March 2008

Topological Repairing of 3D Digital Images

Marcelo Siqueira
Universidade Federal de Mato Grosso do Sul

Longin Jan Latecki
University of Pennsylvania

Nicholas Tustisin
University of Pennsylvania

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

James C. Gee
University of Pennsylvania, gee@mail.med.upenn.edu

Postprint version. Published in *Journal of Mathematical Imaging and Vision*, Volume 30, Issue 3, March 2008, pages 249-274.
Publisher URL: http://dx.doi.org/10.1007/s10851-007-0054-1

Topological Repairing of 3D Digital Images

Abstract
We present here a new randomized algorithm for repairing the topology of objects represented by 3D binary digital images. By "repairing the topology", we mean a systematic way of modifying a given binary image in order to produce a similar binary image which is guaranteed to be *well-composed*. A 3D binary digital image is said to be well-composed if, and only if, the square faces shared by background and foreground voxels form a 2D manifold. Well-composed images enjoy some special properties which can make such images very desirable in practical applications. For instance, well-known algorithms for extracting surfaces from and thinning binary images can be simplified and optimized for speed if the input image is assumed to be well-composed. Furthermore, some algorithms for computing surface curvature and extracting adaptive triangulated surfaces, directly from the binary data, can only be applied to well-composed images. Finally, we introduce an extension of the aforementioned algorithm to repairing 3D digital multivalued images. Such an algorithm finds application in repairing segmented images resulting from multi-object segmentations of other 3D digital multivalued images.

Keywords
well-composed images, digital topology, randomized algorithms

Comments
Postprint version. Published in *Journal of Mathematical Imaging and Vision*, Volume 30, Issue 3, March 2008, pages 249-274.
Publisher URL: http://dx.doi.org/10.1007/s10851-007-0054-1

Topological Repairing of 3D Digital Images

Marcelo Siqueira[†]
Longin Jan Latecki[‡]
Nicholas Tustison[♭]
Jean Gallier[*]
James Gee[♭]

[†]Departamento de Computação e Estatística
Universidade Federal de Mato Grosso do Sul
Av. Costa e Silva, S/N, Campus Universitário
Campo Grande, MS 79070-900, Brazil
marcelo@dct.ufms.br

[‡]Department of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
latecki@temple.edu

[♭]Penn Image and Computing Sciences Laboratory
Department of Radiology
University of Pennsylvania
Philadelphia, PA 19104, USA
tustison@grasp.cis.upenn.edu
gee@mail.med.upenn.edu

[*]Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
jean@cis.upenn.edu

Abstract

We present here a new randomized algorithm for repairing the topology of objects represented by 3D binary digital images. By "repairing the topology", we mean a systematic way of modifying a given binary image in order to produce a similar binary image which is guaranteed to be *well-composed*. A 3D binary digital image is said to be well-composed if, and only if, the square faces shared by background and foreground voxels form a 2D manifold. Well-composed images enjoy some special properties which can make such images very desirable in practical applications. For instance, well-known algorithms for extracting surfaces from and thinning binary images can be simplified and optimized for speed if the input image is assumed to be well-composed. Furthermore, some algorithms for computing surface curvature and extracting adaptive triangulated surfaces, directly from the binary data, can only be applied to well-composed images. Finally, we introduce an extension of the aforementioned algorithm to repairing 3D digital multivalued images. Such an algorithm finds application in repairing segmented images resulting from multi-object segmentations of other 3D digital multivalued images.

keywords: well-composed images, digital topology, randomized algorithms.

1 Introduction

In several important applications, such as computer-aided diagnosis, videoconferencing, and fluid dynamics simulation, geometric objects are represented by 3D digital (multivalued) images. A 3D digital image consists of a finite grid of points of $\mathbb{Z}^3$, each of which is assigned some value in a gray-level or color scale via a *digitization* process. In general, the digitization process is carried out by a sampling device, such as a CT scanner or a CCD camera, which assigns a value to a point of the image grid that is proportional to some physical quantity measured by the device at the point location.

By a *segmentation* process, the image points are grouped together to form *digital sets*. For example, for simple threshold binary segmentation, the points whose intensity values are greater than a given threshold value are classified as belonging to a certain digital set. The output of the segmentation process is a 3D digital *binary* image in which the points representing the digital set are assigned the value 1 and the remaining points are assigned the value 0. By a *reconstruction* process, we commonly identify each point assigned the value 1 by the segmentation process with a cuboid (voxel) centered at the point. The union of these voxels forms a subset of $\mathbb{R}^3$, called the *continuous analog of the foreground*, which can be viewed as the "continuous" representation of one or more objects represented by the multivalued image.

Whenever a 3D digital image is used to represent geometric objects in practical applications, the continuous analog of the foreground corresponding to any object in the image is expected to exhibit some fundamental properties of the object. In particular, the topology of the continuous analog of the foreground is expected to be the same as the topology of the object. Unfortunately, due to several factors related to the digitization process, the segmentation and reconstruction processes may not be able to produce a topologically correct continuous analog of the foreground [1, 2]. To make things worse, the topology of the object may not be known or even available for the application, which makes the correction of the topology of the continuous analog of the foreground generally not feasible.

In many applications, the only geometric objects involved are *solids*, i.e., objects that can be viewed as bounded and closed subsets of $\mathbb{R}^3$ whose boundary is a (topological) surface (i.e., a 2D manifold). If a 3D digital binary image correctly represents a solid object, then the image must be *well-composed*, i.e., the boundary of the continuous analog of the foreground representing a solid must be a surface [3]. If it is not, we can be sure that the topology of the continuous analog is incorrect. However, if we do not know the topology of the solid or are not given any information that allow us to derive the correct topology, any attempt to modify the topology of the continuous analog is at best a good guess.

Despite the resulting topological ambiguities, there are still practical advantages in modifying the incorrect 3D digital binary image so that the boundary of the continuous analog of the foreground becomes a surface, i.e., in order to produce a well-composed image. This is true even if the resulting

well-composed image is not the correct one, i.e., even if the surface corresponding to the boundary of the continuous analog of the foreground is not topologically equivalent to the surface of the solid. The reason is that well-composed images have very interesting topological properties, which enable us to simplify and optimize for speed algorithms commonly used in computer graphics, image processing and computer vision applications.

In particular, if the input image is well-composed, the popular Marching Cubes (MC) algorithm for extracting triangulated isosurfaces from 3D binary digital images will always produce a topologically consistent surface [4]. The reason is that the "ambiguity" problem will not occur [5]. Although there are several extensions of the MC algorithm that ensure the extraction of a topologically consistent surface from any given 3D digital binary image (see [6, 7, 8, 9, 10] to name a few), these algorithms demand more work if the input image is not well-composed. In addition, thinning algorithms can be simplified and naturally made parallel if the input image is well-composed [11, 12], and some algorithms for computing surface curvature [13, 14] or extracting adaptive triangulated surfaces [15], which operate *directly* on the binary data, do assume that the input image is well-composed.

The aforementioned advantages of well-composed images motivated the development of the so-called *repairing algorithms* [16], which are algorithms for modifying a 3D digital binary image (that is not well-composed) to produce a well-composed image. A repairing algorithm can be used whenever a 3D digital binary image representing a solid is not well-composed and we have no means to find out the correct topology of the solid. However, in order to be really useful in practice, a repairing algorithm must produce a well-composed image by making only a few modifications in the input binary image, so that the the resulting well-composed image will not differ very much from the input image.

Here, we introduce a new repairing algorithm for generating a 3D well-composed image from a given 3D digital binary image. The key operation of our algorithm is to change the value assigned to a point of a binary image from 0 to 1. The result is a new binary image. The operation is firstly executed on the input binary image, and then on the image obtained by the previous execution of the operation. The algorithm stops when the image resulting from the most recent execution of the key operation is well-composed, which we prove to be the case after a finite amount of executions of the operation.

Our algorithm is randomized, and its time and space complexities are linear in the number of image points. We provide an upper bound on the expected point difference between the input image and its well-composed counterpart produced by the algorithm. We also introduce an extension of our algorithm that repairs 3D digital multivalued images. We assume that the given multivalued image is the result of a multi-object segmentation of another 3D digital multivalued image (see [17] for an example of such a segmentation algorithm). The repaired image is also a well-composed image with respect to each digital set corresponding to a distinct object of the input segmented image.

The remainder of this paper is organized as follows. Section 2 reviews some relevant and related work. Section 3 introduces some basic concepts of digital topology. Section 4 gives an important characterization of 3D well-composed images. Section 5 describes the algorithm for producing a 3D well-composed image from a 3D digital binary image, and derives an upper bound on the expected point-wise difference between the input image and its well-composed counterpart produced by the algorithm. Section 6 introduces the extension of the algorithm in Section 5 to dealing with 3D digital multivalued images. Section 7 shows the results of the application of our algorithms on medical images. Finally, Section 8 summarizes our contributions and results, and discusses future work.

2 Related Work

Latecki, Eckhardt, and Rosenfeld [11] provided a formal definition of 2D well-composed images, and characterized these images by the absence of certain 2×2 neighborhoods of image points, the so-called *critical configurations*. Later, Latecki [3] extended the notion of well-composedness to three dimensions, and characterized 3D well-composed images in terms of the absence of certain 2×2 and $2 \times 2 \times 2$ neighborhoods of image points, which are the corresponding critical configurations in three dimensions.

Latecki, Conrad, and Gross [1] showed that a topology-preserving digitization process of a 2D *r-regular* object (i.e., the 2D analogous of a solid) must yield a 2D well-composed image. Furthermore, they derived conditions relating the properties of a 2D r-regular object to the image grid size such that if these conditions are met by the digitization process, then the topology of the object can be correctly recovered from the image by the segmentation and reconstruction processes. Unfortunately, the conditions derived in [1] depends on the knowledge of properties of the object that are hardly known in practice or they cannot be met by the sampling device available for the digitization process.

Stelldinger and Köthe [2] showed that the conditions given in [1] are not sufficient to guarantee that the topology of a 3D r-regular object (i.e., a solid) can be correctly recovered from the image by the segmentation and reconstruction processes. Later, Stelldinger, Latecki, and Siqueira [18] gave sufficient conditions to correctly reconstruct the topology of a solid from a 3D digital binary image. However, these conditions suffer from the same practical limitations that the conditions given in [1] do.

The idea of modifying a binary image to produce a well-composed one was given by Latecki [16], who also proposed the first algorithm for producing a 2D well-composed image from a 2D digital binary image. However, his algorithm cannot be extended to dealing with 3D digital binary images. Later, Rosenfeld, Kong, and Nakamura [19] introduced an image operator, called *simple deformation*, that can be used to produce 2D and 3D well-composed, but the resulting well-composed

images have 9 and 27 times more points than the input 2D and 3D binary images[1], respectively. To our best knowledge, the algorithm described in this paper for producing a 3D well-composed image from a 3D digital binary image is the first one to generate an output image with the same size as the input image.

Our repairing algorithm is related to several results regarding the correct segmentation and reconstruction of the human brain cortical surface from MR images [20, 21, 22, 23, 24, 25, 26, 27, 28]. The cerebral cortex is the largest part of the human brain, and its correct reconstruction is an important goal in medicine and neuroscience. Although the human brain cortex is highly folded, its intrinsic structure is that of a two-dimensional sheet, several millimeters thick. More specifically, the cortex is a thin folded sheet of gray matter (GM) that lies inside the cerebrospinal fluid (CSF) and outside the white matter (WM) of the brain. If the opening at the brain stem is artificially closed, the surface of the cortex has the topology of a sphere, and the major topological problem that results from an incorrect digitization or segmentation is the presence of handles in the reconstructed surface [24].

Algorithms for segmenting and reconstructing cortical models can be broadly divided into two types: those that incorporate some sort of topology-preserving mechanism into the segmentation process [20, 21, 25, 26], and those that that do not [22, 29, 23, 24, 28, 27]. The topology-preserving algorithms typically start with a surface of known topology and then iteratively warp it so that it closely approximates the geometry of the cortical surface. There are two major problems regarding topology-preserving segmentation algorithms. First, they may lead to large geometric inaccuracies. Second, they also require an initialization close to the cortex. So, correcting the cortex topology is often necessary, either before or after an initial, unconstrained segmentation [27], which can be performed using local intensity, prior probabilities, and geometric information without regard to topology [30].

There are several algorithms to retrospectively correct the topology of an already segmented MR brain image. The underlying idea behind them is to identify handles and then choose between cutting a handle or filling a hole. Some algorithms assume that the handles are located at the thinnest parts of the image region of interest, and make their decision by minimally modifying the region or a triangulated surface approximating the region boundary [23, 24]. Although they often lead to accurate results, topological corrections may not be optimal. Some other algorithms have achieved better results by integrating statistical or geometric information to the decision making process [22, 29, 28]. More recently, Bazin and Pham [27] proposed an algorithm that corrects the topology of an already segmented image using a topology-preserving distance function. Their algorithm modifies the color associated with the image points in order to preserve the topology of an isosurface defined by a distance function. The topological changes are detected by keeping track of the function critical points.

There are two major differences between the above algorithms and the repairing algorithms

[1]The algorithm in [19] inserts extra "slices" in the original image.

presented here. First, the above algorithms assume that the desired topology is known *a priori*, while our repairing algorithms do not. More specifically, our repairing algorithms solve a related, but yet different problem from the one of correcting topology: the problem of restoring the manifold property of digital sets. In this problem, the correct topology is not known and the goal is to obtain a well-composed image that is similar to the input image. The topology of the resulting image may not be correct. As we pointed out before, there are several applications in computer graphics, image processing, and computer vision that can benefit from dealing with well-composed images. Second, most of the above topology correction algorithms are restricted to correcting the topology of MR images of the human brain cortex, while our repairing algorithms may be applied to more general MR images.

Our repairing algorithms are also related to algorithms for simplifying the topology of shapes represented by 3D digital images [31, 32, 33, 34]. By "simplifying topology", we mean the removal of topological artifacts in the form of tiny handles from the image isosurfaces. These nearly invisible artifacts are responsible for slowing down the performance of algorithms for isosurface simplification, remeshing, and parametrization. Like the repairing algorithms presented here, the simplification algorithms in [31, 32, 33, 34] do not assume any prior knowledge of the correct topology, and they also produce images that are suitable for some particular applications. Despite these similarities, the images produced by topology simplification algorithms are not necessarily well-composed, and the underlying ideas of some of those algorithms are quite different from the ones of our repairing algorithms.

3 Preliminaries

This section introduces some basic concepts of digital topology, which is the field that studies the topological properties of digital images. We refer the reader to [35, 36] for detailed discussions of the concepts introduced here.

For any positive integer δ, let

$$\delta\mathbb{Z}^3 = \{(\delta \cdot z_1, \delta \cdot z_2, \delta \cdot z_3) \in \mathbb{Z}^3 \mid (z_1, z_2, z_3) \in \mathbb{Z}^3\}.$$

Every point p in $\delta\mathbb{Z}^3$ is called a *grid point* and is the center of a *grid cube* with edges of length $\delta \in \mathbb{R}^+$, oriented parallel to the Cartesian coordinate axes. We denote the grid cube centered at p by $\mathcal{V}(p)$, and we commonly refer to a grid cube as a *voxel*, to a corner of a voxel as a *grid vertex*, to an edge of a voxel as a *grid edge*, and to the side of a voxel as a *grid square*. Note that a grid vertex may be a point in $\mathbb{R}^3$.

In what follows we define several important adjacency and connectedness relations on $\delta\mathbb{Z}^3$, for any positive integer δ:

Definition 3.1. *Two distinct points $p = (p_1, p_2, p_3)$ and $q = (q_1, q_2, q_3)$ of $\delta\mathbb{Z}^3$ are said to be* face-

adjacent *if* $|p_1 - q_1| + |p_2 - q_2| + |p_3 - q_3| = \delta$*, or equivalently, if* $\mathcal{V}(p)$ *and* $\mathcal{V}(q)$ *share a grid square.*

Definition 3.2. *Two distinct points* $p = (p_1, p_2, p_3)$ *and* $q = (q_1, q_2, q_3)$ *of* $\delta\mathbb{Z}^3$ *are said to be* edge-adjacent *if* $|p_i - q_i| = \delta$ *and* $|p_j - q_j| = \delta$*, for any* $i, j \in \{1, 2, 3\}$*, with* $i \neq j$*, and* $p_k = q_k$*, for* $k \in \{1, 2, 3\}$*, with* $k \neq i$ *and* $k \neq j$*; or equivalently, if* $\mathcal{V}(p)$ *and* $\mathcal{V}(q)$ *share a grid edge but not a grid square.*

Definition 3.3. *Two distinct points* $p = (p_1, p_2, p_3)$ *and* $q = (q_1, q_2, q_3)$ *of* $\delta\mathbb{Z}^3$ *are said to be* corner-adjacent *if* $|p_i - q_i| = \delta$*, for all* $i \in \{1, 2, 3\}$*, or equivalently, if* $\mathcal{V}(p)$ *and* $\mathcal{V}(q)$ *share a grid vertex but not a grid edge.*

For an example, refer to Figure 1 and suppose that the points a, b, c, and d in the figure are all grid points of the set $\delta\mathbb{Z}^3$. Then, points a and b are face-adjacent, points b and c are edge-adjacent but not face-adjacent, and points c and d are corner-adjacent but not edge-adjacent nor face-adjacent.

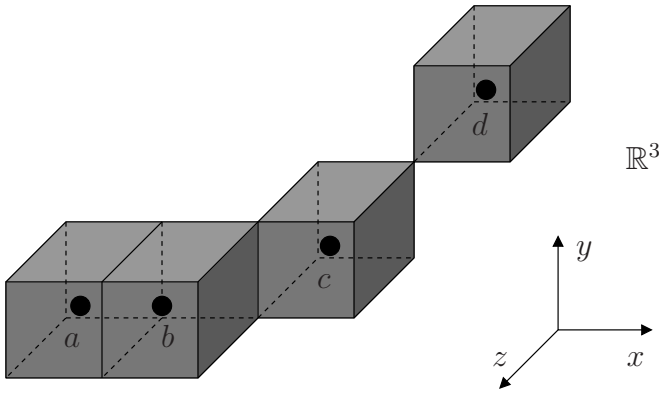

Figure 1: Four points of $\delta\mathbb{Z}^3$ and their corresponding voxels.

The face-adjacency relation is also known as 6-*adjacency*. Two other relevant adjacency relations are the 18- and 26-adjacency relations. Two distinct points of $\delta\mathbb{Z}^3$ are said to be 18-*adjacent* if they are face- or edge-adjacent, and 26-*adjacent* if they are face-, edge-, or corner-adjacent. For instance, in Figure 1, points a and b are 6-adjacent, 18-adjacent, and 26-adjacent, points b and c are 18-adjacent and 26-adjacent but not 6-adjacent, and points c and d are 26-adjacent but not 18-adjacent.

Definition 3.4. *Let* A *be any subset of points of* $\delta\mathbb{Z}^3$*, and let* $\rho \in \{6, 18, 26\}$*. For any* p *and* q *in* A*, the sequence* $\langle x^{(0)}, \ldots, x^{(n)} \rangle$ *of points of* A*, where* $n \in \mathbb{Z}$*, with* $n \geq 0$*, is said to be a* ρ-path in A connecting p to q *if* $x^{(0)} = p$*,* $x^{(n)} = q$*, and* $x^{(i-1)}$ *is* ρ*-adjacent to* $x^{(i)}$*, where* $i \in \mathbb{Z}$ *and* $1 \leq i \leq n$*. In particular, there are* ρ*-paths of length zero; for instance,* $\langle p \rangle$*. We refer to* ρ*-paths of length zero 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*.*

Definition 3.5. *Let A be any subset of points of $\delta\mathbb{Z}^3$, and let $\rho \in \{6, 18, 26\}$. We say that A is* ρ-connected *if for all $p, q \in A$, there is a ρ-path in A connecting p to q. If B is a maximal ρ-connected subset of A then we say that B is a* ρ-connected component*, or simply* ρ-component*, of A.*

For an example, suppose that A is the set of points in Figure 1. Then, point a is 6-connected to point b, 18-connected to point c, and 26-connected to point d. So, there is a 26-path of length 3 in A connecting a to d. We also have that A is 26-connected and that $B = \{a, b, c\}$ is a 18-component of A. Note that ρ-connectedness is an equivalence relation, i.e., it is reflexive, symmetric, and transitive. So, every ρ-component of A is nonempty and any two distinct ρ-components of A are disjoint.

Definition 3.6. *Let A be any subset of $\delta\mathbb{Z}^3$. We define the set*

$$bd(A) = \{(p, q) \in \delta\mathbb{Z}^3 \times \delta\mathbb{Z}^3 \mid p \in A,\ q \in \overline{A},\ \textit{and } p \textit{ and } q \textit{ are face-adjacent}\},$$

which is called the digital boundary in $\delta\mathbb{Z}^3$ between A and $\overline{A}$*, where $\overline{A}$ is the complement of A with respect to $\delta\mathbb{Z}^3$.*

Given any subset A of $\delta\mathbb{Z}^3$, we define

$$\mathcal{V}(A) = \bigcup_{p \in A} \mathcal{V}(p) \quad \text{and} \quad \mathcal{V}(bd(A)) = \bigcup_{(p,q) \in bd(A)} (\mathcal{V}(p) \cap \mathcal{V}(q)),$$

i.e., $\mathcal{V}(A)$ is the point set corresponding to the union of all voxels of points in A, and $\mathcal{V}(bd(A))$ is the point set corresponding to the union of all grid squares shared by a point in A and a point not in A. Note that $\mathcal{V}(bd(A))$ is the (topological) boundary of $\mathcal{V}(A)$ in $\mathbb{R}^3$. We call $\mathcal{V}(A)$ and $\mathcal{V}(bd(A))$ the *continuous analog* of A and the *continuous analog of the digital boundary in $\delta\mathbb{Z}^3$ between A and $\overline{A}$*, respectively [3].

Definition 3.7. *A* 3D digital (multivalued) image *is a function*

$$\mathcal{F} : \mathbb{G}_{n_1,n_2,n_3,\delta} \to C$$

from a nonempty and finite subset of $\delta\mathbb{Z}^3$,

$$\mathbb{G}_{n_1,n_2,n_3,\delta} = \{(g_1, g_2, g_3) \in \delta\mathbb{Z}^3 \mid g_i = \delta \cdot d_i,\ d_i \in [1, n_i],\ i \in \{1, 2, 3\}\},$$

where n_1, n_2, n_3, and δ are all positive integers, to a nonempty and finite subset of $\mathbb{Z}$. The domain $\mathbb{G}_{n_1,n_2,n_3,\delta}$ of $\mathcal{F}$ is called a 3D grid with spacing δ and size $n_1 \times n_2 \times n_3$*. The elements of the co-domain C of $\mathcal{F}$ are called* colors*. So, the image $\mathcal{F}$ assigns a color $\mathcal{F}(p)$ from C to each grid point $p \in \mathbb{G}_{n_1,n_2,n_3,\delta}$.*

A special type of 3D digital image is the *3D binary image* which is a two-valued 3D digital image whose co-domain is $\{0, 1\}$. We shall denote a 3D binary image $\mathcal{F}$ by the pair $(\mathbb{G}_{n_1,n_2,n_3,\delta}, X)$, where $\mathbb{G}_{n_1,n_2,n_3,\delta}$ is the image grid and X is the subset of all points p of $\mathbb{G}_{n_1,n_2,n_3,\delta}$ such that $\mathcal{F}(p) = 1$. We commonly refer to X and to $\mathbb{G}_{n_1,n_2,n_3,\delta} - X$ as the *foreground* and *background* of the image $(\mathbb{G}_{n_1,n_2,n_3,\delta}, X)$, respectively. From now on, *we will assume that* $\delta = 1$. The reason is that the algorithms and proofs described henceforward do not depend on the "size" of the image voxels, and the use of δ as an arbitrary parameter can mislead the reader to think otherwise. For simplicity, we will also let $\mathbb{G}$ denote a 3D grid $\mathbb{G}_{n_1,n_2,n_3,1}$ with spacing 1 and size $n_1 \times n_2 \times n_3$, for some given positive integers n_1, n_2, n_3, and we may sometimes denote the foreground and background of $(\mathbb{G}, X)$ by X_1 and X_0, respectively.

4 3D Well-Composed Images

This section gives the formal definition of 3D well-composed images and provides a characterization of 3D well-composed images in terms of certain configurations of grid points, called critical configurations.

Definition 4.1 ([3])**.** *Let $(\mathbb{G}, X)$ be any 3D digital binary image. We say that $(\mathbb{G}, X)$ is a 3D* well-composed *binary image, or simply* well-composed*, if, and only if, the continuous analog $\mathcal{V}(bd(X_1))$ of the digital boundary $bd(X_1)$ in $\mathbb{Z}^3$ between X_1 and $\overline{X_1} = (\mathbb{Z}^3 - X_1)$ is a (topological) surface (i.e., a 2D manifold) in $\mathbb{R}^3$.*

Recall that a subset $S \subset \mathbb{R}^3$ is called a *topological surface*, or just *surface* for short, if each point $p \in S$ has an open neighborhood $\mathcal{N}_\epsilon(p) = \{q \in S \mid d(p, q) < \epsilon\}$ that is homeomorphic to the open disk

$$\mathbb{D}^2 = \{x = (x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 < 1\},$$

where $d(x, y)$ is the Euclidean distance from x to y, with $x, y \in \mathbb{R}^3$, and ϵ is some positive real number.

For instance, if $(\mathbb{G}, X)$ is the binary image whose foreground $X_1 = X$ is the set of points $\{a, b, c, d\}$ in Figure 1, then $(\mathbb{G}, X)$ is not well-composed, as any point in the edge $\mathcal{V}(b) \cap \mathcal{V}(c)$, as well as the point $\mathcal{V}(c) \cap \mathcal{V}(d)$, have no open neighborhood that is homeomorphic to $\mathbb{D}^2$. But, if X_1 is the set $\{a, b\}$, then $(\mathbb{G}, X)$ is well-composed, as $\mathcal{V}(bd(X_1))$ is the boundary of a parallelepiped (i.e., a surface in $\mathbb{R}^3$).

Although Definition 4.1 does not provide us with any explicit property that could be used to decide whether a given image is well-composed, well-composedness is indeed equivalent to the absence of certain configurations of subsets of four and eight points of the image grid as defined below (refer to Figure 2):

Definition 4.2. *Let A be any set of four points of $\mathbb{G}$. We say that A is an instance of the* critical configuration 1 *in $(\mathbb{G}, X)$, or C1 for short, if two points of A are in X_1, the other two are in $\overline{X}_1$, the two points in X_1 (resp. $\overline{X}_1$) are edge-adjacent, and the voxels of the four points share an edge.*

Definition 4.3. *Let A be any set of eight points of $\mathbb{G}$. We say that A is an instance of the* critical configuration 2 *in $(\mathbb{G}, X)$, or C2 for short, if two (resp. six) points of A are in X_1, the other six (resp. two) are in $\overline{X}_1$, the two points in X_1 (resp. $\overline{X}_1$) are corner-adjacent, and the voxels of the eight points share a corner.*

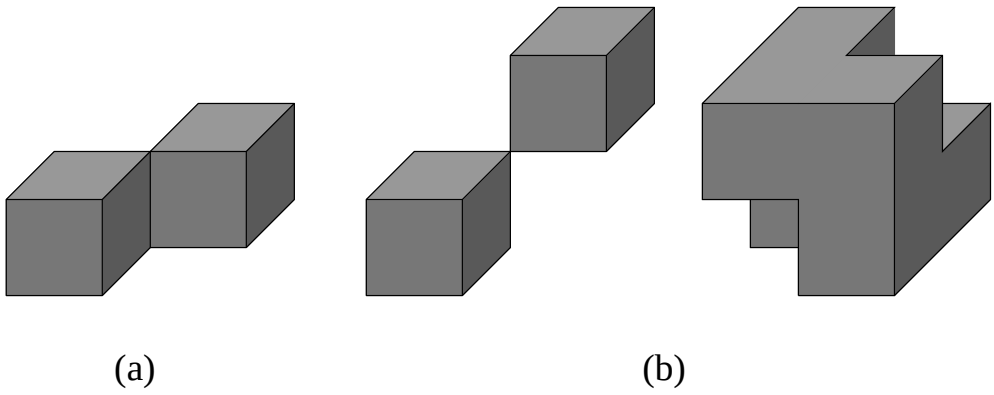

Figure 2: . (a) Critical configuration 1. (b) Critical configuration 2. For the sake of clarity, only the two voxels in X_1 (or $\overline{X}_1$) are shown in (a), and only the two (or six) voxels in X_1 (or $\overline{X}_1$) are shown in (b).

The following theorem stated and proved by Latecki [3] establishes an equivalence between a 3D well-composed image and the absence of instances of the critical configurations 1 and 2 in the image grid:

Theorem 4.1. *A 3D digital binary image $(\mathbb{G}, X)$ is well-composed if, and only if, there is no instances of C1 and C2 in $(\mathbb{G}, X)$.*

There is a straightforward algorithm to decide whether a given 3D digital binary image $(\mathbb{G}, X)$ is well-composed: for each subset A of four (resp. eight) points of $\mathbb{G}$, whose voxels share an edge (resp. a corner), it suffices check if A is an instance of C1 (resp. C2) in $(\mathbb{G}, X)$ using Definitions 4.2 and 4.3.

Theorem 4.1 also implies that there is only one kind of ρ-connectedness in well-composed images, for $\rho \in \{6, 18, 26\}$; that is, every 26-component of the foreground X (resp. background X_0) of $(\mathbb{G}, X)$ is a 18-component of X (resp. X_0), which in turn is a 6-component of X (resp. X_0). However, the converse is not generally true; that is, if every 26-component of X (resp. X_0) is a 18-component of X (resp. X_0), and every 18-component of X is a 6-component of (resp. X_0), the image is not

necessarily well-composed. For instance, suppose that $\mathbb{G}$ is the grid $[0,3] \times [0,3] \times [0,1]$, and let X be the set

$$X = \{(1,1,0),(2,2,0)\} \cup \{(x,y,z) \in \mathbb{G} \mid z = 1\},$$

as shown in Figure 3. Note that the foreground X (resp. background X_0) of $(\mathbb{G}, X)$ has only one ρ-component, for any $\rho \in \{6, 18, 26\}$. The 26-component of X (resp. X_0) is a 18-component of X (resp. X_0), and the 18-component of X (resp. X_0) is in turn a 6-component of X (resp. X_0). Yet, $(\mathbb{G}, X)$ is not well-composed.

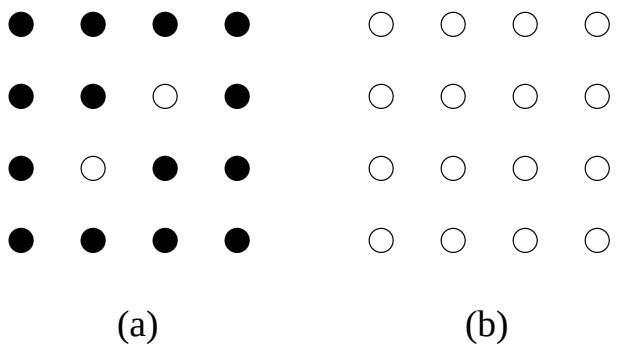

Figure 3: (a) Points with coordinates $(x, y, 0)$ and (b) points with coordinates $(x, y, 1)$ of the 3D binary digital image $(\mathbb{G}, X)$, where $\mathbb{G} = [0,3] \times [0,3] \times [0,1]$ and $X = \{(1,1,0),(2,2,0)\} \cup \{(x,y,z) \in \mathbb{G} \mid z = 1\}$. Foreground and background points are represented by open circles and solid circles, respectively.

5 A Repairing Algorithm for 3D Binary Images

This section presents a new algorithm for repairing a given 3D binary image $(\mathbb{G}, X)$ that is not well-composed. By repairing, we mean to produce a 3D well-composed binary image $(\mathbb{G}, X')$ by iteratively changing the color value of certain background of $(\mathbb{G}, X)$, so that they become foreground points of $(\mathbb{G}, X')$. The algorithm is randomized and has linear space and time complexities in $|\mathbb{G}|$, where $|\mathbb{G}|$ is the number of grid points. The input of the algorithm consists of $\mathbb{G}$ and X, and its output is X'.

To compute X', the algorithm finds a subset P of the background X_0 of $(\mathbb{G}, X)$ such that $(\mathbb{G}, X \cup P)$ is well-composed, and then lets $X' = X \cup P$. So, P can be viewed as a subset of the background whose assigned colors are changed from 0 to 1 in order to produce $(\mathbb{G}, X')$. Such a subset P always exists, as $P = X_0$ would make $(\mathbb{G}, X \cup P) = (\mathbb{G}, \mathbb{G})$, which is clearly a well-composed image. However, in the context of practical applications, it is important to have $(\mathbb{G}, X)$ and $(\mathbb{G}, X')$ as similar as possible.

We can maximize the similarity between $(\mathbb{G}, X)$ and $(\mathbb{G}, X')$ by finding the smallest subset P of X_0 such that $(\mathbb{G}, X \cup P)$ is well-composed. We can trivially do that by enumerating and testing all subsets of X_0. However, this procedure has exponential time complexity in the size $|\mathbb{G}|$ of $\mathbb{G}$, which rules out its practical use. Our algorithm is not guaranteed to find a smallest set P, but it has linear time complexity in $|\mathbb{G}|$. Furthermore, the size of the sets P computed by our algorithm when tested against biomedical images typically found in practical applications were satisfactorily small (see Section 7).

The algorithm starts by letting $P = \emptyset$. Then, it loops over all points of $\mathbb{G}$ seeking for instances of C1 and C2 in $(\mathbb{G}, X)$. If no instance of a critical configuration is found, the algorithm terminates with $X' = X$. Otherwise, for each instance of C1 and C2, the algorithm iteratively inserts points from $X_0 - P$ into P until the image $(\mathbb{G}, X \cup P)$ becomes well-composed. Every time the algorithm inserts one or more points into P, it *eliminates* at least one instance of C1 or C2 from the current image, $(\mathbb{G}, X \cup P)$. However, the point insertion operation may also give rise to other instances of C1 and C2 in $(\mathbb{G}, X \cup P)$, which will be eventually eliminated from the resulting image by further point insertions.

5.1 The Elimination of Critical Configurations

Let n_1, n_2, n_3 be the positive integers defining $\mathbb{G}$ (see Definition 3.7), and for any $p = (p_1, p_2, p_3) \in \mathbb{Z}^3$, let

$$\begin{aligned}
\mathcal{N}_x(p) &= \{(p_1, y, z) \mid y \in \{p_2, p_2 + 1\} \text{ and } z \in \{p_3, p_3 + 1\}\}, \\
\mathcal{N}_y(p) &= \{(x, p_2, z) \mid x \in \{p_1, p_1 + 1\} \text{ and } z \in \{p_3, p_3 + 1\}\}, \\
\mathcal{N}_z(p) &= \{(x, y, p_3) \mid x \in \{p_1, p_1 + 1\} \text{ and } y \in \{p_2, p_2 + 1\}\},
\end{aligned}$$

and

$$\mathcal{N}(p) = \{p_1, p_1 + 1\} \times \{p_2, p_2 + 1\} \times \{p_3, p_3 + 1\}.$$

Figure 4 illustrates the above definitions.

We also define

$$\mathcal{J}(p) = \{\mathcal{N}_x(p), \mathcal{N}_y(p), \mathcal{N}_z(p), \mathcal{N}(p)\}.$$

If any of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$ is an instance of C1 in $(\mathbb{G}, X)$, we say that $\mathcal{J}(p)$ contains an instance of C1. Likewise, if $\mathcal{N}(p)$ is an instance of C2 in $(\mathbb{G}, X)$, we say that $\mathcal{J}(p)$ contains an instance of C2. Note that each instance of C1 consists of the four points in one of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, and $\mathcal{N}_z(p)$, for some $p \in \mathbb{G}$. Likewise, each instance of C2 consists of the eight points in $\mathcal{N}(p)$, for some $p \in \mathbb{G}$. Note also that if $\mathcal{N}(p)$ is an instance of C2 in $(\mathbb{G}, X)$, then none of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, and $\mathcal{N}_z(p)$ can be an instance of C1 in $(\mathbb{G}, X)$. Conversely, if at least one of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, and $\mathcal{N}_z(p)$ is an instance of C1 in $(\mathbb{G}, X)$ then $\mathcal{N}(p)$ cannot be an instance of C2 in $(\mathbb{G}, X)$. So, either $\mathcal{J}(p)$ contains no instances of a critical configuration, or it contains at least one instance of C1, or it contains exactly one instance of C2.

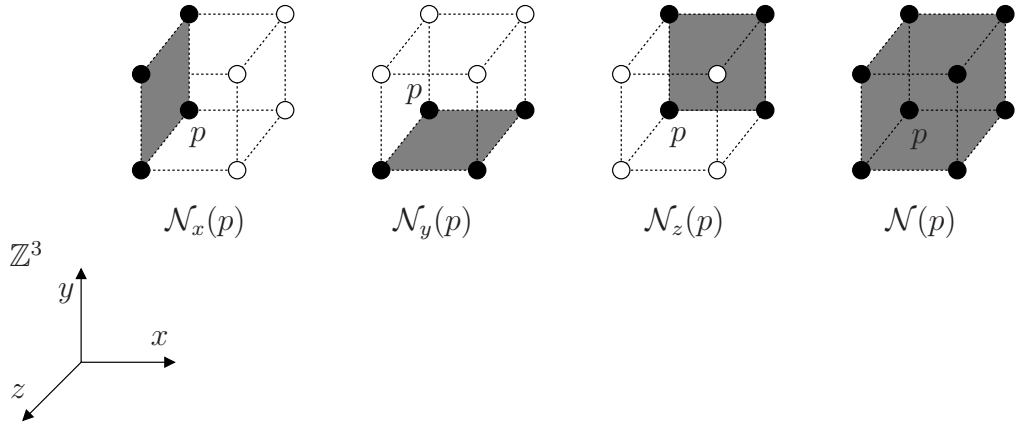

Figure 4: Illustration of the definitions of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$. Points in $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$ are represented by solid circles.

The above observations led us to a simple procedure to find all instances of C1 and C2 in the input image, $(\mathbb{G}, X)$. Let Q be an empty queue. For each point $p \in \mathbb{G}$, check if $\mathcal{J}(p)$ contains a critical configuration. If so, we insert p into Q. If Q remains empty then $(\mathbb{G}, X)$ has no critical configurations and the algorithm terminates with $X' = X$. Otherwise, the algorithm initiates the step of elimination of critical configurations. Note that a point $p \in \mathbb{G}$ is in Q if, and only if, $\mathcal{J}(q)$ contains an instance of C1 or C2. For the purpose of the description of the algorithm, let us assume that Q is not empty.

The step of elimination of critical configurations is basically a loop that iterates until the queue Q becomes empty. For the time being, assume that Q will become empty after a finite number, n, of iterations of the loop. Then, for each $i \in \{1, \ldots, n\}$, the i-th iteration of the loop removes exactly one point q from Q and produces the i-th subset, $X^{(i)}$, of a sequence, $X^{(1)}, \ldots, X^{(n)}$, of subsets of $\mathbb{G}$, where $X \subset X^{(1)}$ and $X^{(j-1)} \subset X^{(j)}$, for each $j \in \{2, \ldots, n\}$. More specifically, for each $i \in \{1, \ldots, n\}$, the subset $X^{(i)}$ is obtained by letting $X^{(i)} = X^{(i-1)} \cup S^{(i)}$, where $X^{(0)} = X$ and $S^{(i)} \subseteq (X_0 - X^{(i-1)})$. In other words, $S^{(i)}$ is a subset of the background of the image $(\mathbb{G}, X^{(i-1)})$, and hence the foreground, $X^{(i)}$, of $(\mathbb{G}, X^{(i)})$ is a superset of the foreground, $X^{(i-1)}$, of $(\mathbb{G}, X^{(i-1)})$. The set $S^{(i)}$ is carefully chosen by the algorithm in such a way that $\mathcal{J}(q)$, where q is the point removed from Q at the beginning of the i-th iteration, does not contain any critical configuration with respect to $(\mathbb{G}, X^{(i)})$. When the loop ends, we let $X' = X^{(n)}$, which is equivalent to say that $X' = X \cup P$, with

$$P = \bigcup_{i=1}^{n} S^{(i)} .$$

To efficiently compute the set $S^{(i)}$, the algorithm relies in three simple facts. First, if q is the point removed from Q in the i-th iteration and $\mathcal{J}(q)$ contains an instance, A, of C1 or C2 in

$(\mathbb{G}, X^{(i-1)})$, then all points of A must belong to $\mathbb{G}$, as $\mathbb{G}$ is a rectangular grid. Second, it can be shown that A will not be a critical configuration in $(\mathbb{G}, X^{(i)})$ only if a subset of the background points of $(\mathbb{G}, X^{(i-1)})$ that belong to A are inserted into $S^{(i)}$. So, the set $S^{(i)}$ can always be computed and the choices of points of $(X_0 - X^{(i-1)})$ to be inserted into $S^{(i)}$ are limited to the background points of $(\mathbb{G}, X^{(i-1)})$ in A. Third, the set $\mathcal{J}(q)$ contains either 0, exactly one, exactly two, or exactly three instances of C1 in $(\mathbb{G}, X^{(i-1)})$, or exactly one instance of C2 in $(\mathbb{G}, X^{(i-1)})$; that is, these five cases are mutually exclusive.

For each of the four mutually exclusive cases in which $\mathcal{J}(q)$ contains an instance of C1 or C2, the algorithm computes a set of sets denoted by $B(q)$. The elements of $B(q)$ are called *choices*, and $S^{(i)}$ is chosen to be one of the elements of the choices. Table 1 describes the choices of $B(q)$ in each case, and Figures 5-9 illustrates these cases. To choose a subset $S^{(i)} \in B(q)$, the algorithm follows a simple *rule of choice*: pick a choice, $S^{(i)}$, of $B(q)$ such that $(\mathbb{G}, X^{(i)})$ does not contain any instance of C1 or C2 that is not also in $(\mathbb{G}, X^{(i-1)})$. If there is more than one choice, pick the one with smallest cardinality. If there is still a tie, break it at random. Finally, if every choice is such that $(\mathbb{G}, X^{(i)})$ contains an instance of C1 or C2 that is not in $(\mathbb{G}, X^{(i-1)})$, then pick any $S^{(i)}$ at random.

Case	**#C1**	**#C2**	**B(q)**	**Figure**
A1	1	0	$\{\{a\},\{b\}\}$	5
A2	2	0	$\{\{a\},\{b,c\}\}$	6
A3(1)	3	0	$\{\{a\},\{b,c,d\}\}$	7
A3(2)	3	0	$\{\{b,c\},\{b,d\},\{c,d\}\}$	7
B(1)	0	1	$\{\{a\},\{b\}\}$	8
B(2)	0	1	$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\}\}$	9

Table 1: Description of the choices of $B(q)$ for each of the four mutually exclusive cases (and their sub-cases) in which $C(p)$ contains an instance of C1 or C2: case denomination (first column), number of instances of C1 in $\mathcal{J}(q)$ (second column), number of instances of C2 in $\mathcal{J}(q)$ (third column), the choices of $B(q)$ (fourth column), and the number of the figure illustrating the choice (fifth column).

It is straightforward to verify that $\mathcal{J}(q)$ cannot contain any instance of C1 nor C2 in $(\mathbb{G}, X^{(i)})$ if, and only if, $S^{(i)} \subseteq (X_0 - X^{(i-1)})$ is one of the choices of $B(q)$. To decide whether a choice for $S^{(i)}$ is such that $(\mathbb{G}, X^{(i)})$ contains an instance of C1 or C2 that is not in $(\mathbb{G}, X^{(i-1)})$, the algorithm does not have to verify all configurations $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$, for every point $p \in \mathbb{G}$. Since $X^{(i)} = X^{(i-1)} \cup S^{(i)}$, it suffices to consider the points that are 26-adjacent to any point in $\mathcal{N}(q)$, or equivalently, the points of the $4 \times 4 \times 4$ grid

$$[q_1 - 1, q_1 + 2] \times [q_2 - 1, q_2 + 2] \times [q_3 - 1, q_3 + 2] \subset \mathbb{Z}^3,$$

where (q_1, q_2, q_3) are the coordinates of q. So, testing for newly *created* critical configurations in $(\mathbb{G}, X^{(i)})$ can be done in constant time. Finally, for each newly created critical configuration, $\mathcal{J}(p)$, in $(\mathbb{G}, X^{(i)})$, the point p is inserted into Q.

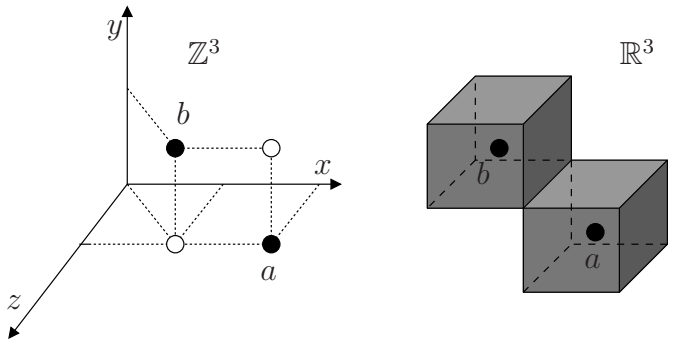

Figure 5: Illustration of case A1 in Table 1. Points a and b are the two background points of $(\mathbb{G}, X^{(i-1)})$ that belong to the instance of C1.

A pseudo code for the step of elimination of critical configurations is given below:

(01) $P \leftarrow \emptyset$
(02) **while** Q is not empty **do**
(03) $\quad q \leftarrow \text{DEQUEUE}(Q)$
(04) $\quad$ **if** $\mathcal{J}(q)$ contains a critical configuration of $(\mathbb{G}, X \cup P)$ **then**
(05) $\quad\quad$ compute $B(q)$
(06) $\quad\quad$ choose S from $B(q)$ according to the *rule of choice*
(07) $\quad$ **endif**
(08) $\quad$ **for** each $p \in \mathbb{G}$ such that $\mathcal{J}(p)$ contains a newly created critical configuration **do**
(09) $\quad\quad \text{ENQUEUE}(p, Q)$
(10) $\quad$ **endfor**
(11) $\quad$ let $P \leftarrow P \cup S$
(12) **endwhile**
(13) **return** $X' = X \cup P$

It remains to be discussed the fact that the queue Q will eventually become empty, which implies the termination of the algorithm. Indeed, this fact follows from the following observations: (1) the **while** loop is entered if, and only if, Q is not empty; (2) every loop iteration of the algorithm removes exactly one point from Q and inserts none, one or more points into Q; (3) each point of $\mathbb{G}$ can be inserted into Q at most four times. In particular, every time a point p of $\mathbb{G}$ is inserted into Q, one element of $\mathcal{J}(p)$ is a newly created critical configuration in $(\mathbb{G}, X \cup P)$. But, each of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$ can become a new critical configuration only once, as the color of the points are changed from 0 to 1 only. So, Q will eventually become empty, and the algorithm will terminate.

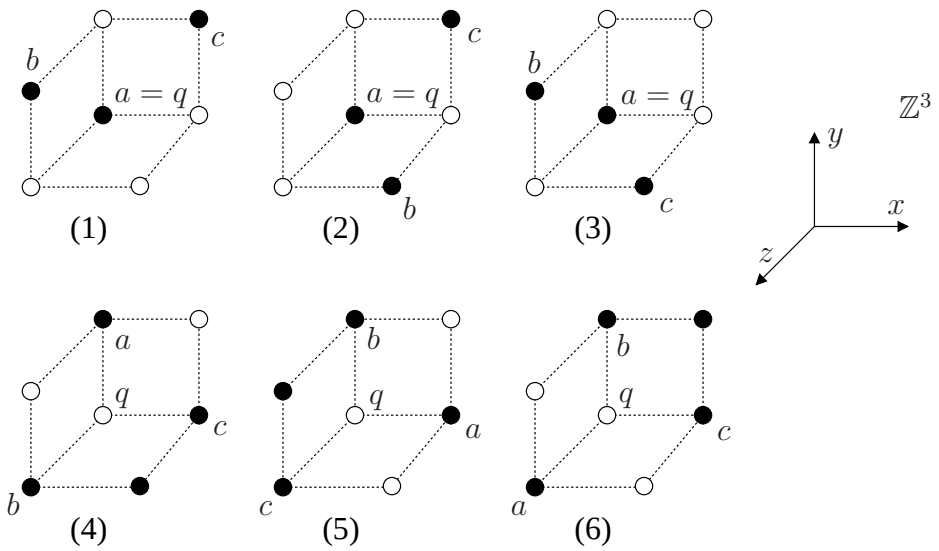

Figure 6: Illustration of all occurrences of case A2 in Table 1. The foreground and background points of $(\mathbb{G}, X^{(i-1)})$ that belong to the two instances of C1 are shown as open and solid circles, respectively.

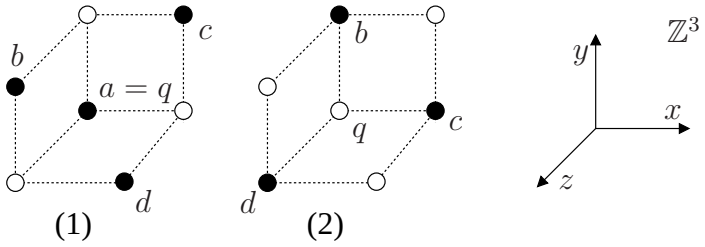

Figure 7: Illustration of the sub-cases, (1) and (2), of case A3 in Table 1. The foreground and background points of $(\mathbb{G}, X^{(i-1)})$ that belong to the two instances of C1 are shown as open and solid circles, respectively.

Finally, observations (1), (2), and (3) also imply that the loop cannot iterate more than $4 \cdot |\mathbb{G}|$ times. Since each loop iteration runs in constant time, the time complexity of the algorithm is $\mathcal{O}(|\mathbb{G}|)$. The space complexity (i.e., the amount of memory usage) is also clearly $\mathcal{O}(|\mathbb{G}|)$. It is worth to remark that we can also obtain a well-composed image from $(\mathbb{G}, X)$ by executing our algorithm on $(\mathbb{G}, \mathbb{G} - X)$. However, the resulting well-composed image is not necessarily the same as the one obtained by executing the algorithm on $(\mathbb{G}, X)$.

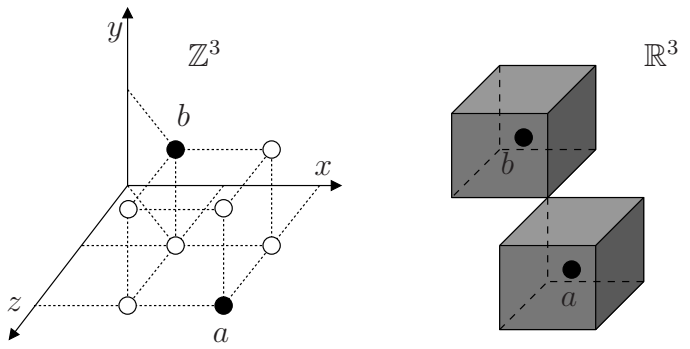

Figure 8: Illustration of the sub-case (1) of case B in Table 1. Points a and b are the background points of $(\mathbb{G}, X^{(i-1)})$ that belong to the instance of C2.

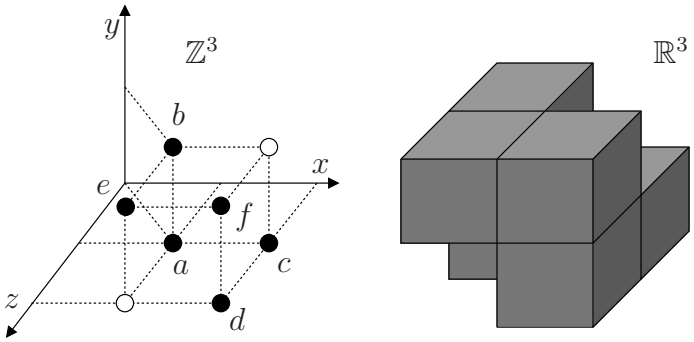

Figure 9: Illustration of the sub-case (2) of case B in Table 1. Points a, b, c, d, e, and f are the background points of $(\mathbb{G}, X^{(i-1)})$ that belong to the instance of C2.

5.2 How Effective is the Algorithm?

The randomized algorithm described in the previous section does not necessarily find a smallest set P of background points of $(\mathbb{G}, X)$ such that $(\mathbb{G}, X \cup P)$ is well-composed. Actually, the set P can in principle be the entire set X_0 of background points of $(\mathbb{G}, X)$. So, it is natural to ask ourselves how effective the algorithm really is. Here, we derive a probabilistic upper bound for the *expected* size of

the set P computed by the algorithm. The size $|P|$ of P is bounded above by $m+t$, where m is the number of critical configurations in $(\mathbb{G}, X)$ and t is the number of critical configurations created by the algorithm to generate the well-composed image $(\mathbb{G}, X')$, where $X' = X \cup P$. This is because the number $|S|$ of points of the set $S \subseteq (X_0 - P)$ computed by the algorithm (see Line (06)) is always smaller than or equal to the number of critical configurations in $\mathcal{J}(q)$. So, $|P|$ cannot exceed $m+t$. We can view m as an intrinsic feature of $(\mathbb{G}, X)$. In contrast, we can view t as an intrinsic feature of the algorithm, and hence we derive an upper bound for the expected value of t, $E[t]$, in terms of m.

Recall that before the **while** loop is entered, the queue Q contains all points $q \in \mathbb{G}$ such that $\mathcal{J}(q)$ contains at least one instance of a critical configuration of $(\mathbb{G}, X)$. Let R be the set of points of Q before the loop is entered. The size, $|R|$, of R is at most m. Let $r^{(j)} \in R$ be the point removed from Q in the j-th iteration of the loop. Let $C(r^{(j)})$ denote the number of *new* critical configurations of $(\mathbb{G}, X^{(j)})$ created by the algorithm in the j-th iteration (i.e., the number of critical configurations that are in $(\mathbb{G}, X^{(j)})$ but not in $(\mathbb{G}, X^{(j-1)})$.) Since Q is a queue, the point r will be removed from Q before the removal of any point inserted into Q in the end of the j-th iteration. Since $\mathcal{N}_x(p) \neq \mathcal{N}_x(s)$, $\mathcal{N}_y(p) \neq \mathcal{N}_y(s)$, $\mathcal{N}_z(p) \neq \mathcal{N}_z(s)$, and $\mathcal{N}(p) \neq \mathcal{N}(s)$, for any two points $p, s \in \mathbb{G}$, with $p \neq s$, the total number, t', of critical configurations created by the algorithm until the $|R|$-th iteration is then

$$t' = \sum_{j=1}^{|R|} C(r^{(j)})\,.$$

Let us consider $C(r^{(j)})$ as a random variable. Then, the expected value, $E[t']$, of t' is given by

$$E[t'] = E\big[\sum_{j=1}^{|R|} C(r^{(j)})\big] = \sum_{j=1}^{|R|} E[C(r^{(j)})]\,,$$

where $E[C(r^{(j)})]$ is the expected value of $C(r^{(j)})$. By definition of expected value, $E[C(r^{(j)})]$ is given by

$$E[C(r^{(j)})] = \sum_{i=0}^{N} i \cdot P\{C(r^{(j)}) = i\},$$

where $P\{C(r^{(j)}) = i\}$ is the probability that $C(r^{(j)})$ is equal to i, and N is the largest value of $C(r^{(j)})$. So, if we are able to compute $P\{C(r^{(j)}) = i\}$, for each $i \in \{0, \ldots, N\}$ and each $j \in \{1, \ldots, |R|\}$, we can compute $E[t']$.

Recall that the set $S = S^{(j)}$, computed by the algorithm in the j-th iteration of the loop, can only contain points from the set $\mathcal{N}(r^{(j)})$. So, if any new instance of a critical configuration is created in $(\mathbb{G}, X^{(i)})$, this instance must belong to the $4 \times 4 \times 4$ grid,

$$G(r^{(j)}) = [r_1^{(j)} - 1, r_1^{(j)} + 2] \times [r_2^{(j)} - 1, r_2^{(j)} + 2] \times [r_3^{(j)} - 1, r_3^{(j)} + 2] \subset \mathbb{Z}^3\,,$$

where $(r_1^{(j)}, r_2^{(j)}, r_3^{(j)})$ are the coordinates of $r^{(j)}$. So, to compute $P\{C(r^{(j)}) = i\}$, we can restrict our attention to $G(r^{(j)})$. In what follows, we compute $P\{C(r^{(j)}) = i\}$ by making the following three simplifying assumptions:

1) The restriction of the input image $(\mathbb{G}, X)$ to $G(r^{(j)})$ is equally likely to be any of the 2^{64} binary images in the set
$$\mathcal{H} = \{(G(r^{(j)}, W) \mid W \subseteq G(r^{(j)}\}\,.$$

2) Any choice of $B(r^{(j)})$ is equally likely to be selected by the algorithm to be the set $S^{(j)}$.

3) The random variables, $C(r^{(1)}), \ldots, C(r^{(|R|)})$, are independent.

The first assumption implies that the probability, $P\{(G(r^{(j)}) \cap X) = W\}$, that the restriction of $(\mathbb{G}, X)$ to $G(r^{(j)})$ is equal to $(G(r^{(j)}), W)$ is $1/|\mathcal{H}|$. The third assumption means that the elimination of all critical configurations of $C(r^{(j)})$ does not change the color of the points of $(\mathbb{G}, X)$ in the grids $G(r^{(j+1)}), \ldots, G(r^{(|R|)})$. In turn, this means that $P\{(G(r^{(j)}) \cap X^{(j)}) = W\} = P\{(G(r^{(j)}) \cap X) = W\} = 1/|\mathcal{H}|$.

Let $P\{S^{(j)}\}$ denote the probability that a given set S of $B(q)$ is chosen by the algorithm. Then, for any $0 \leq i \leq N$, the probability that the algorithm will create exactly i new critical configurations in $(\mathbb{G}, X^{(j)})$ is
$$P\{S^{(j)}\} \cdot I(S^{(j)}, W, i),$$
where $I(S^{(j)}, W, i)$ is an *indicator random variable* that is equal to 1 if the algorithm creates exactly i new critical configurations in $(\mathbb{G}, X^{(j)})$, and is equal to 0 if no new critical configuration is created. So,
$$P\{C(r^{(j)}) = i\} = \sum_{(G(r^{(j)}), W) \in \mathcal{H}} P\{(G(r^{(j)} \cap X^{(j)}) = W\} \cdot \Big(\sum_{S \in B(r)} P\{S^{(j)}\} \cdot I(S^{(j)}, W, i) \Big)\,.$$

By assuming that the third assumption holds, we built a computer program to calculate the values of $P\{S^{(j)}\}$ and $I(S^{(j)}, W, i)$ for all possible binary images $(G(r^{(j)}), W) \in \mathcal{H}$, all sets of choices $S \in B(r)$, and all values of i, with $0 \leq i \leq N$, where $N = 36$ is the largest value of $C(r^{(j)})$, which can also be determined by a computer program. This is because $G(r^{(j)})$ has only 2^{64} points, and we can compute the above unknowns by case enumeration. Our program computed the following values for $P\{C(r^{(j)}) = i\}$: $P\{C(r^{(j)}) = 0\} = 0.837498$, $P\{C(r^{(j)}) = 1\} = 0.0916715$, $P\{C(r^{(j)}) = 2\} = 0.0411385$, $P\{C(r^{(j)}) = 3\} = 0.0160874$, $P\{C(r^{(j)}) = 4\} = 0.00726092$, $P\{C(r^{(j)}) = 5\} = 0.0036895$, $P\{C(r^{(j)}) = 6\} = 0.00173523$, $P\{C(r^{(j)}) = 7\} = 0.00070917$, $P\{C(r^{(j)}) = 8\} = 0.000186012$, $P\{C(r^{(j)}) = 9\} = 2.32515 \times 10^{-5}$, and $P\{C(r^{(j)}) = k\} < 10^{-8}$, for all $9 < k \leq 36$. So, we get
$$E[C(r^{(j)})] = \sum_{i=0}^{N=36} i \cdot P\{C(r^{(j)}) = i\} = 0.286775.$$

Now, we can derive an upper bound for

$$E[t'] = \sum_{j=1}^{|R|} E[C(r^{(j)})]$$

in terms of m. Since $|R| \leq m$ and $E[C(r^{(j)})] = 0.286775$, we have that

$$E[t'] = \sum_{j=1}^{|R|} E[C(r^{(j)})] = 0.286775 \cdot |R| \leq 0.286775 \cdot m < \frac{m}{3}.$$

The above upper bound for $E[t']$ can be used to find an upper bound for $E[t]$, the expected number of new critical configurations created during the *entire* execution of the algorithm. In order to do that, we suppose that the assumptions 1), 2), and 3) also hold for the t newly created critical configurations.

Since our repairing algorithm uses a queue to keep track of critical configurations, we can think of the whole process of elimination of critical configurations as carried out in K steps, for some positive integer K. More specifically, the first step consists of the elimination of *all* critical configurations in the input image, $(\mathbb{G}, X)$. The second step consists of the elimination of *all* critical configurations created as a result of eliminating the critical configurations of the first step, and so on. For each $h \in \{1, \ldots, K\}$, let m_h denote the total number of critical configurations created in the h-th step. Note that $m_h = 0$ for $h = K$, as the resulting image (i.e., $(\mathbb{G}, X')$) has no critical configurations. We already know that $E[m_1] < m/3$. So, let $\{1, \ldots, M_h\}$ be all possible values of m_h, and assume that each m_h, for $2 \leq h < K$, takes any value from $\{1, \ldots, M_h\}$ with equal probability. Then, for any integer i, with $2 \leq i < K$,

$$E[m_i] = \sum_{l=1}^{M_{i-1}} P\{m_{i-1} = l\} \Big(\sum_{q \in Q_l} E[C(q)] \Big)$$

where Q_l consists of all points q in Q that are removed from Q during the $i-1$ step, and $\mathcal{J}(q)$ contains a critical configuration in the image produced by the **while** loop iteration that precedes the iteration in which q is removed from Q. Since m_{i-1} is any of $1, \ldots, M_{i-1}$ with equal probability, we have that

$$P\{m_{i-1} = l\} = \frac{1}{|\{1, \ldots, M_{i-1}\}|} = \frac{1}{M_{i-1}},$$

where $|\{1, \ldots, M_{i-1}\}| = M_{i-1}$ is the size of $\{1, \ldots, M_{i-1}\}$. So, the expected value $E[m_i]$ of m_i is given by

$$E[m_i] = \frac{1}{M_{i-1}} \sum_{l=0}^{M_{i-1}} \Big(\sum_{q \in Q_l} E[C(q)] \Big).$$

Using the argument for deriving $E[m_1]$ again, we get

$$\sum_{q \in Q_l} E[C(q)] = 0.286775 \cdot |Q_l| \,,$$

where $|Q_l|$ is the size of Q_l. Thus,

$$E[C(q)] \leq 0.286775 \cdot l < \frac{l}{3} \quad \text{and} \quad \mathrm{E}[m_i] < \frac{1}{3} \cdot \frac{\sum_{l \in \{1,\ldots,M_{i-1}\}} l}{M_{i-1}}.$$

But,

$$\frac{\sum_{l \in \{1,\ldots,M_{i-1}\}} l}{M_{i-1}}$$

is precisely the expected value of m_{i-1}. So, $E[m_i] < \frac{1}{3} \cdot E[m_{i-1}]$, and

$$E[t] = E\Big[\sum_{k=1}^{K} m_k\Big] = E\Big[\sum_{k=1}^{K-1} m_k\Big] = \sum_{k=1}^{K-1} E[m_k] < \sum_{k=1}^{K-1} \Big(\frac{1}{3}\Big)^k \cdot m < \sum_{k=1}^{\infty} \Big(\frac{1}{3}\Big)^k \cdot m = \frac{m}{2}.$$

So, we have the following theorem:

Theorem 5.1. *Let $(\mathbb{G}, X)$ be a 3D binary image with m critical configurations. If assumptions 1), 2), and 3) hold, then the expected value $E[t]$ of the number t of critical configurations created by our algorithm on input $(\mathbb{G}, X)$ is less than $\frac{m}{2}$.*

At first glance, the third assumption may not seem reasonable, but there are two reasons for making this assumption. First, it makes it easier to compute a probability distribution for $P\{C(r^{(j)}) = i\}$. Second, as j gets larger, the probability that $G(r^{(j)}) \cap X^{(j)}$ has more foreground than background points increases. So, $P\{C(r^{(j)}) = i\}$ will be larger for smaller values of i. Since we are interested in an upper bound for $E[t']$, assumption 3) is reasonable for the purpose of obtaining this upper bound. Finally, since the size, $|P|$, of the set P obtained by the repairing algorithm is bounded above by $m + t$, Theorem 5.1 implies that the expected size $E[|P|]$ of P is bounded above by $\frac{3m}{2}$.

5.3 Topological Repairing and Morphological Operators

It is tempting to think that one can repair a 3D digital binary image, which is not well-composed, using a combination of dilations and erosions. Unfortunately, such a morphological operation may not work, as a new critical configuration may always arise as the result of the elimination of another one. For the sake of clarity, we illustrate this fact using a *2D digital binary image* $(\mathbb{G}, X)$ shown in Figure 10(a), where $\mathbb{G} = [1, 7] \times [1, 7]$. Figure 10(b) shows the image resulting from a dilation of the image in Figure 10(b) using a 2×2 neighborhood as the structuring element, and Figure 10(c)

shows the image resulting from an erosion of the image in Figure 10(a) using the same structuring element.

Although the only instance of C1 in the image in Figure 10(a) has been removed from the dilated image in Figure 10(b), another instance of C1 has been created. Similarly, the only instance of C1 in the image in Figure 10(b) has been eliminated from the eroded image in Figure 10(c), but another one has been created. Note that we can obtain the image in Figure 10(b) by dilating the image in Figure 10(c). Note also that the images in Figure 10(a) and 10(c) are the same, which means that the image in Figure 10(a) is invariant with respect to the composition of dilation followed by erosion.

The counterexample just presented shows that the composition of dilation followed by erosion cannot repair any given 2D digital binary images. A similar counterexample can be built to show that the composition of erosion followed by dilation cannot repair any given 2D digital binary images either. Finally, analogous examples can be given for a 3D digital binary images, which can be thought of as consisting of three "stacked" copies of the 2D digital binary image shown in Figure 10(a).

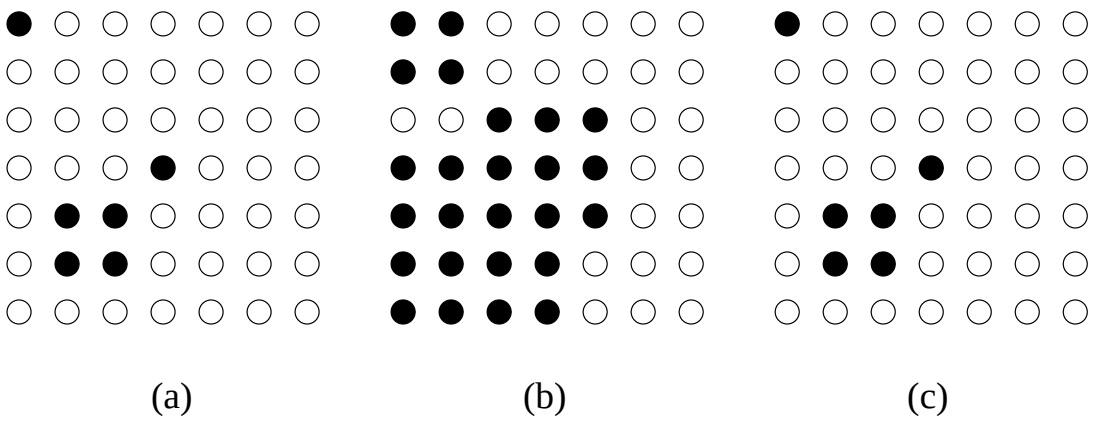

Figure 10: (a) A 2D digital binary image $(\mathbb{G}, X)$, where $\mathbb{G} = [1, 7] \times [1, 7] \subseteq \mathbb{Z}^2$. (b) The 2D digital binary image resulting from a dilation of the image in (a). (c) The 2D digital binary image resulting from an erosion of the image in (b). Foreground and background points are represented by open circles and solid circles, respectively.

6 An Algorithm for Repairing 3D Multivalued Images

This section describes an extension of the algorithm in Section 5 to repairing 3D digital multivalued images, which are assumed to be the result of multi-object segmentations. More specifically, the input for the algorithm is the grid $\mathbb{G}$ of a 3D digital multivalued image $\mathcal{F} : \mathbb{G} \to \{c_1, \ldots, c_k\}$, for

some integer k, with $k > 2$, and the k sets $X_1, \ldots, X_k$, where $X_i = \{p \in \mathbb{G} \mid \mathcal{F}(p) = c_i\}$, for every $i \in \{1, \ldots, k\}$. The output consists of k sets, $X'_1, \ldots, X'_k$, where $X'_i = \{p \in \mathbb{G} \mid \mathcal{F}'(p) = c_i\}$ and $\mathcal{F}' : \mathbb{G} \to \{c_1, \ldots, c_k\}$ is a *3D multivalued well-composed image*, i.e., each binary image $(\mathbb{G}, X'_i)$ is a well-composed image.

The motivation for extending our repairing algorithm to 3D digital multivalued images is that such images may represent the segmentation of other 3D digital multivalued images into more than just background and foreground. This is particularly the case if *fuzzy connectedness* is employed as the segmentation paradigm [37, 17]. So, it is natural to think of a repairing approach that modifies the segmented multivalued image as a whole, so that the repaired image could enjoy the same benefits as well-composed binary images. Unfortunately, the application of our repairing algorithm in Section 5 to each binary image $(\mathbb{G}, X_1), \ldots, (\mathbb{G}, X_m)$ does not necessarily yield well-composed images, $(\mathbb{G}, X'_1), \ldots, (\mathbb{G}, X'_m)$. The reason is that the repairing of one binary image $(\mathbb{G}, X_i)$ may affect a previously repaired image $(\mathbb{G}, X_j)$, for some $i, j \in \{1, \ldots, m\}$, with $i \neq j$. Furthermore, the *repeated* application of the algorithm in Section 5 to repair the previously repaired images may never terminate.

The idea behind our extended algorithm is to repeatedly apply a modified version of the algorithm in Section 5 to the images $(\mathbb{G}, X_1), \ldots, (\mathbb{G}, X_k)$, in this order. Every time the algorithm repairs the image $(\mathbb{G}, X_i)$, for any $i \in \{1, \ldots, k\}$, it can modify a previously repaired image, $(\mathbb{G}, X_j)$, for some $j \in \{i+1, \ldots, k\}$. When $(\mathbb{G}, X_1), \ldots, (\mathbb{G}, X_k)$ are *all* well-composed images after the repairing of any of them (which eventually happens), the algorithm lets $X'_i = X_i$, for each $i \in \{1, \ldots, k\}$, and terminates.

While repairing $(\mathbb{G}, X_i)$, for each $i \in \{1, \ldots, k\}$, the algorithm may insert or remove points from X_i. Each point inserted into X_i is previously removed from a set X_j, where $j \in \{i + 1, \ldots, k\}$. Likewise, each point removed from X_i is later inserted into a set X_j, where $j \in \{1, \ldots, i - 1\}$. We can view these insertion/removal operations as changing the color assigned to a point from c_j to c_i (or from c_i to c_j). This means that the resulting well-composed image depends on the order of its colors in the sequence $c_1, \ldots, c_k$. In particular, the components consisting of points assigned the color c_1 can only have their sizes increased, while the components consisting of points assigned the color c_k can only have their sizes decreased. So, before using the algorithm, the color sequence $c_1, \ldots, c_k$ should be defined according to some rank of importance of the color components of the input image.

The repairing of each $(\mathbb{G}, X_i)$ is carried out in two steps. The first step finds all critical configurations of $(\mathbb{G}, X_i)$. More specifically, for each point $q \in \mathbb{G}$, if $\mathcal{J}(q)$ contains a critical configuration of $(\mathbb{G}, X_i)$ then q is inserted into a queue Q_i, which is initially empty. So, $(\mathbb{G}, X_i)$ is well-composed if, and only if, Q_i is empty when the first step is over. If at least one of Q_i is not empty when the first step is over, the second step is carried out. Otherwise, the algorithm terminates with $X'_i = X_i$, for all $i \in \{1, \ldots, k\}$. The second step is a slight modification of the algorithm in Section 5. The goal now is to compute a subset X'_i of $\mathbb{G}$, for each $i \in \{1, \ldots, k\}$, such that the binary image $(\mathbb{G}, X'_i)$ is well-composed. Each set X'_i is made equal to X_i before the second step starts. To compute each

X'_i, the algorithm repeatedly considers the queues $Q_1, \ldots, Q_k$, in this order. For each Q_i that is not empty, the algorithm iteratively removes one point q at a time from Q_i, modifies Q_i and some of $X_1, \ldots, X_k$, and may eventually insert one or more points from $\mathbb{G}$ into any of the sets $Q_1, \ldots, Q_k$.

Let Q_i be the nonempty queue currently considered by the algorithm, for some $i \in \{1, \ldots, k\}$, and let q be the point currently removed from Q_i. If $\mathcal{J}(q)$ contains a critical configuration of $(\mathbb{G}, X'_i)$, then the algorithm computes the set $B(q)$ in the same way the repairing algorithm for binary image does. Next, the algorithm computes a subset of *valid choices* of $B(q)$, denoted by $B'(q)$, and defined as

$$B'(q) = \{S \in B(q) \mid (S \cap X'_j) = \emptyset, \text{ for all } j \text{ with } 1 \leq j < i\}.$$

The set $B'(q)$ contains only the members S of $B(q)$ whose points are not foreground points of the images $(\mathbb{G}, X'_j)$, for all $j \in \{1, \ldots, i-1\}$. So, if the algorithm chooses a set from $B'(q)$ to eliminate the critical configurations of $\mathcal{J}(q)$, it will not change the color of a point from c_j to c_i with $j < i$. Consequently, the images $(\mathbb{G}, X'_j)$, for all $j \in \{1, \ldots, i-1\}$, will not be modified by the change of colors.

Unlike the set $B(q)$, the set $B'(q)$ may be empty. This is the case if, and only if, every set S of $B(q)$ contains a point that belongs to a set X'_j, with $j < i$. Whenever $B'(q)$ is not empty, the algorithm chooses a set S from $B'(q)$ and lets $X'_i = X'_i \cup S$; that is, every point of S has its color changed to c_i. Next, the algorithm removes the points of S from the corresponding sets X'_h, for $h \in \{i+1, \ldots, k\}$. The choice of S is made by the same rule of choice used by the repairing algorithm for binary images. If $B'(q)$ contains only one set, S is this set. If $B'(q)$ has two or more elements, the algorithms picks S such that $(\mathbb{G}, X'_i \cup S)$ does not contain any critical configuration that is not also in $(\mathbb{G}, X'_i)$. If there is a tie or if there is no such set, the algorithm picks S at random.

If $B'(q)$ is empty then there is no point whose color can be changed from c_j to c_i, with $j > i$, in order to eliminate the critical configurations of $\mathcal{J}(q)$ in $(\mathbb{G}, X'_i)$. So, we are left with the option of changing the color assigned with points of X'_i from c_i to some c_j, with $j < i$; that is, we remove points from X'_i, or equivalently, insert points into $\overline{X'_i}$. To do that, the algorithm computes $B(q)$ for the image $(\mathbb{G}, \overline{X'_i})$. Since $\mathbb{G}$ is a rectangular grid, any critical configuration $(\mathbb{G}, \overline{X'_i})$ is also a critical configuration of $(\mathbb{G}, X'_i)$, and vice-versa. Furthermore, every point of any set $S \in B(q)$ is a point of $(\mathbb{G}, X'_i)$. So, the algorithm can use the same *rule of choice* used by the repairing algorithm in Section 5. Next, the algorithm lets $X'_i = X'_i - S$ and, for each critical configuration A of $(\mathbb{G}, X'_i)$ in $\mathcal{J}(q)$, the algorithm lets $X'_j = X'_j \cup S$, where j is the largest integer in $\{1, \ldots, i-1\}$ such that $(X'_j \cap A) \neq \emptyset$.

The motivation for choosing j as the largest integer in $\{1, \ldots, i-1\}$ such that $(X'_j \cap A) \neq \emptyset$, for each critical configuration A of $(\mathbb{G}, X'_i)$ in the set $\mathcal{J}(q)$, is two-fold. First, we avoid creating "isolated" one-voxel, same-colored components. Second, we replace c_i with a color c_j with the least degree of "importance" that is larger than the one of c_i. This criterion is based on the assumption that c_j is more similar to c_i than to any other color c_h, with $h < j$. However, we can easily

incorporate other criteria for choosing c_j into the algorithm. For instance, we could choose c_j to be the color assigned to the majority of the points of A whose color index is smaller than i. However, whatever criterion is chosen, the resulting well-composed image will depend on the ordering of the colors determined by the criterion.

After modifying X'_i and X'_j, the algorithm finds all points $p \in \mathbb{G}$ such that $\mathcal{J}(p)$ contains a critical configuration of $(\mathbb{G}, X'_i)$ (resp. $(\mathbb{G}, X'_j)$), which did not exist before a point of S is inserted or removed from X'_i (resp. X'_j). Next, the algorithm inserts p into Q_i (resp. Q_j). Like the repairing algorithm for binary images, the search for p is limited to the grid $[p_1-1, p_1+2] \times [p_2-1, p_2+2] \times [p_3-1, p_3+2]$, where (p_1, p_2, p_3) are the coordinates of p. So, each p can be found in constant time. Whenever Q_i becomes empty, $(\mathbb{G}, X'_i)$ is well-composed, and the nonempty queue from $\{Q_1, \ldots, Q_k\}$ with the smallest index is considered. In what follows, we present a pseudo code for the algorithm just described:

(01) $X'_1, \ldots, X'_k \leftarrow \emptyset$
(02) **while** any of $Q_1, \ldots, Q_k$ is not empty **do**
(03) let i be the smallest integer in $\{1, \ldots, k\}$ such that Q_i is not empty
(04) **while** Q_i is not empty **do**
(05) remove a point q from Q_i
(06) **if** any $\mathcal{J}(q)$ contains a critical configuration **then**
(07) compute $B'(q)$ for $(\mathbb{G}, X'_i)$
(08) **if** $B'(q) \neq \emptyset$ **then**
(09) choose S from $B'(q)$ according to the *rule of choice*
(10) let $X'_i \leftarrow X'_i \cup S$
(11) let $X'_j \leftarrow X'_j - S$ for all $j \in \{i+1, \ldots, k\}$
(12) for each $j \in \{i, \ldots, k\}$, for all $p \in \mathbb{G}$, find all *new* critical configurations of $(\mathbb{G}, X'_j)$ in $\mathcal{J}(p)$, and insert p into Q_j
(13) **else**
(14) compute $B(q)$ for $(\mathbb{G}, \overline{X'_i})$
(15) choose S from $B(q)$ according to the *rule of choice*
(16) let $X'_i \leftarrow X'_i - S$
(17) for each $A \in \mathcal{J}(q)$, find the largest $l \in \{1, \ldots, i-1\}$ such that $(X'_l \cap A) \neq \emptyset$
(18) let $X'_l \leftarrow X'_l \cup S$
(19) find all *new* critical configurations $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, or $\mathcal{N}(p)$ in $(\mathbb{G}, X'_j)$, for $j = l$ and $j = i$ and $p \in \mathbb{G}$, and insert p into Q_j
(19) **endif**
(20) **endif**
(21) **endwhile**
(22) **endwhile**
(23) **return** $X'_1, \ldots, X'_k$

The algorithm above always terminates. This is obviously true if the input image $\mathcal{F}$ is already well-composed. So, assume otherwise. The outer **while** loop (Lines (01)-(22)) is entered whenever one of $Q_1, \ldots, Q_k$ is not empty. In each iteration of this loop, the nonempty Q_i with the smallest $i \in \{1, \ldots, k\}$ is selected, and the inner **while** loop on (Lines (04)-(21)) iterates until Q_i becomes empty. In each iteration of the inner loop, the algorithm removes a point from Q_i and may eventually insert one or more points in Q_i. Since each point p inserted into Q_i is a point from $\mathbb{G}$ and $\mathbb{G}$ is finite, the algorithm cannot keep inserting points into Q_i indefinitely if each point p is inserted into Q_i finitely many times only. The outer **while** loop is executed whenever one of $Q_1, \ldots, Q_k$ is not empty. In each iteration of this loop, the nonempty Q_i with smallest $i \in \{1, \ldots, k\}$ is selected, and the inner **while** loop on line (03) iterates until Q_i becomes empty. In each iteration of the inner loop, the algorithm removes a point q from Q_i and may eventually insert one or more points in Q_i. Each point inserted into Q_i is a point from $\mathbb{G}$.

A point p is inserted into Q_i if, and only if, $\mathcal{J}(p)$ contains a critical configuration of $(\mathbb{G}, X_i')$. Furthermore, once a critical configuration is eliminated from $(\mathbb{G}, X_i')$ by inserting or removing points from X_i', it can no longer occur in $(\mathbb{G}, X_i')$. The reason is that the following *insertion invariant* holds: every point inserted into X_i' has been removed from a set X_j', with $j > i$, or dually, every point removed from X_i' is inserted into a set X_j', with $j < i$. So, each $p \in \mathbb{G}$ can be inserted into Q_i at most four times, each of which corresponds to one of $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$, and $\mathcal{N}(p)$ becoming a critical configuration. Since $\mathbb{G}$ is finite and every iteration of the inner loop removes a point from Q_i, the queue Q_i eventually becomes empty just before the end of the n-th iteration of the loop, for some finite positive integer n. However, Q_i may become nonempty again as a result of the repairing of $(\mathbb{G}, X_j')$, for some $j \in \{1, \ldots, m\}$, with $j \neq i$, which will force the outer **while** loop to be entered again. However, since each $p \in \mathbb{G}$ can be inserted into Q_i at most four times, Q_i cannot become nonempty indefinitely, which means that the outer **while** loop will eventually terminate with all the queues empty.

The time complexity of the algorithm is $\mathcal{O}(k \cdot |\mathbb{G}|)$. This follows from the fact that a given point $p \in \mathbb{G}$ belongs to exactly one set X_i' at any given time during the execution of the algorithm, and p can be removed from a set X_i' and inserted into a set X_j' at most k times, for any $i, j \in \{1, \ldots, k\}$, with $i \neq j$. So, the combined number of iterations of the outer and inner loops is $\mathcal{O}(k \cdot |\mathbb{G}|)$. However, the space complexity (memory usage) of the algorithm is still linear in $|\mathbb{G}|$. Although the extended repairing algorithm was devised for multivalued images, it can be applied to binary images as well. Given a 3D binary digital image $(\mathbb{G}, X)$, if we run the algorithm with $X_1 = X$ and $X_2 = (\mathbb{G} - X)$, the output will be the same well-composed image obtained by the repairing algorithm in Section 5 when given $(\mathbb{G}, X)$.

7 Experimental Results

This section describes and discusses some results obtained by running the algorithm in Section 5 and its extension in Section 6 on several 3D digital binary and multivalued images, respectively. To produce the results, we used the open source and freely available implementations of our repairing algorithms in the National Library of Medicine (NLM) Insight Segmentation and Registration Toolkit (ITK)[2]. This toolkit also has an implementation of the repairing algorithm in [16] for 2D binary images (see [38]).

For the purpose of describing the aforementioned results, we divided the images into three groups as follows:

G1. This group contains only one image: a 3D digital multivalued image corresponding to a multi-object segmented normal brain image produced by the *brain web* MR image simulator [39]. Each point of this image is assigned one of 10 colors from the set $\{0, 1, \ldots, 9\}$.

G2. This group contains three 3D digital binary images, each of which was obtained from the multivalued image in G1. In particular, the foreground of the first binary image is the set of points of the multivalued image in G1 that are assigned color 1 (CSF segmentation); the foreground of the second binary image is the set of points of the multivalued image in G1 that are assigned color 2 (gray matter segmentation); and the foreground of the third binary image is the set of points of the multivalued image in G1 that are assigned color 3 (white matter segmentation).

G3. For each of the three binary images in G2, we created four binary images with varying levels of noise as follows: For each image in G2 and each $n \in \{1, \ldots, 4\}$, we created another binary image by adding Gaussian noise with zero mean and standard deviation equal to $\frac{(n+1)}{10}$, and then thresholded the resulting image such that voxel values less than or equal to 0.5 were reassigned a value of 0 and 1 otherwise. Figure 11(a)-(d) shows an axial slice of the four binary, white matter segmented images in G3 obtained by adding noise and thresholding, as described before.

Table 7 shows the total number of instances of C1 and C2 in each binary image from the groups G2 and G3. By examining this table, we see that the total number of instances of C1 and C2 in some images from group G3 can be as large as 15% of the number of points in the image grid, while the total number of instances of C1 and C2 in any image from group G2 is no larger than 0.24% of the number of points in the image grid. We ran the ITK implementation of our repairing algorithms in Section 5 on each binary image from G2 and G3, ten times per image. For each execution, we computed the minimum, maximum, and average values of the point-wise color difference and Hausdorff distances between the input and output images. We also computed the standard deviation of the average values. The results are shown in Table 3 and Table 4.

[2] `http://www.itk.org`.

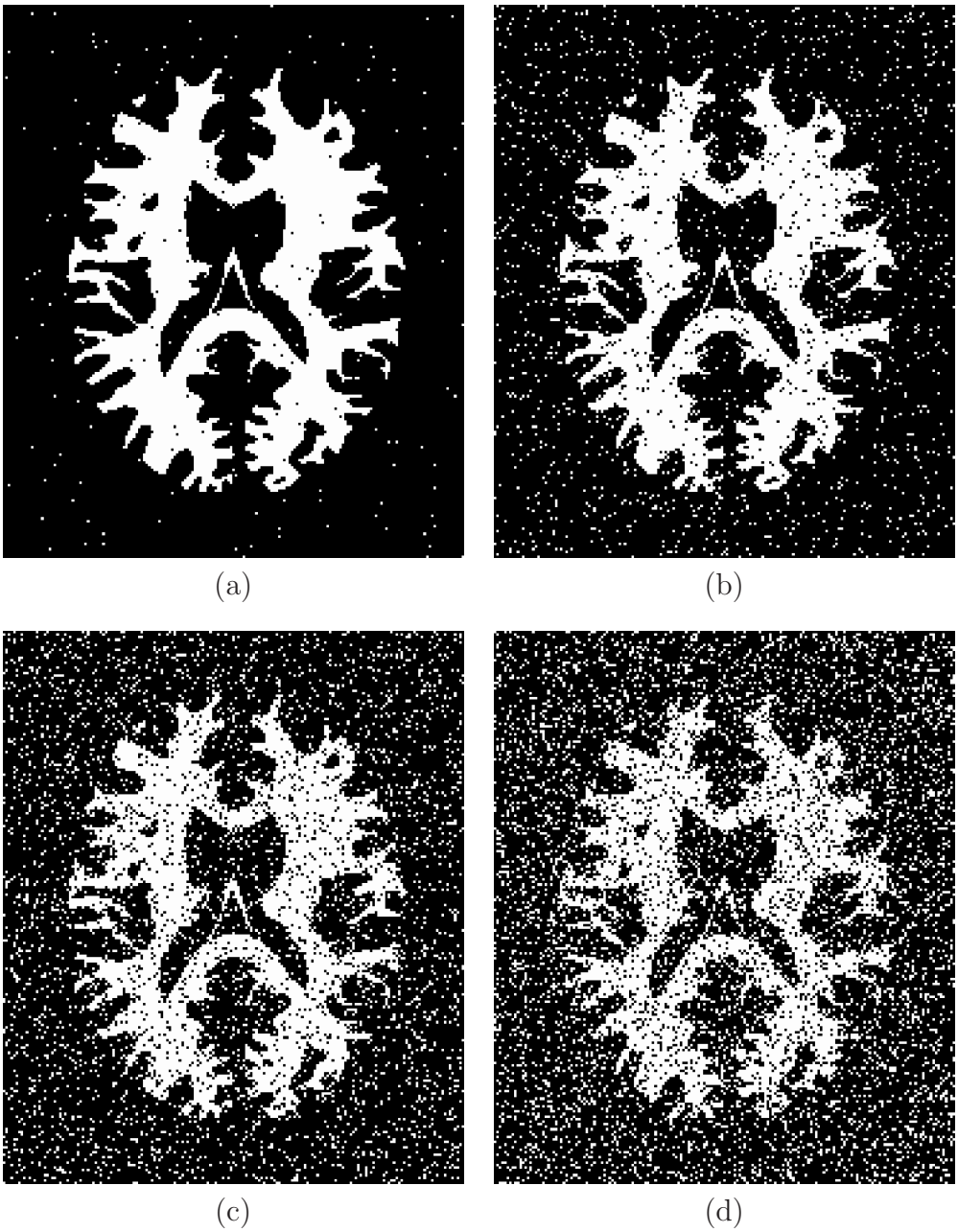

Figure 11: Axial slices of the binary, white matter segmented images in group G3, which were obtained by adding noise with zero mean and standard deviation (a) 0.2, (b) 0.3, (c) 0.4, and (d) 0.5.

If $(\mathbb{G}, X)$ and $(\mathbb{G}, X')$ are the input and output images, respectively, then the *point-wise color difference* (PWCD) is equal to

$$\mathrm{PWCD}(X, X') = \sum_{p \in \mathbb{G}} d(p)$$

where $d(p) = 0$ if $p \in ((X \cap X') \cup ((\mathbb{G} - X) \cap (\mathbb{G} - X')))$, and $d(p) = 1$ otherwise, and the *Hausdorff distance* (HD) is equal to

$$\mathrm{HD}(X, X') = \max(h(X, X'), h(X', X)),$$

where

$$h(X, X') = \max_{p \in X} \min_{q \in X'} \|p - q\|,$$

and $\|p-q\|$ is the Euclidean distance between p and q. Intuitively, if the Hausdorff distance between $(\mathbb{G}, X)$ and $(\mathbb{G}, X')$ is $\delta \in \mathbb{R}$, then every point of X must be within a distance δ of some point in X', and vice-versa.

Image	**Color or Noise**	**#C1**	**#C2**	**#C1 + #C2**	**# Fore.**	**# Points**
G2, 1	1	12,939	3,883	17,778	371,945	7,109,137
G2, 2	2	10,708	2,831	13,539	902,912	7,109,137
G2, 3	3	3,038	633	3,671	674,777	7,109,137
G3, 1	0.2	17,709	5,549	23,258	411,345	7,109,137
G3, 1	0.3	120,219	52,674	172,893	677,089	7,109,137
G3, 1	0.4	418,152	159,883	578,035	1,043,984	7,109,137
G3, 1	0.5	792,005	245,073	1,037,078	1,380,349	7,109,137
G3, 2	0.2	17,467	4,946	22,413	935,915	7,109,137
G3, 2	0.3	129,109	53,136	182,245	1,157,080	7,109,137
G3, 2	0.4	437,460	158,859	596,319	1,463,816	7,109,137
G3, 2	0.5	811,702	242,952	1,054,654	1,743,108	7,109,137
G3, 3	0.2	7,485	2,350	9,835	710,587	7,109,137
G3, 3	0.3	108,457	50,470	158,927	950,657	7,109,137
G3, 3	0.4	406,008	158,954	564,962	1,283,543	7,109,137
G3, 3	0.5	784,443	245,995	1,030,438	1,589,396	7,109,137

Table 2: Color value (resp. standard deviation of Gaussian noise) defining the image foreground in group G2 (resp. G3) (second column), total number of instances of C1 and C2 (third, fourth and fifth columns), and number of foreground points and grid points (sixth and seventh columns) of the binary images in the groups G2 and G3.

From Theorem 5.1, we know that the average point-wise color difference is expected to be no larger than one and a half the number of instances of C1 and C2 in the input image. From Table 3 and Table 2, we can see that the average point-wise color difference for the binary images in G2 and G3 is actually smaller than the number of critical configurations of the input image for most

of the images, and it never exceeds the upper bound in Theorem 5.1. Table 3 also shows that the average PWCD is smaller than 5% and 0.25% of the number of foreground points and grid points for the images in group G2. In contrast, the average PWCD can be as large as 91% and 18% of the number of foreground points and grid points, respectively, for the G3 images with high level of noise. This is particularly the case for the CSF segmented, binary image with high level of noise, G3, 1.

Image	**Color or Noise**	**Average**	**Min.**	**Max.**	**St. Dev.**	**% F. P.**	**% G. P.**
G2, 1	1	17,778.8	17,738	17,856	35.6	4.78	0.25
G2, 2	2	14,468.3	14,364	14,566	56.4	1.60	0.20
G2, 3	3	3,810.5	3,782	3,840	20.2	0.56	0.05
G3, 1	0.2	24,830.1	24,758	24,888	50.7	6.04	0.35
G3, 1	0.3	214,796.0	213,983	215,279	428.2	31.78	3.02
G3, 1	0.4	732,630.3	731,270	733,473	694.5	70.18	10.31
G3, 1	0.5	1,249,342.1	1,246,763	1,252,529	1,695.7	90.51	17.57
G3, 2	0.2	23,230.3	23,144	23,278	40.4	2.48	0.33
G3, 2	0.3	211,953.9	211,412	212,412	353.4	18.32	2.98
G3, 2	0.4	709,239.8	708,294	709,996	591.2	48.45	9.98
G3, 2	0.5	1,200,595.6	1,198,866	1,203,269	1,324.8	68.88	16.89
G3, 3	0.2	10,600.2	10,516	10,629	30.9	1.49	0.15
G3, 3	0.3	196,828.6	196,349	197,135	266.5	20.70	2.77
G3, 3	0.4	703,754.8	702,654	704,725	701.0	54.83	9.90
G3, 3	0.5	1,214,396.8	1,212,939	1,216,030	1,025.7	76.41	17.08

Table 3: Color value (resp. standard deviation of Gaussian noise) defining the image foreground in group G2 (resp. G3) (second column), average PWCD (third column), minimum PWCD (fourth column), maximum PWCD (fifth column), and standard deviation (sixth column) for the images in Table 2. The seventh and the eighth columns show the average PWCD divided by number of foreground points and by the number of grid points, respectively, multiplied by 100.

From Table 4, we can see that the maximum Hausdorff distance between any image in G3 and its well-composed counterpart is no larger than $\sqrt{3}$. Since the grid spacing of all images used in our experiments is 1, we have that every point added to the foreground of the output image must be 26-adjacent to some foreground point of the input image. Furthermore, it is highly unlikely that the algorithm introduced a "hole" in the surface of the well-composed images. For the images in group G3, the maximum Hausdorff distance may reach 2.24. So, in this case, the repairing algorithm is likely to introduce undesirable topological artifacts, such as small cavities or holes" in the well-composed image. However, the distance value is still small, and it basically does not scale up with the level of noise. This is an indication that the addition of a large amount of noise to an image does not force the repairing algorithm to relabel many points of the image in order to generate a well-composed one.

Although the CSF segmented image (image G2, 1 in Table 2) has about half the number of

foreground points as the white matter segmented image (image G2, 3 in Table 2), the former image has about five times more critical configurations than the latter image. Consequently, the ratio between the average PWCD and the number of foreground points is about five times larger for the CSF segmented image (see Table 3). The large number of critical configurations of the CSF segmented image (compared to its number of foreground points) is due to the fact that the cerebrospinal fluid has very limited thickness, and therefore its topology can often be incorrectly captured by the digitization process. This also explains why the CSF segmented images with varying levels of noise have the larger statistical measures for the PWCD and HD in Table 3 and Table 4, respectively.

Image	**Color or Noise**	**Average**	**Minimum**	**Maximum**	**St. Dev.**
G2, 1	1	1.45	1.41	1.73	0.10
G2, 2	2	1.45	1.41	1.73	0.10
G2, 3	3	1.41	1.41	1.41	0.00
G3, 1	0.2	1.57	1.41	1.73	0.17
G3, 1	0.3	2.17	2.00	2.24	0.11
G3, 1	0.4	2.17	2.00	2.24	0.11
G3, 1	0.5	2.24	2.24	2.24	0.00
G3, 2	0.2	1.65	1.41	2.00	0.27
G3, 2	0.3	2.09	2.00	2.24	0.12
G3, 2	0.4	2.24	2.24	2.24	0.00
G3, 2	0.5	2.24	2.24	2.24	0.00
G3, 3	0.2	1.54	1.41	1.73	0.16
G3, 3	0.3	2.12	2.00	2.24	0.12
G3, 3	0.4	2.24	2.24	2.24	0.00
G3, 3	0.5	2.24	2.24	2.24	0.00

Table 4: The average Hausdorff distance (HD) between the images in Table 2 and their well-composed counterparts.

Figure 12 shows the surface of the continuous analog of the foreground of the well-composed image obtained from image G3, 3. The faces of the voxels corresponding to points whose assigned colors were changed by the repairing algorithm are shown in yellow. Note that the "yellow" voxels are grouped together in small neighborhoods, which are scattered throughout the surface. This tells us that the elimination of any given critical configuration from the input image did not give rise to a long sequence of change of color of neighboring points. This observation supports the adoption of assumption 3) in the probabilistic analysis of our repairing algorithm for binary images (see Section 5.2).

Table 5 shows the average frequency of cases A1, A2, A3, B(1), and B(2) of the rule of choice of the repairing algorithm (see Section 5) when given the images in groups G2 and G3. By examining this table, we see that the number of times that A1 is applied is larger than the total number of times that the remaining cases are applied. Furthermore, we verified that the average number of

instances of C1 and C2 eliminated and created by each application of case A1 is typically about 1.15 and 0.21, 1.23 and 0.18, and 1.12 and 0.12 for the CSF, gray matter, and white matter segmented, binary images in G2, respectively. Since each application of case A1 changes the color of only one point, we can say that our algorithm either generates a well-composed image that is very close to the "optimal" one (i.e, that minimizes the PWCD) or it is very conservative regarding the number of critical configurations eliminated per change of color. In Section 8, we discuss simple heuristics to increase (resp. decrease) the number of critical configurations eliminated (resp. created) per change of color.

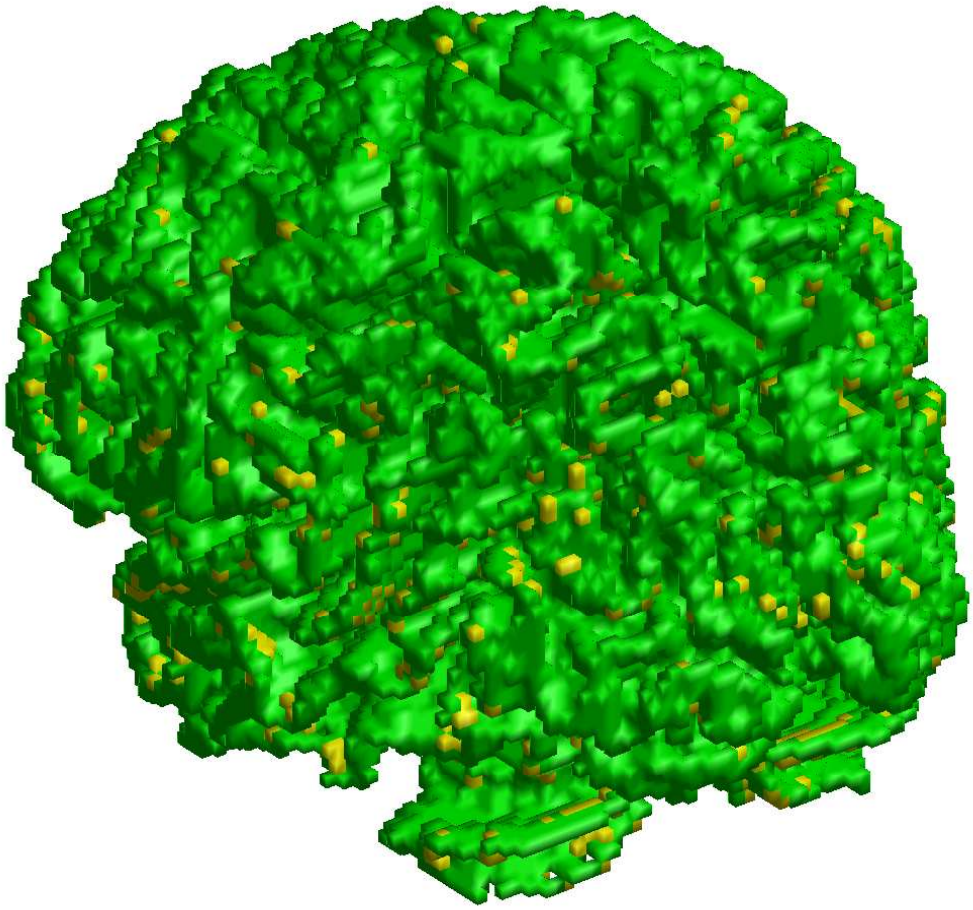

Figure 12: Continuous analog of the digital boundary between foreground and background of the well-composed image generated by our repairing algorithm for the white matter segmentation of the normal brain image in group B1. The faces of the voxels corresponding to points whose colors have been changed by the algorithm are shown in yellow.

Table 5 also shows that the frequency of case B(2) increases as the level of noise increases. This

is because an increase in the level of noise introduces several foreground points scattered throughout the background of the image (see Figure 11), which in turn increases the likelihood that each $2\times2\times2$ neighborhood, $\mathcal{J}(p)$, of a grid point, p, contains an instance of C2 with two foreground points and six background points.

Image	**A1** (%)	**A2** (%)	**A3** (%)	**B(1)** (%)	**B(2)** (%)
G2, 1	61.67	4.16	0.39	13.21	3.09
G2, 2	68.19	4.78	0.64	17.97	8.42
G2, 3	70.70	4.22	0.56	20.53	3.99
G3, 1	64.80	3.87	0.47	22.80	8.37
G3, 1	69.48	3.94	0.68	3.96	23.15
G3, 1	69.11	4.03	1.10	4.17	21.59
G3, 1	69.04	5.18	1.44	5.90	18.44
G3, 2	69.61	4.51	0.62	14.04	11.22
G3, 2	69.83	3.04	0.74	4.76	21.62
G3, 2	69.22	4.14	1.13	5.14	20.36
G3, 2	69.03	5.29	1.46	6.68	17.53
G3, 3	71.05	3.56	0.49	10.77	14.13
G3, 3	69.80	2.51	0.66	3.27	23.76
G3, 3	69.03	3.94	1.12	4.46	21.46
G3, 3	68.94	5.17	1.45	6.28	18.15

Table 5: Frequency of occurrence of the cases of the rule of choice of the algorithm in Section 5 when given the images in G2 and G3.

To evaluate the effect of our repairing algorithm on the global topology of the input image, we compared the continuous analog of the digital surface between the foreground and background of the input image with the continuous analog of the digital surface between the foreground and background of its well-composed counterpart. To do so, we first approximated the former continuous analog by a surface and then compared the Euler characteristics of this surface with the Euler characteristics of the surface corresponding to the latter continuous analog. This is because the continuous analog of the digital surface between the foreground and background of the input image is not a surface. Thus, the notion of Euler characteristic is not applicable to it. Although we are not using the "true" continuous analog in this comparison, the approximate surface should give us a good idea of the topology of the large structures in the image. To generate the approximate surface and to extract the surface from the well-composed image, we used the Marching Cubes (MC) algorithm [4].

Table 6 shows the number of components, vertices, edges, faces, and holes of the approximate (triangulated) surfaces generated by the MC algorithm from the CSF, gray matter, and white matter segmented, binary images in group G2. The other components of the surfaces are really small compared to the largest ones. Furthermore, almost all of the remaining components are homeomorphic to a sphere. In turn, the Euler characteristics of the surfaces extracted by the MC

algorithm from the well-composed counterparts of the images in G2 (in each of the ten executions per image) disclosed two facts:

1. The number of vertices, edges, faces, and holes of the largest component of the extracted surfaces is smaller than the corresponding numbers in the largest component of the approximate surfaces. This is particularly the case for the surfaces extracted from the well-composed counterparts of the CSF segmented, binary image, as the number of holes of the largest component is about three times smaller than the number of holes of the largest component of the approximate surface.

2. The number of connected components of the extracted surfaces is larger than the one of the approximate surfaces. In particular, the extracted surfaces contains several small connected components (e.g., an average around 55, 123, and 15 connected components for the surfaces extracted from the well-composed images obtained from the CSF, gray matter, and white matter segmented, binary images in group G2, respectively.) These "extra" components are very small (e.g., 6 vertices, 12 edges, 8 faces, and 0 holes) and are all homeomorphic to a sphere.

The first fact indicates that the repairing algorithm thickens and smooths out the connected components of the input image (see Figures 13 and 14). The second fact tells us that the algorithm tends to introduce small "sphere-like components" in the resulting image. Luckily, these "sphere-like components" are really small, and they can be easily filtered out from the resulting image using an algorithm for removing small connected components. Such an algorithm will never introduce critical configurations back in the image. So, we can definitely incorporate such an algorithm as a post-processing step in our repairing algorithm.

Image	**# CC**	**# Vertices**	**# Edges**	**# Faces**	**# Holes**
G2, 1	34	467,678	1,417,236	944,824	2,368
G2, 2	31	719,096	2,164,974	1,443,316	1,282
G2, 3	11	393,798	1,183,266	788,844	313

Table 6: The number of connected components of the approximate surfaces obtained by the MC algorithm when given the images in group G2 (second column), and the number of vertices (third column), edges (fourth column), faces (fifth column), and holes (sixth column) of the largest connected component.

The number of instances of C1 (resp. C2) in the multivalued image in group G1 is 169,828 (resp. 26,858), which give us a total of 196,686 critical configurations. This total number of critical configurations is equal to the sum of the number of instances of C1 and C2 in each binary image that can be obtained by letting the foreground be the set of points of the multivalued image assigned the same color. On computing the total number of instances of C1 and C2, the instances that

occur in more than one binary image are counted only once. Note that the total number of critical configurations in G1 is about 2.77% of the number of points in the image grid, which is equal to 7,109,137.

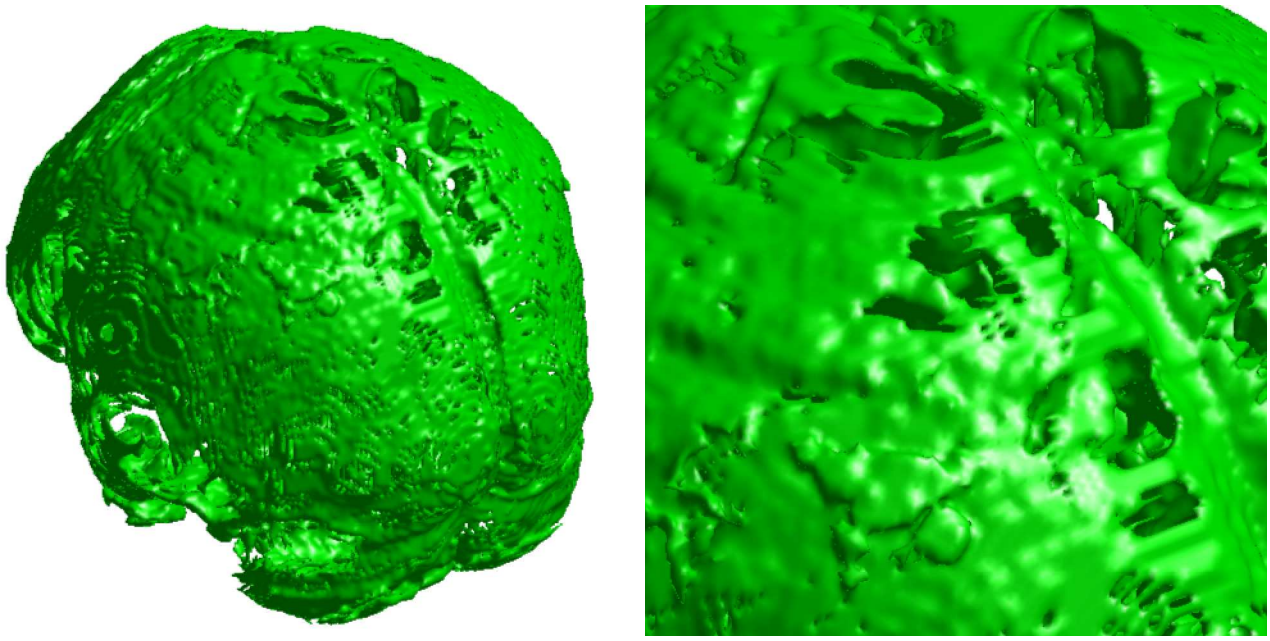

Figure 13: Approximate surface produced by the MC algorithm [4] from the CSF segmented, binary image in group G2.

We ran the ITK implementation of our repairing algorithm for multivalued images (see Section 6) on the image in group G1 using six distinct sequences for the rank of colors of the image. For each distinct sequence, we ran the algorithm ten times and computed the average PWCD and the average HD with respect to the input image and its resulting well-composed counterpart. Since the CSF, gray matter, and white matter are the most "important" structures in the image G1, each of the six color sequences started with a permutation of the values in $\{1, 2, 3\}$, which are the CSF, gray matter, and white matter values in G1. The order of the remaining colors in the sequence was the same for all sequences, namely, $8, 5, 7, 4, 9, 6, 0$. Table 7 shows the name of the structure associated with each color of the image in G1, the number of grid points assigned each color, the number of critical configurations in each binary image obtained from G1 by letting the foreground be the set of points associated with one color, and the ratio of the two previous quantities. Recall that the same instance of a critical configuration may show up in more than one of these binary images. So, the total number of critical configurations is not equal to the sum of the figures in the fourth column of Table 7.

Table 8 shows the average, minimum, and maximum PWCD for the ten executions of our algorithm on the image in G1 with color sequence $1, 2, 3, 8, 5, 7, 4, 9, 6, 0$. Table 8 also shows the

standard deviation from the average PWCD and the ratio PWCD divided by the number of grid points associated with each color. From Table 7 and Table 8, we can see that the average PWCD depends mostly on (1) the rank of the color in the input color sequence (the larger the rank the larger the average PWCD), and (2) the ratio in the fifth column of Table 7 (the larger the ratio the larger the average PWCD.) More specifically, a structure whose associated color has a larger rank will "lose" points during the repairing. Moreover, the larger the number of critical configurations of a structure associated with a color, c, the larger the amount of grid points whose assigned color changes to or from c.

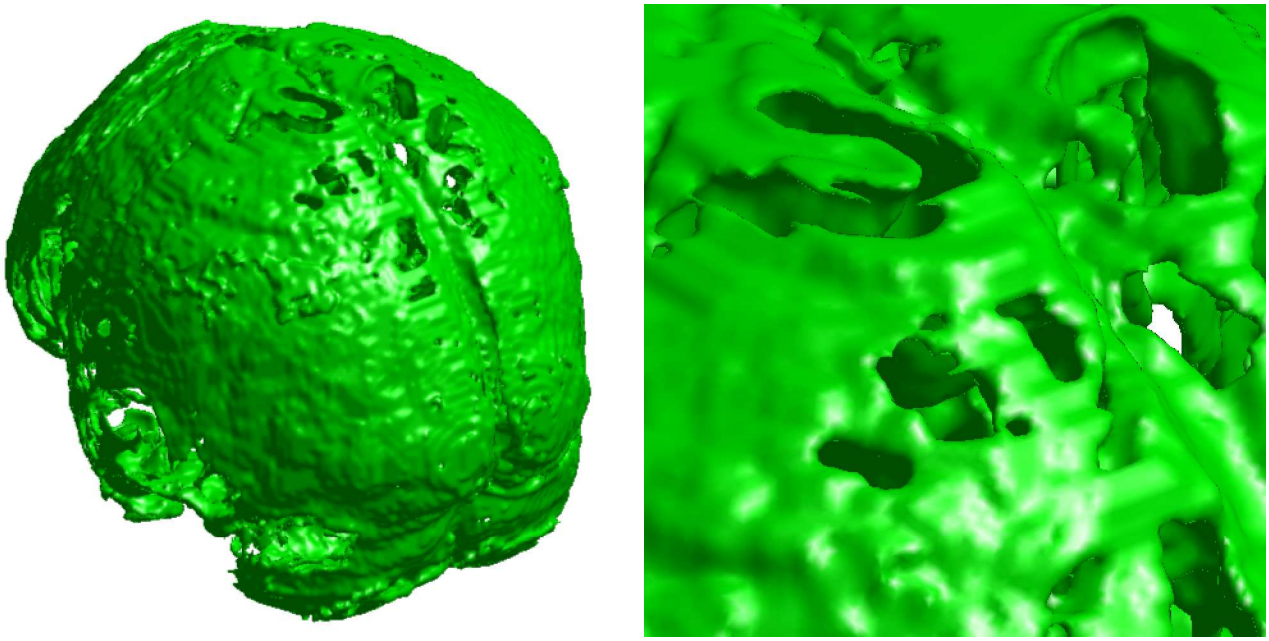

Figure 14: Surface extracted by the MC algorithm [4] from a well-composed image, which was in turn obtained by the repairing algorithm in Section 5 from the CSF segmented, binary image in group G2.

The results obtained by executing our repairing algorithm with the remaining five permutations were very similar to the ones in Table 8, with respect to the structures with colors $8, 5, 7, 4, 9, 6$, and 0. So, we only show the same measurements of Table 8 to the structures associated with the colors 1, 2, and 3 (see Table 9). By examining Table 9, we can notice an intricate effect of the strategy of color change used by our algorithm: the average PWCD for the white matter structure is smaller for the results associated with permutation 1,2,3 than it is for the results associated with permutation 1,3,2. Since color 3 ranks second in 1,3,2 and third in 1,2,3, this may seem odd. However, recall from Section 6 that whenever the algorithm eliminates a critical configuration in a $2 \times 2 \times 2$ neighborhood, $\mathcal{J}(p)$, without creating new ones, the algorithm changes the color of a point in $\mathcal{J}(p)$ to its "nearest" input sequence color among the colors assigned to all points in $\mathcal{J}(p)$. So, when the algorithm takes in the color sequences with permutation 1, 3, 2, the critical configurations

associated with the CSF structure will be mostly eliminated by changing color values from 1 to 3 and from 3 to 1.

Structure	**Color**	**# points**	**# CC**	**% (# CC / # points)**
Background	0	3,001,960	384	0.012
CSF	1	371,945	16,822	4.523
Grey matter	2	902,912	13,539	1.500
White matter	3	674,777	3,671	0.544
Fat	4	146,514	6,087	4.155
Muscle/Skin	5	617,482	75,783	12.273
Skin	6	726,649	63,245	8.704
Skull	7	362,561	9,271	2.557
Glial matter	8	5,987	3,292	54.985
Connective	9	298,350	51,002	17.095

Table 7: Each of the ten structures of the multivalued image in group G1 (first column), the color associated with each structure (second column), the number of grid points associated with each color (third column), the number of critical configurations in the binary image obtained from G1 by considering the foreground the points assigned the color in the second column (fourth column), and the ratio corresponding to the quantity in the fourth column divided by the quantity in the third column (fifth column).

Color	**Average**	**Minimum**	**Maximum**	**St. Dev.**	**%(PWCD / # Points)**
1	17,871.0	17,741	17,871	38.76	4.79
2	20,085.5	19,988	20,206	59.37	2.22
3	11,760.8	11,688	11,794	30.69	1.74
8	2,649.1	2,632	2,662	11.86	44.25
5	114,303.8	114,009	114,545	184.98	18.51
7	37,186.9	36,940	37,396	138.96	10.26
4	9,622.6	9,525	9,713	59.28	6.57
9	73,038.1	72,793	73,413	196.06	24.48
6	102,122.8	101,837	102,390	207.59	14.05
0	18,441.9	18,281	18,624	94.62	0.61

Table 8: The average (second column), minimum (third column), and maximum (fourth column) PWCD between each structure in the image in group G1 and the same structure in the well-composed image generated by our repairing algorithm using the color sequence $1, 2, 3, 8, 5, 7, 4, 9, 6, 0$; the standard deviation (fifth column); and the ratio of the average PWCD and the number of grid points associated with the color (first column) of each structure in the image in group G1 (sixth column).

From Table 9, we can also see that the average PWCD for the grey matter structure is much

smaller for the permutation 3,2,1 than it is for the permutation 1,2,3, even though color 2 ranks second in both permutations. This is because the algorithm never changes a color value from 2 or 3 to 1 when given a color sequence with permutation 3, 2, 1. So, for this permutation, many critical configurations associated with color 1 must have been eliminated by changing color values from 1 to c and from c to 1, where $c > 3$. Moreover, since the number of critical configurations associated with the white matter structure (color 3) is much smaller than the one associated with the color 1 (CSF structure in Table 7), the number of changes of color from 2 to 3 is smaller than the number of changes of color from 2 to 1 in the executions of the algorithm that take in permutations 3,2,1 and 1,2,3, respectively.

Color	**Perm.**	**Average**	**Min.**	**Max.**	**St. Dev.**	**%(PWCD / # Points)**
1	1,2,3	17,871.0	17,741	17,871	38.76	4.79
2	1,2,3	20,085.5	19,988	20,206	59.37	2.22
3	1,2,3	11,760.8	11,688	11,794	30.69	1.74
1	1,3,2	17,790.4	17,709	17,824	39.30	4.78
2	1,3,2	20,760.8	11,688	11,794	92.09	2.29
3	1,3,2	12,303.1	12,200	12,425	83.39	1.82
1	2,1,3	21,228.5	21,148	21,299	50.19	5.71
2	2,1,3	14,623.4	14,567	14,662	32.63	1.62
3	2,1,3	8,823.0	8,751	8,905	50.84	1.31
1	2,3,1	21,289.8	21,187	21,399	75.99	5.72
2	2,3,1	14,606.2	14,528	14,662	53.64	1.62
3	2,3,1	8,828.3	8,729	8,896	48.25	1.31
1	3,1,2	18,008.2	17,929	18,099	50.62	4.84
2	3,1,2	20,609.1	20,458	20,730	87.50	2.28
3	3,1,2	12,300.3	12,156	12,408	89.66	1.82
1	3,2,1	24,615.6	24,546	24,705	53.97	6.62
2	3,2,1	15,532.4	15,474	15,652	50.99	1.72
3	3,2,1	6,335.0	6,294	6,391	28.32	0.94

Table 9: The color of the structure (first column), the permutation, π, of the the colors 1, 2, and 3 (second column), the average (third column), minimum (fourth column), and maximum (fifth column) PWCD between each structure in the image in group G1 and the same structure in the well-composed image generated by our repairing algorithm using the color sequence $\pi(1), \pi(2), \pi(3), 8, 5, 7, 4, 9, 6, 0$; the standard deviation (sixth column); and the ratio of the average PWCD and the number of grid points associated the color (first column) of each structure in the image in group G1 (seventh column).

From the above discussion, we can conclude that the results of our repairing algorithm for multivalued images also depend on the adjacency relations of the structures in the image and their number of critical configurations. So, in many cases, it might be very hard, if not impossible, to specify an input color sequence that will enable the algorithm to produce a well-composed image

that minimizes the PWCD and yet favors certain structures. Furthermore, while the average PWCD for the CSF, grey matter, and white matter structures are still small (see Table 9) and relatively close to the average PWCD for the binary images with the same structures (see Table 3), the average PWCD for the structures associated with larger color ranks may be too large for certain applications. This is the case for structures with a large number of critical configurations (compared to their number of points) and with medium to high color ranks, e.g., the glial matter structure (see Table 7 and Table 8).

Table 10 shows the average, minimum, and maximum HD, as well as the standard deviation, for the ten executions of our algorithm on the image in G1 and color sequence $1, 2, 3, 8, 5, 7, 4, 9, 6, 0$. Unlike the average PWCD, the average HD gets larger very quickly as the color rank gets larger. However, it is also mostly dependent on the color rank and the number of critical configurations of the structure (relative to its number of points). The effect of the "change to the nearest color" strategy used by our repairing algorithm can also explain why the average HD gets larger very quickly as the color rank gets larger. Once a grid point has its assigned color changed to a color, say c, the color c may become an option for the set of color choices in a $2 \times 2 \times 2$ neighborhood that did not have any point assigned c. So, the algorithm may create a small path of points assigned the color c, diverging from the color component associated with c. Since the HD measure is very sensitive to the presence of such paths, we observe the large variations in the averages shown in Table 10.

Color	**Average**	**Minimum**	**Maximum**	**St. Dev.**
1	1.45	1.41	1.73	0.10
2	2.53	1.45	2.83	0.16
3	12.33	12.12	12.53	0.21
8	3.47	3.00	3.74	0.25
5	14.34	14.18	15.00	0.35
7	7.81	7.81	7.81	0.00
4	9.66	8.25	11.18	0.85
9	7.60	7.55	8.06	0.16
6	7.07	7.07	7.07	0.00
0	20.83	20.64	21.02	0.20

Table 10: The average (second column), minimum (third column), and maximum (fourth column) HD between each structure in the image in group G1 and the same structure in the well-composed image generated by our repairing algorithm using the color sequence $1, 2, 3, 8, 5, 7, 4, 9, 6, 0$; and the standard deviation (fifth column).

The results obtained by executing our repairing algorithm with the five remaining permutations were very similar to the ones in Table 10, with respect to the structures with associated colors $8, 5, 7, 4, 9, 6$, and 0. Therefore, in Table 11, we only provide the measurements associated with the colors 1, 2, and 3 (see Table 11). Note that the values of the average HD and the values of the

average PWCD (see Table 9) for the CSF, grey matter, and white matter structures agree: the larger the average PWCD the larger the average HD. Figure 15 shows an axial slice of the resulting well-composed image from one of the ten executions of the algorithm on the image in G1 and color sequence 1, 2, 3, 8, 5, 7, 4, 9, 6, 0.

Color	**Perm.**	**Average**	**Min.**	**Max.**	**St. Dev.**
1	1,2,3	1.45	1.41	1.73	0.10
2	1,2,3	2.53	1.45	2.83	0.16
3	1,2,3	12.33	12.12	12.53	0.21
1	1,3,2	1.41	1.41	1.41	0.00
2	1,3,2	4.99	4.12	5.92	0.91
3	1,3,2	3.19	2.83	4.12	0.50
1	2,1,3	1.41	1.41	1.41	0.00
2	2,1,3	4.95	4.24	5.83	0.77
3	2,1,3	1.88	1.41	2.45	0.28
1	2,3,1	5.15	4.47	5.83	0.72
2	2,3,1	1.77	1.41	2.24	0.25
3	2,3,1	9.61	8.54	12.12	1.72
1	3,1,2	1.71	1.41	3.16	0.55
2	3,1,2	4.84	3.61	5.83	1.00
3	3,1,2	3.20	2.45	4.12	0.55
1	3,2,1	5.37	4.36	7.21	1.21
2	3,2,1	4.51	3.61	5.10	0.65
3	3,2,1	2.64	2.24	3.16	0.39

Table 11: The color of the structure (first column); the permutation, π, of the the colors 1, 2, and 3 (second column), the average (third column), minimum (fourth column), and maximum (fifth column) HD between each structure in the image in group G1 and the same structure in the well-composed image generated by our repairing algorithm using the color sequence $\pi(1), \pi(2), \pi(3), 8, 5, 7, 4, 9, 6, 0$; and the standard deviation (sixth column).

8 Conclusion and Future Work

We described a new repairing algorithm for generating a 3D well-composed image from a given 3D digital binary image. Our algorithm is randomized and its time and space complexities are linear in the size of the input image grid. The algorithm produces its output by iteratively modifying the input image, changing the color of its background points so that these points become foreground points. Each color change eliminates a critical configuration from the currently modified input image. We derived an upper bound on the expected number of background points of the input image that are made into foreground points in the output image. This upper bound is a theoretical

guarantee for the similarity between the input and output images. We also showed the results of an experiment in which our algorithm was executed on human brain images with and without varying levels of noise.

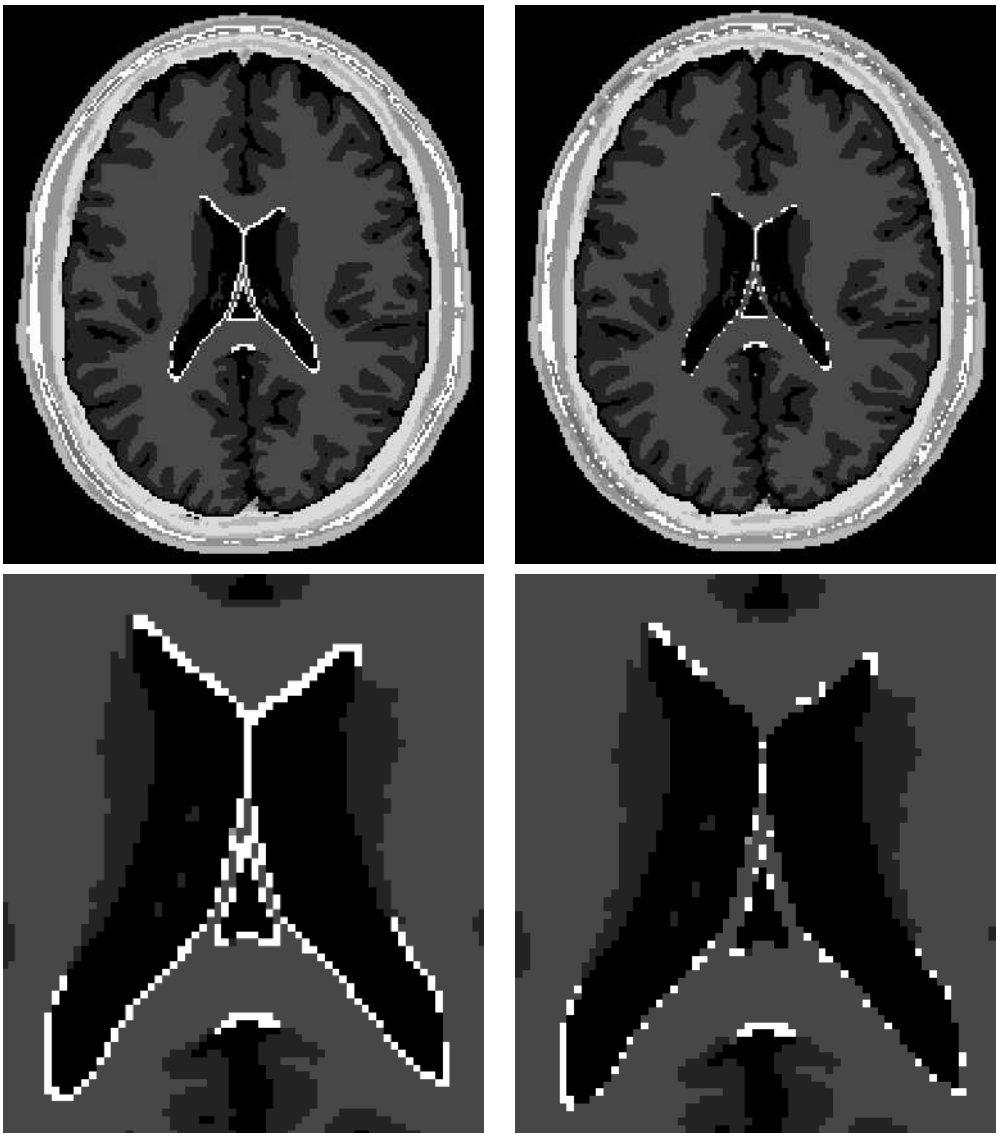

Figure 15: The top left image is an axial slice of the multivalued image in G1 whereas the top right image is the same slice in the well-composed image produced by our repairing algorithm in Section 6 from the image in G1. The bottom images are close-up views of the ventricles in the top images.

We also introduced an extension of our algorithm for the repairing of 3D digital multivalued images. This extended algorithm can be used for generating a 3D well-composed multivalued image

from a given 3D digital multivalued image, which is assumed to be the result of a multi-object segmentation of another 3D digital multivalued image [17]. We also presented results from an experiment in which our extended repairing algorithm was tested against a 3D multivalued digital image of the human brain.

The applicability of both algorithms depends on how sensitive the target application is to the point-wise color difference between the input and output images. On the one hand, our experiments showed that our repairing algorithm for binary images seems to produce images for which the point-wise color difference is about the same as the number of critical configurations in the input image. Since the number of instances of C1 and C2 in images encountered in practical applications is typically a small percentage of the image grid size, we strongly believe that our algorithm can be successfully used by many practical applications that can be simplified or optimized for speed whenever the input is a well-composed image. On the other hand, our repairing algorithm for multivalued images seems to largely modify the image structures associated with least "important" colors. So, it may be more appropriate for a situation in which the input image has only a few important structures (e.g., the human brain), and we do not care for the modifications in the least important ones.

As we mentioned in Section 5, there is a trivial algorithm for producing the most similar 3D well-composed image from a given 3D digital binary image, but its time complexity is exponential in the size of the input image grid, which rules out its practical use. So, it is natural to ask if there is a polynomial time algorithm that computes the most similar well-composed image. Currently, we do not know if such an algorithm exists, and we still have not ruled out the possibility that the equivalent *decision problem* may be **NP**-complete: given a 3D digital binary image and a positive integer k, is there a 3D well-composed image whose point-wise color difference from the given image is less than or equal to k? We are currently looking at this decision problem. Recall that our repairing algorithm for binary images can only change the color of background points. The purpose of this restriction is to guarantee the termination of the algorithm. If we remove this restriction and find a way of guaranteeing the termination of the modified algorithm, the resulting well-composed images may have smaller point-wise color difference than the ones generated by our current algorithm.

There are two simple heuristics that can be incorporated into our algorithms to increase (resp. decrease) the number of critical configurations eliminated (resp. created) by one change of color. First, to increase the number of critical configurations eliminated by one change of color, we can scan the image grid for background points only. Then, for each background point, p, such that (1) p is part of a critical configuration and (2) no new critical configurations is created by changing the color of p to 1, we place p in a dynamic max-heap, where the key of p is the number, $n(p)$, of critical configurations eliminated by changing the color of p to 1. Finally, we remove a point from the heap at a time, change its color, and then update the key of each point, q, in the heap for which $n(q)$ changed. So, we change the colors of the background points that eliminate more critical configurations first, and yet do not create any new critical configuration. This heuristic

is not randomized. Second, to decrease the number of new critical configurations created by one change of color, we can use a branch-and-bound strategy that looks ahead at the number of new critical configurations created by the algorithm in a bigger grid neighborhood. Then, the algorithm can choose among a small set of "sequences" of color changes, as opposed to small sets of points. We plan to incorporate both heuristics in the next version of the current implementations of our algorithms in the ITK library [38].

We also believe that the application of a topology simplification algorithm (e.g., [33]), followed by the application of a repairing algorithm, may produce a well-composed (binary) image with very few topological artifacts. The reason is that the application of a topology simplification algorithm will produce a binary image with a few (or none) topological artifacts (i.e., very tiny handles), and our repairing algorithm is very unlikely to add tiny handles to the image, as shown by the Hausdorff distance measurements in Table 4. We intend to perform some experimental tests to validate our assumption.

A major drawback of both of our repairing algorithms is that they are solely based on topological information. Furthermore, they make decisions regarding the elimination of critical configurations that do not take into account neither the local nor the global topology of the color components of the image. Some of the most recent algorithms for correcting topology [27, 28] incorporate statistical, topological and geometric information into their decision making process. We believe that our repairing algorithms can leverage some of the features of these algorithms. In particular, the first author of this manuscript is currently collaborating with some other researchers on the modification of the segmentation algorithm in [17]. The goal is to incorporate the topological repairing into the segmentation process, so that we can generate well-composed images as a result of the segmentation process. By using the statistical and geometric information available for the algorithm in [17], we hope to improve the decision making process of the repairing. Some preliminaries results have already been obtained.

References

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

[2] P. Stelldinger and U. Köthe. Towards a general sampling theory for shape preservation. *Image and Vision Computing Journal*, 23(2):237–248, 2005.

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

[4] 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.

[5] J.-O. Lachaud and A. Montanvert. Continuous analogs of digital boundaries: A topological approach to iso-surfaces. *Graphical Models*, 62(3):129–164, 2000.

[6] G.M. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proceedings of the 2nd IEEE Conference on Visualization (Visualization'91)*, pages 83–91, San Diego, California, USA, October, 22-25 1991.

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

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

[9] T. Lewiner, H. Lopes, A. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[10] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–27, 2003.

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

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

[13] E.M. Stokely and S.Y. Wu. Surface parametrization and curvature measurement of arbitrary 3-d objects: Five practical methods. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(8):833–840, 1992.

[14] H. Delingette. Initialization of deformable models from 3d data. In *Proceedings of the 6th International Conference in Computer Vision (ICCV'98)*, pages 311–316, Bombay, India, January, 4-7 1998.

[15] N. Krahnstoever and C. Lorenz. Computing curvature-adaptive surface triangulations of three-dimensional image data. *The Visual Computer*, 20(1):17–36, 2004.

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

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

[18] P. Stelldinger, L.J. Latecki, and M. Siqueira. Topological equivalence between a 3d object and the reconstruction of its digital image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):126–140, 2007.

[19] 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.

[20] J.-F. Mangin, V. Frouin, I. Bloch, J. Regis, and J. Lopez-Krahe. From 3d magnetic resonance images to structural representations of the cortex topography using topology preserving deformations. *Journal of Mathematical Imaging and Vision*, 5:297–318, 1995.

[21] D. MacDonald, N. Kabsni, D. Avis, and A.C. Evans. Automated 3-d extraction of inner and outer surfaces of cerebral cortex from mri. *Neuroimage*, 12(3):340–355, 2000.

[22] B. Fischl, A. Liu, and A. M. Dale. Automated manifold surgery: Constructing geometrically accurate and topologically correct models of the human cerebral cortex. *IEEE Transactions om Medical Imaging*, 20(1):70–80, 2001.

[23] D.W. Shattuck and R. M. Leahy. Automated graph-based analysis and correction of cortical volume topology. *IEEE Transactions on Medical Imaging*, 20(11):464–472, 2001.

[24] X. Han, C. Xu, U. Braga-Neto, and J. L. Prince. Topology correction in brain cortex segmentation using a multiscale, graph-based approach. *IEEE Transactions om Medical Imaging*, 21(2):109–121, 2002.

[25] X. Han, C. Xu, and J.L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):755–768, 2003.

[26] S. Bischoff and L. Kobbelt. Sub-voxel topology control for level-set surfaces. *Computer Graphics Forum*, 22(3):273–280, 2003.

[27] P.-L. Bazin and D. L. Pham. Topology correction using fast marching methods and its application to brain segmentation. In J. S. Duncan and G. Gerig, editors, *Proceedings of the 8th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 3750 of *Lecture Notes in Computer Science*, pages 484–491, Palm Springs, California, USA, October 26-29 2005.

[28] F. Segonne, E. Grimson, and B. Fischl. A genetic algorithm for the topology correction of cortical surfaces. In Gary Christensen and Milan Sonka, editors, *International Conference on Information Processing in Medical Imaging*, volume 3565 of *Lecture Notes in Computer Science*, pages 393–405, Glenwood Springs, Colorado, USA, July 10-15 2005.

[29] N. Kriegeskorte and R. Goeble. An efficient algorithm for topologically segmentation of the cortical sheet in anatomical mr volumes. *Neuroimage*, 14(2):329–346, 2001.

[30] K. V. Leemput, F. Maes, D. Vandermeulen, and P. Suetens. Automated model-based tissue classification of mr images of the brain. *IEEE Transactions on Medical Imaging*, 18(10):897–908, 1999.

[31] I. Guskov and Z. Wood. Topological noise removal. In *Proceedings of the 2001 Conference on Graphics Interface*, pages 19–26, Ottawa, Ontario, Canada, June 7-9 2001.

[32] Z. Aktouf, G. Bertrand, and L. Perroton. A three-dimensional holes closing algorithm. *Pattern Recognition Letters*, 23(5):523–530, 2002.

[33] Andrzej Szymczak and James Vanderhyde. Extraction of topologically simple isosurfaces from volume datasets. In *Proceedings of the 14th IEEE Conference on Visualization 2003 (Visualization'03)*, pages 67–74, Seattle, WA, USA, October 19-24 2003.

[34] Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, 2004.

[35] Gabor Herman. *Geometry of Digital Spaces*. Birkhäuser: Boston, MA, USA, 1998.

[36] Reinhard Klette and Azriel Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan-Kaufmann: San Francisco, CA, USA, 2004.

[37] A. Rosenfeld. Fuzzy digital topology. *Information and Control*, 40:76–87, 1979.

[38] N. Tustison, M. Siqueira, and J. Gee. Well-composed image filters for repairing 2-d and 3-d binary images. *The Insight Journal*, July-December 2006. (http://hdl.handle.net/1926/305).

[39] 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.