface, the algorithm connects the vertices computed in the intersecting edges of the cube according to the connection specified in the look-up table entry corresponding to the cube configuration.

A more sophisticated solution is to define a tri-affine interpolant over the vertices computed in the

intersecting edges of the cube, and then use this interpolant to consistently form the 3D polygons in the ambiguous configurations [16, 17, 18, 20, 21, 22]. The idea behind this solution is to assume that the isosurface defined by the (unknown) scalar field f has the same topology as the zero set of the tri-affine interpolant inside the cube and inside its faces. Note that, because the MC algorithm does not have any further knowledge about f other than its values at the points of D, none of the proposed solutions can guarantee that the resulting simplicial surface has the same consistent topology as the one of $f^{-1}(\alpha)$, i.e., the simplicial surface generated by the MC algorithm is at best a good guess for $f^{-1}(\alpha)$. Furthermore, $f^{-1}(\alpha)$ may not even be a surface for all continuous scalar fields $f : \mathbb{R}^3 \to \mathbb{R}$ such that $\mathcal{I}(p) = f(p)$, for every $p \in D$. However, if f is known, then it is possible to decide how to connect points in the ambiguous configurations such that the output is always a simplicial surface *and* has exactly the same topology as $f^{-1}(\alpha)$ [23, 24]. Unfortunately, for most image-based applications, the continuous scalar field f is unknown.

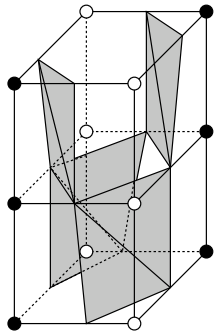

Figure 11: The canonical configurations of the MC algorithm and examples of how the 3D polygons can be simplicial. Vertices of the cube represented by circles with the same color are in the same side of $f^{-1}(\alpha)$.

Unlike 2D polygons, a 3D polygon need not be planar nor triangulable. By triangulable, we mean a 3D polygon that admits a non-self-intersecting triangulation that defines a simplicial complex whose underlying space is homeomorphic to the open unit disk. It turns out that the problem of deciding whether a 3D polygon is triangulable is **NP**-complete [25]. However, a lemma given and proved in [25] states that, if the orthogonal projection of a 3D polygon onto some plane is a 2D polygon, then the 3D polygon is triangulable. Fortunately, this is obviously the case for the 3D polygons of any of the 14 canonical configurations in Figure 10 of the MC algorithm. Furthermore, any triangulation of the 2D polygon obtained from the aforementioned projection of any of the 3D polygons in Figure 10 induces a non-self-intersecting triangulation of the 3D polygon, and this triangulation always defines a simplicial complex whose underlying space is homeomorphic to the open unit disk. So, the MC algorithm does not have to verify if a triangulation is "valid".

Because the MC algorithm uses a *uniform* cube subdivision of $conv(D)$, i.e., a cube subdivision in which all cubes have the same size, the simplicial surface generated by the MC algorithm has, in general, an excessive number of triangles to represent the surface geometry. This issue has also been addressed by

several researchers, and the most common solution is to use an *adaptive* cube subdivision [26] managed by a balanced octree. By adaptive subdivision, we mean a subdivision in which the cubes may have different size. An adaptive subdivision is typically obtained by either refining a coarse uniform subdivision or merging cubes of a dense uniform subdivision.

Shekhar, Fayyad, Yagel, and Cornhill [6] and Ju, Losasso, Schaefer, and Warren [7] introduced similar algorithms that employ the latter strategy. Both algorithms determine the appropriate size for a cube in any region of the convex hull $conv(D)$ of D by estimating the local variation of surface curvature in the region. The higher the curvature variation the smaller the cube size. As a result, the simplicial surface produced by these algorithms contains less triangles than the one generated by the MC algorithm, but it still can faithfully represent the geometry of the surface generated by the MC algorithm. An interesting feature of both algorithms is that they determine the topology of the simplicial surface using the same uniform subdivision of $conv(D)$ as the MC algorithm does, and the same cube configurations illustrated by Figure 10.

8.2 Extraction of Simplicial Surfaces from Well-Composed Images

Our approach to extract simplicial surfaces from 3D binary digital images consists of two steps: well-composedness and surface extraction, in this order. The first step, *well-composedness*, takes as input a scalar $\alpha \in \mathbb{R}$ and a 3D digital image $\mathcal{I} : D \to V$, where D is a finite rectilinear grid, i.e., $D = [d_{11}, d_{12}] \times [d_{21}, d_{22}] \times [d_{31}, d_{32}]$, where $d_{i1}, d_{i2} \in \mathbb{Z}$ and $d_{i1} < d_{i2}$, for $1 \leq i \leq 3$, and the values of $\mathcal{I}$ at points of the "boundary" of D are either all less than or equal to α or all greater than α. A 3D binary digital image (D, X) is generated by thresholding $\mathcal{I}$: $X = \{p \in D \mid \mathcal{I}(p) > \alpha\}$ and $X^c = D \setminus X = \{p \in D \mid \mathcal{I}(p) \leq \alpha\}$. Finally, if (D, X) is not well-composed, then our repairing algorithm in Section 5 is used to produce a well-composed image (D, X'), where $X \subset X'$. Otherwise, we let $X' = X$.

Since (D, X') is well-composed, Theorem 4.1 tells us that $bdCA(X')$ is a surface. Since $bdCA(X')$ is also the boundary between $CA(X'_0)$ and $CA(X'_1)$, we have that $bdCA(X')$ separates the set X'_0 of background points from the set X'_1 of foreground points of (D, X'). Also, since the values of $\mathcal{I}$ at points of the "boundary" of D are either all less than or equal to α or all greater than α, these points are either all background points or all foreground points of (D, X). From an observation in the end of Section 5, we know that the points of the boundary of D are either all background points or foreground points of (D, X'). This observation implies that the surface $bdCA(X')$ in entirely contained in the *interior* of the convex hull $conv(D)$ of D.

The second step, *surface extraction*, generates a simplicial surface that has the same topology as $bdCA(X')$ and also separates the points of X'_0 from the ones of X'_1. This simplicial surface can be computed using the MC algorithm or any of the octree-based algorithms by Shekhar, Fayyad, Yagel, and Cornhill [6] and Ju, Losasso, Schaefer, and Warren [7]. Furthermore, we shall see that there is no need to include any special procedure for handling topological inconsistencies. Note that the input for the

second step is (D, X') and any scalar value β such that $0 \leq \beta < 1$, as we want a simplicial surface that separates the points of X'_0 from the ones of X'_1.

Since we described our repairing algorithm in Section 5 and the MC algorithm in Section 8.1, we now focus on proving our claim that, given a well-composed, binary digital image (D, X') and any scalar value β such that $0 \leq \beta < 1$, the MC algorithm produces a simplicial surface that has the same topology as $bdCA(X')$ and also separates the set X'_0 of background points from the set X'_1 of foreground points of (D, X'). We also show that this is also the case with any of the algorithms in [6, 7]. Before we present our proof, we remark four important facts used by our proof. Let $\mathcal{I}' : D \rightarrow \{0, 1\}$ be the map corresponding to (D, X'). Then,

1. From the description of the MC algorithm in Section 8.1, we know that an edge $[q, s]$ of a subdivision cube is intersecting if and only if either $\mathcal{I}'(q) \leq \beta$ and $\mathcal{I}'(s) > \beta$ or $\mathcal{I}'(q) > \beta$ and $\mathcal{I}'(s) \leq \beta$. Because $0 \leq \beta < 1$ and $\mathcal{I}'(p)$ is either 0 or 1, for every $p \in D$, we have that an edge $[q, s]$ of a subdivision cube is intersecting if and only if either $q \in X'_0$ and $s \in X'_1$ or $q \in X'_1$ and $s \in X'_0$.

2. From the previous fact, we can conclude that only six out of the fourteen canonical configurations of the MC algorithm can occur if the input image is a well-composed one. These six configurations are configurations 0, 1, 2, 5, 8, 9, and 11 in Figure 10. The reason is that $\mathcal{I}'(p) \leq \beta$ (resp. $\mathcal{I}'(p) > \beta$) if and only if $p \in X'_0$ (resp. $p \in X'_1$), for every $p \in D$. So, if one of configurations 3, 4, 6, 7, 10, 12, and 13 in Figure 10 occurs, then the input image cannot be well-composed. Recall that configurations 3, 4, 6, 7, 10, 12, and 13 are exactly the ambiguous configurations of the MC algorithm.

3. A subdivision cube intersects the surface $bdCA(X')$ if and only if its vertices are not all points of X_0 nor all points of X_1. So, a subdivision cube intersects the surface $bdCA(X')$ if and only if it contains an intersecting edge. This implies that both $bdCA(X')$ and the collection of triangles produced by the MC algorithm are enclosed by the exact same set of cubes of the subdivision of $conv(D)$.

4. Since the points in the boundary of D are either all background points or foreground points of (D, X'), the values of $\mathcal{I}'$ at these points are either all 0 or all 1, respectively. So, each intersecting edge of the cube subdivision built by the MC algorithm is shared by exactly four cubes of the subdivision.

Since the ambiguous configurations of the MC algorithm never occur if the input image is well-composed, our approach to extract simplicial surfaces from well-composed images can use a simplified version of the MC algorithm, which does not include the handling of ambiguous configurations. In the following, we state and prove our claim that the MC algorithm always produces a simplicial surface topologically equivalent to $bdCA(X')$ if the input is a well-composed binary image (D, X') and any scalar value β such that $0 \leq \beta < 1$:

Theorem 8.1. *Let (D, X') be a 3D well-composed, binary digital image. Then, the output of the MC algorithm on input (D, X') and $\beta \in \mathbb{R}$, with $0 \leq \beta < 1$, is a simplicial surface whose the underlying surface separates the set X'_0 of background points from the set X'_1 of foreground points of (D, X') and is homeomorphic to $bdCA(X')$.*

Proof. Let $\mathcal{K}$ be the collection of triangles generated by the MC algorithm along with their edges and vertices. We claim that $\mathcal{K}$ is a simplicial surface in $\mathbb{R}^3$. First, we show that $\mathcal{K}$ is a simplicial complex in $\mathbb{R}^3$. By definition of $\mathcal{K}$, any vertex and edge of $\mathcal{K}$ is a vertex and edge of a triangle in $\mathcal{K}$. So, for every simplex of $\mathcal{K}$, its faces are simplices of $\mathcal{K}$. Now, we must show that the intersection of any two simplices in $\mathcal{K}$ is either empty or a simplex of $\mathcal{K}$. From the description of the MC algorithm in Section 8.1, we know that every triangle produced by the MC algorithm is a triangle of a triangulation of a 3D polygon defined on the faces of a subdivision cube. This triangulation is a simplicial complex whose underlying space is homeomorphic to the open unit disk. So, the intersection of any two edges (resp. triangles) of $\mathcal{K}$ inside the *same* cube is either empty or a common vertex (resp. common vertex or edge) of $\mathcal{K}$. Furthermore, since every triangle produced by the MC algorithm is entirely contained in a single cube, the intersection of any two triangles of $\mathcal{K}$ that are produced by the MC algorithm inside distinct subdivision cubes is either empty or a common vertex or edge of the 3D polygons defined on the faces of the two cubes. Since every edge of $\mathcal{K}$ is an edge of a triangle of $\mathcal{K}$, the intersection of any two edges inside distinct subdivision cubes is either empty or a common vertex of the 3D polygons defined on the faces of the two cubes. So, the intersection of any two edges (resp. triangles) in $\mathcal{K}$ is either empty or a vertex (resp. vertex or edge) of $\mathcal{K}$, and therefore $\mathcal{K}$ is a simplicial complex in $\mathbb{R}^3$.

We now show that $\mathcal{K}$ is also a simplicial surface. From Lemma 3.2, we have that $\mathcal{K}$ is a simplicial surface if $\mathcal{K}$ is a 2-complex and each edge of $\mathcal{K}$ is an edge of exactly two triangles of $\mathcal{K}$, and the underlying space of the link of each vertex of $\mathcal{K}$ is homeomorphic to the 1-sphere. Each edge of $\mathcal{K}$ is an edge of the triangulation of a 3D polygon defined over the faces of a subdivision cube. Since the underlying space of such a triangulation is homeomorphic to the open unit disk, if the edge does not belong to the 3D polygon, it is not a boundary edge of the triangulation and therefore it must be shared by exactly two triangles of the triangulation. If it belongs to the 3D polygon, then it is in the face of a subdivision cube. Since the input image (D, X') is well-composed, the MC algorithm uses only configurations 1, 2, 5, 8, 9, and 11 in Figure 10 to produce triangles. Because the MC algorithm produces exactly one 3D polygon in each of these configurations, each edge of a 3D polygon computed in a cube is also an edge of the 3D polygon computed in the adjacent cube that contains the face in which the edge lies. So, the edge must be shared by one triangle in each cube. Hence, each edge of $\mathcal{K}$ is an edge of exactly two triangles of $\mathcal{K}$. Note that this need not be true if any of the ambiguous configurations can occur, as shown in Figure 11. Now, note that every vertex v of $\mathcal{K}$ is a point in the interior of an intersecting edge of a cube of the subdivision of $conv(D)$ built by the MC algorithm. Since each intersecting edge is shared by exactly four subdivision cubes and each cube containing an intersecting edge corresponds to one of the cubes in configurations 1, 2, 5, 8, 9, and 11, the vertex v is the common vertex of the 3D

polygons of four subdivision cubes that share an edge. This implies that v is the common vertex of the triangulations of these four 3D polygons. Since every edge of $\mathcal{K}$ is shared by exactly two triangles, we can conclude that every edge of $\mathcal{K}$ containing v is shared by two triangles inside the same subdivision cube or by two triangles in distinct, but face-adjacent cubes. Since the underlying space of any of the triangulations of the four 3D polygons sharing v is homeomorphic to the open unit disk, the collection $\mathcal{K}$ of all triangles incident to v along with their vertices and edges must be a connected 2-complex in $\mathbb{R}^3$ whose the underlying space is homeomorphic to the open unit disk. Since the link $lk(v, \mathcal{K})$ of v in $\mathcal{K}$ is the boundary of this 2-complex, we have that the underlying space of $lk(v, \mathcal{K})$ is homeomorphic to the 1-sphere, and our claim follows.

We know that a cube of the subdivision intersects $bdCA(X')$ if and only if the cube contains an intersecting edge. But, a subdivision cube contains an intersecting edge if and only if it intersects the simplicial surface $\mathcal{K}$. So, both $bdCA(X')$ and $\mathcal{K}$ are enclosed by the same set of cubes of the subdivision. Since the configurations of the subdivision cubes can one of configurations 0, 1, 2, 5, 8, 9, and 11 in Figure 10, the intersection of $bdCA(X')$ with a cube in any of configurations 1, 2, 5, 8, 9, and 11 in Figure 10 is also homeomorphic to the open unit disk, as illustrated by Figure 12. In order to prove that $bdCA(X')$ and $|\mathcal{K}|$ are homeomorphic, we define to simplicially isomorphic simplicial surfaces, $\mathcal{K}_1$ and $\mathcal{K}_2$, in $\mathbb{R}^3$, such that $|\mathcal{K}_1|$ and $|\mathcal{K}_2|$ are identical to $|\mathcal{K}|$ and $bdCA(X')$, respectively. From Lemma 3.1, if $\mathcal{K}_1$ and $\mathcal{K}_2$ are simplicially isomorphic complexes then $|\mathcal{K}_1|$ and $|\mathcal{K}_2|$ are homeomorphic. But, since $|\mathcal{K}_1| = |\mathcal{K}|$ and $|\mathcal{K}_2| = bdCA(X')$, we can conclude that $bdCA(X')$ and $|\mathcal{K}|$ are homeomorphic. So, by providing $\mathcal{K}_1$ and $\mathcal{K}_2$, we prove our claim.

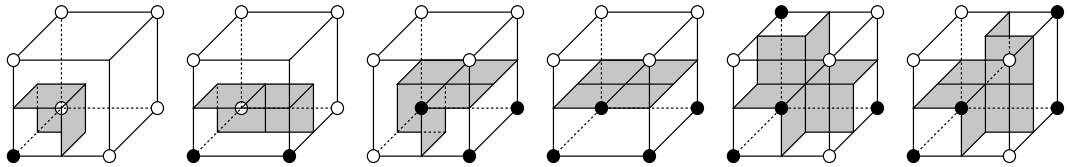

Figure 12: All possible surface patches resulting from the intersection of $bdCA(X')$ and a subdivision cube when (D, X') is well-composed.

We define $\mathcal{K}_1$ as a subdivision of $\mathcal{K}$. The idea is to subdivide the triangulations that gave rise to the simplices of $\mathcal{K}$ inside each cube of the subdivision of $conv(D)$ built by the MC algorithm. Figure 13 illustrates the subdivision of these triangulations corresponding to configurations in 1, 2, 5, 8, 9, and 11 in Figure 10. Note that each edge of the boundary of any triangulation is subdivided into two ones, so that these triangulations glue nicely along the faces of the subdivision cubes. The complex $\mathcal{K}_2$ is defined by imposing a triangulation on the surface patches resulting from the intersection of $bdCA(X')$ and the subdivision cubes. These patches are shown in Figure 12, and the triangulations imposed on the patches are shown in Figure 14. The vertices, edges, and triangles of the triangulations in Figure 14 lie in $bdCA(X')$, so that $bdCA(X') = |\mathcal{K}_2|$.

Finally, note that any triangulation in Figure 13 corresponding to one of the configurations 1, 2, 5, 8, 9, and 11 in Figure 10 is simplicially isomorphic to the triangulation in Figure 14 corresponding to the same configuration in Figure 10. Hence, we can build a simplicial map from $\mathcal{K}_1$ to $\mathcal{K}_2$ by mapping each vertex of $\mathcal{K}_1$ in the interior of an intersecting edge of the cube subdivision to the vertex of $\mathcal{K}_2$ in the interior of the same edge. Likewise, we map the vertex of $\mathcal{K}_1$ in the interior of a face of the cube subdivision to the vertex of $\mathcal{K}_2$ in the interior of the same face. The vertices of $\mathcal{K}_1$ and $\mathcal{K}_2$ in the interior of the subdivision cubes are mapped in such a way that the triangulations inside the cubes are simplicially isomorphic. So, from Lemma 3.1, the surfaces $|\mathcal{K}_1|$ and $|\mathcal{K}_2|$, are indeed homeomorphic, and since $|\mathcal{K}| = |\mathcal{K}_1|$ and $bdCA(X') = |\mathcal{K}_2|$, we have that the underlying surface $|\mathcal{K}|$ of $\mathcal{K}$ and $bdCA(X')$ are homeomorphic.

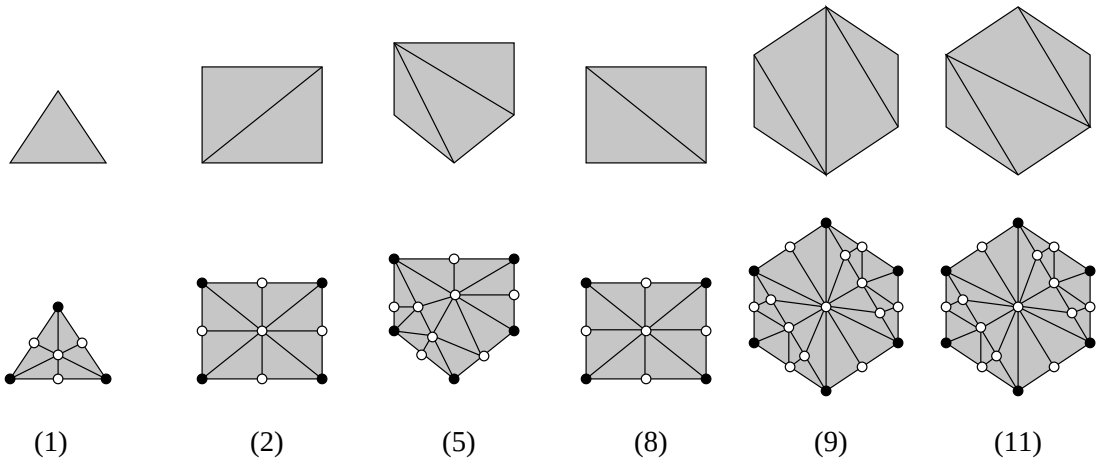

Figure 13: Subdivisions of the triangulations in configurations 1, 2, 5, 8, 9, and 11 in Figure 10. Vertices added to the resulting triangulations are represented by white circles.

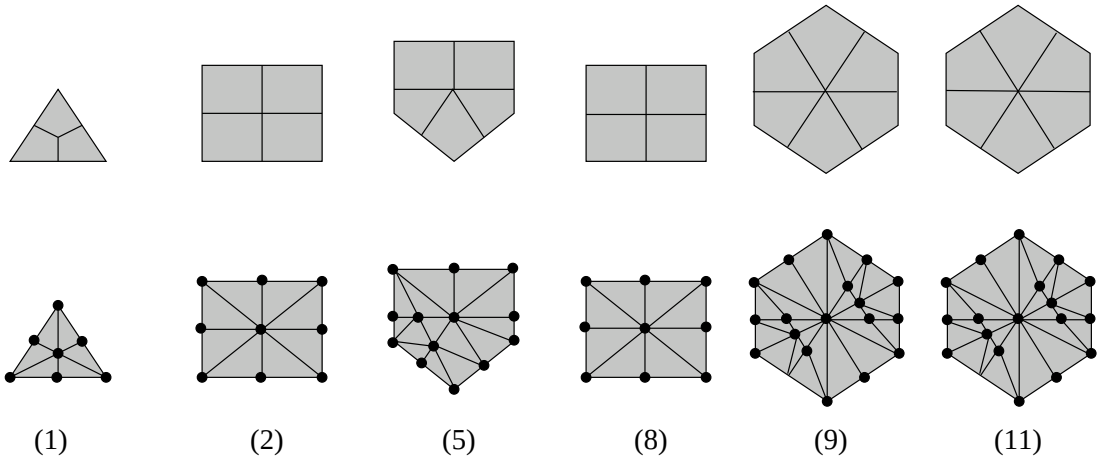

Figure 14: Triangulations imposed on the surface patches resulting from the intersection of $bdCA(X')$ and the subdivision cubes in Figure 12.

We now show that $|\mathcal{K}|$ separates the set X'_0 of background points from the set X'_1 of foreground points of (D, X'). Let R be the subset of $\mathbb{R}^3$ containing all points enclosed by the cubes of the subdivision of

$conv(D)$ that contain $|\mathcal{K}|$. Since $|\mathcal{K}|$ (resp. $bdCA(X')$) is a surface in $\mathbb{R}^3$, each connected component of $\mathcal{K}$ (resp. $bdCA(X')$) partitions a connected component of R into two openly disjoint regions whose common boundary is $\mathcal{K}$ (resp. $bdCA(X')$). Furthermore, each connected component of $\mathbb{R}^3 - int(R)$ is a proper subset of a connected component of $\mathbb{R}^3 - |\mathcal{K}|$ (resp. $bdCA(X') - int(R)$), where $int(R)$ is the interior of R. Since $int(R)$ does not contain any point of D and $bdCA(X')$ separates X'_0 from X'_1, any two points $p \in X'_0$ to a point $q \in X'_1$ cannot belong to the same connected component of $\mathbb{R}^3 - int(R)$. Since each connected component of $\mathbb{R}^3 - int(R)$ is a proper subset of a connected component of $\mathbb{R}^3 - |\mathcal{K}|$, any path from p to q must cross $|\mathcal{K}|$, and therefore $|\mathcal{K}|$ also separates X'_0 from X'_1. □

As we remarked before, the algorithms by Shekhar, Fayyad, Yagel, and Cornhill [6] and Ju, Losasso, Schaefer, and Warren [7] also determined the topology of the resulting simplicial surface using the same cube subidivision of the convex hull $conv(D)$ of D as the MC algorithm. This means that, if the input image (D, X') is well-composed, then the topology of the resulting simplicial surface is also the same as that of $bdCA(X')$, and the task of determining this topology can also be simplified. The benefits from using any of the algorithms in [6, 7] is that the resulting simplicial surface will not contain an excessive number of triangles to faithfully represent the geometry of the underlying surface. However, the underlying surface may no longer separate the background points from the foreground points of (D, X'). This is because the underlying surface may no longer be enclosed by the same set of cubes that encloses $bdCA(X')$ in the uniform subdivision.

8.3 Examples

We extracted simplicial surfaces from the images in our experiments in Section 7. We used both the MC algorithm and the algorithm by Ju, Losasso, Schaefer, and Warren [7]. We ran the MC algorithm on each image before and after making it well-composed. Then, we computed the Euler characteristic of each connected component of the simplicial surfaces, so that we can have an idea of the influence of our repairing algorithm on the topology of the surfaces extracted from the original images and their well-composed counterparts. Although the original images and their well-composed counterparts are almost identical (see Table 1 in Section 7), we noticed that the simplicial surfaces extracted from them may have quite different topologies. This was mostly the case with the brain images, as our repairing algorithm eliminated several small "tunnels" from the input image by "closing" either one or both of their openings. From the topological point of view, several small holes in the simplicial surface extracted from the original image were eliminated or converted into spheres in its well-composed counterpart. Note that, if a hole in the input image is converted into a sphere in its well-composed counterpart, then the simplicial surface extracted from the well-composed image has one more connected component than the one extracted from the original image.

Figures 15, 16, and 17 are screen shots of the simplicial surfaces extracted by the MC algorithm and the algorithm in [7] from the brain image used in our experiment in Section 7. These surfaces

correspond to approximations to the boundary of the white matter region of the brain. Figure 15 shows the simplicial surface generated by the MC algorithm from the original brain image. Figure 16 shows the simplicial surface generated by the MC algorithm from the well-composed counterpart of the original brain image. Figure 17 shows the simplicial surface generated by the algorithm in [7] from the well-composed counterpart of the original brain image. Although the images in Figures 15 and 16 look very similar, the Euler characteristics of their corresponding simplicial surfaces show that they are different from the topological point of view. The simplicial surface in Figure 15 has two connected components: One component has 88748 vertices, 267294 edges, 178196 faces, and 176 holes, and the other has 82 vertices, 240 edges, 160 faces, and no holes. The simplicial surface in Figure 16 three connected components: One component has 87878 vertices, 263946 edges, 175964 faces, and 53 holes; another one has 82 vertices, 240 edges, 160 faces, and no holes; and the third one has 6 vertices, 12 edges, 8 faces, and no holes.

The simplicial surface in Figure 17 has also three connected components: One component has 54850 vertices, 164862 edges, 109908 faces, and 53 holes; another one has 8 vertices, 18 edges, 12 faces, and no holes; and the third one has 41 vertices, 117 edges, 78 faces, and no holes. Note that the topology of this surface is exactly the same as the one of the simplicial surface in Figure 16. This is due to the fact that both surfaces were generated from the same well-composed image. However, since the triangulated surface in Figure 17 was created from an adaptive cube subdivision, it has less faces, edges, and vertices than the one in Figure 16, which was created from a uniform cube subdivision. Note that the fine geometric details of the simplicial surface in Figure 16 were preserved by the simplicial surface in Figure 17. This is because fine geometric details correspond to surface regions with large variation of curvature, and the algorithm in [7] uses small cubes to enclose these surface regions, i.e., cubes are only merged to create larger ones if the surface region enclosed by the cubes to be merged can be faithfully approximated by the triangles computed in the resulting larger cube.

9 Conclusion

We presented a new repairing algorithm for making 3D binary digital images well-composed. Our algorithm is randomized and "repairs" the input image by changing the color of background points so that they become foreground points in the resulting well-composed images. Our algorithm is relatively simple, as it is the case with most randomized algorithms [27]. Each background point made into a foreground point eliminates at least one critical configuration of the image, but it may also give rise to one or more critical configurations. However, our algorithm cannot create new critical configurations indefinitely, as we showed that it always terminates and produces a well-composed image. We also showed that the expected number of new critical configurations is bounded above by a fraction of the number of critical configurations in the input image. Furthermore, experiments with medical imaging data showed that the number of new critical configurations generated by our algorithm is far below the given theoretical bound.

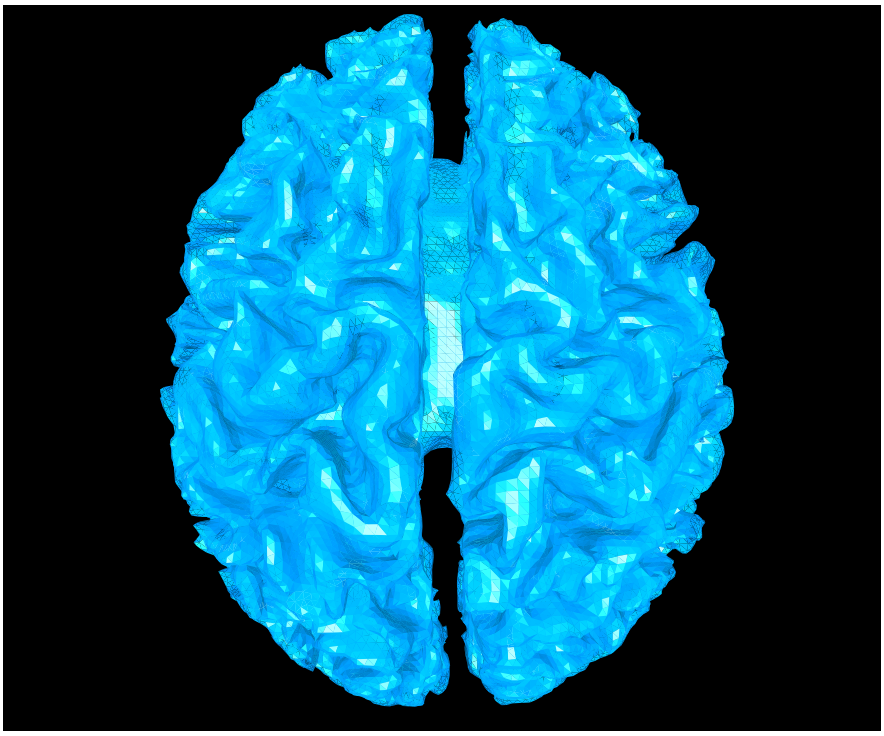

Figure 15: Simplicial surface extracted by the MC algorithm from the brain image used in the experiment in Section 7.

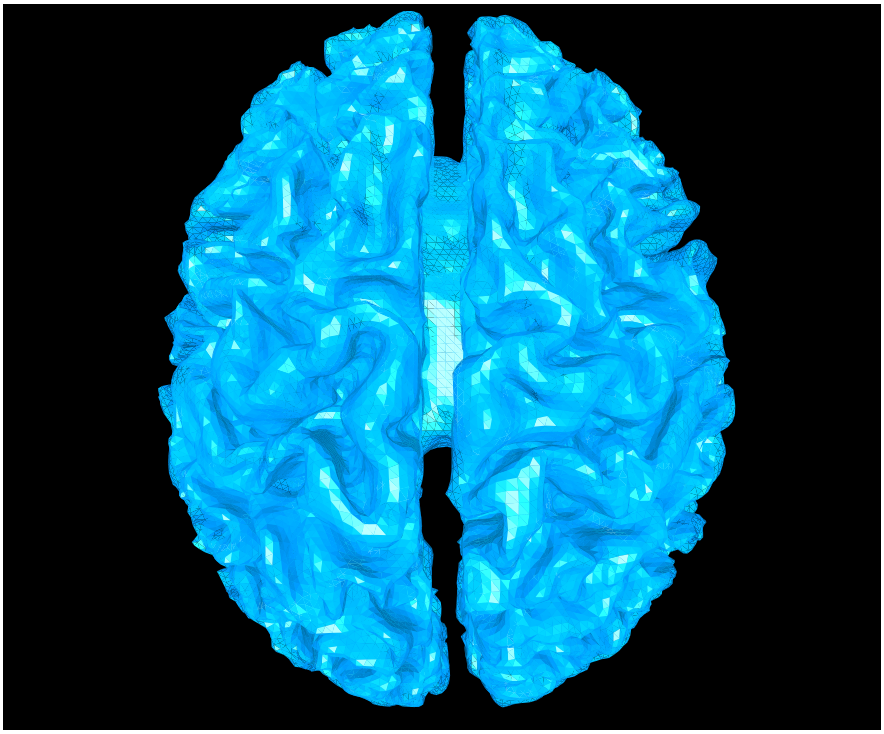

Figure 16: Simplicial surface extracted by the MC algorithm from the well-composed counterpart of the brain image used in the experiment in Section 7.

We described a new approach to extract simplicial surfaces from 3D binary digital images. The idea behind our approach is to make the input image well-composed. By doing so, a topologically consistent simplicial surface can always be generated by the MC algorithm with no need for handling ambiguous configurations. The main advantage of our approach lies in the simplification of the task of generating a topologically consistent simplicial surface using the MC algorithm or any of the octree-based algorithms in [6] and [7]. Our approach relies on the premise that the differences between the input image and its well-composed counterpart are negligible. This premise held for the images used in our experiment in Section 5 and their well-composed counterparts generated by our repairing algorithm. This indicates that our approach can be successfully used in practice.

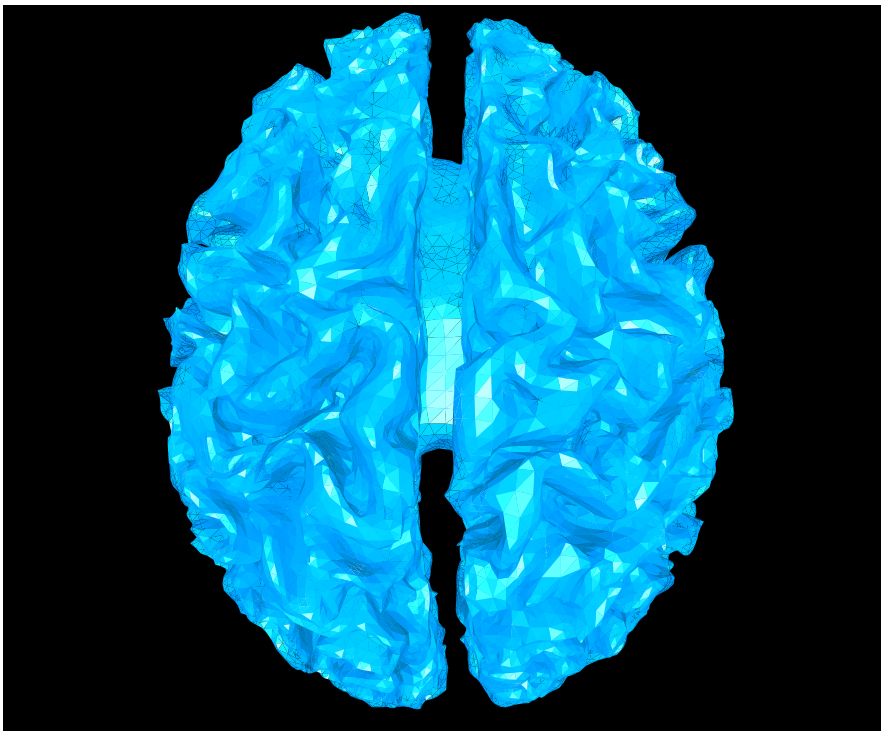

Figure 17: Simplicial surface extracted by the algorithm in [7] from the well-composed counterpart of the brain image used in the experiment in Section 7.

Despite of the good performance of our repairing algorithm in the experiment in Section 7 and of the upper bound given in Section 6 for the expected number of new critical configurations created by our repairing algorithm, our algorithm may in principle (and with very low probability) change the color of a large number of background points of the input image. So, we are currently investigating the existence of a deterministic, polynomial-time repairing algorithm that changes the color of a smallest possible set of background points. We do not know if such an algorithm exists, and we have not yet ruled out the possibility that the *decision* problem equivalent to our optimization problem of finding such a smallest set of background points is **NP**-complete. By the equivalent decision problem, we mean the following problem: Given any 3D binary digital image (D, X) and a positive integer k, can we decide, in polynomial time in $|D|$, whether or not there is a subset P of X_0 such that $(D, X \cup P)$ is well-composed and $|P| \leq k$?

Note that we are considering the case in which background points are made into foreground points, but not the other way around, as it is the case with our repairing algorithm.

We plan to improve the random procedure used by our algorithm to select which background points will become foreground points in order to eliminate critical configurations. Recall that our algorithm chooses points from a very small subset of the background based on whether or not these points create at least one new critical configuration. It would be interesting to evaluate the effect of consecutive choices of points by "looking ahead" the number of new critical configurations created by the algorithm. Then, we could choose a "chain" of changes of colors that lead to less new critical configurations. Certainly, the number of such chains is exponential in the number of points of the image. So, we must evaluate the trade off between runtime and the size of the set of background points whose color is changed by the algorithm.

References

[1] L. Latecki. 3d well-composed pictures. *Graphical Models and Image Processing*, 59(3):164–172, 1997.

[2] L. Latecki, U. Eckhardt, and A. Rosenfeld. Well-composed sets. *Computer Vision and Image Understanding*, 61(1):70–83, 1995.

[3] J. Marchadier, D. Arques, and S. Michelin. Thinning grayscale well-composed images. *Pattern Recognition Letters*, 25(5):581–590, 2004.

[4] A. Rosenfeld, T.Y. Kong, and A. Nakamura. Topology-preserving deformations of two-valued digital pictures. *Graphical Models and Image Processing*, 60(1):24–34, 1998.

[5] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, volume 21, pages 163–169, 1987.

[6] R. Shekhar, E. Fayyad, R. Yagel, and J.F. Cornhill. Octree-based decimation of marching cubes surfaces. In *Proceedings of the IEEE Visualization '96*, pages 335–342, 1996.

[7] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[8] L. Latecki, C. Conrad, and A. Gross. Preserving topology by a digitization process. *Journal of Mathematical Imaging and Vision*, 8(2):131–159, 1998.

[9] L. Latecki. *Discrete Representation of Spatial Objects in Computer Vision*. Kluwer Academic Publishers, 1998.

[10] T. Y. Kong. Topology-preserving deletion of 1's from 2-, 3- and 4-dimensional binary images. In *Proceedings of the DCGI'97*, volume 1347 of *Lecture Notes in Computer Science*, pages 3–18. Springer-Verlag, 1997.

[11] G. Herman. *Geometry of Digital Spaces*. Springer-Verlag, 1998.

[12] E. Bloch. *A First Course in Geometric Topology and Differential Geometry*. Birkhäuser, 1997.

[13] D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.

[14] G. Herman and B. Carvalho. Multiseeded segmentation using fuzzy connectedness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):460–474, 2001.

[15] J.-O. Lachaud. Topologically defined isosurfaces. In *Proceedings of the DGCI'96*, volume 1176 of *Lecture Notes in Computer Science*, pages 245–256. Spriger-Verlag, 1996.

[16] G. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proceedings of the IEEE Visualization '92*, pages 83–91, 1992.

[17] A. van Gelder and J. Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.

[18] B. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994.

[19] J. Bloomenthal. An implicit surface polygonizer. In P. Heckbert, editor, *Graphics Gems IV*, pages 324–350. Morgan Kaufmann, 1994.

[20] E. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical Report CN/95-17, CERN, 1995.

[21] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptively trilinear surfaces. *Computers and Graphics*, 24(3):399–418, 2000.

[22] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):16–29, 2003.

[23] B. Stander and J. Hart. Guaranteeing the topology of an implicit surface polygonizer for interactive modeling. In *Proceedings of the SIGGRAPH'97*, pages 279–286, 1997.

[24] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Meshing implicit surfaces with certified topology. Technical Report 4930, Institut National de Recherche en Informatique et en Automatique (INRIA), Sophia Antipolis, Cedex, France, 2003.

[25] G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. *Computational Geometry: Theory and Applications*, 10(3):155–170, 1998.

[26] J. Bloomenthal and C. Bajaj. *Introduction to Implicit Surfaces*. Morgan-Kaufmann, 1997.

[27] R. Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.