Algorithms for Computing Topological Invariants in Digital Spaces

Li Chen

University of the District of Columbia, Washington, D.C., USA
lchen@udc.edu

Abstract. Based on previous results in digital topology, this paper focuses on algorithms related to topological invariants of objects in 2D and 3D Digital Spaces. Specifically, we are interested in hole counting objects in 2D and closed surface genus calculation in 3D. We also present a proof of the hole counting formula in 2D. This paper includes fast algorithms and implementations for topological invariants such as connected components, hole counting in 2D, and boundary surface genus for 3D. For 2D images, we designed a linear time algorithm to solve the hole counting problem. In 3D, we also designed a $O(n)$ time algorithm to obtain the genus of a closed surface. These two algorithms are both in $O(\log n)$ space complexity.

Keywords: Digital space · Images · Number of holes · Genus of surfaces · Algorithm · Time and space complexity

1 Introduction

Gathering topological properties for objects in 2D and 3D space is an important task in image processing. An interesting problem, called hole counting, involves summarizing the number of holes found in 2D images. In 3D, people in computer graphics or computational geometry usually uses triangulation to represent a 3D object. It uses the marching-cube algorithm to transfer a digital object into the representation of simplicial complexes, which requires a very large amount of computer memory.

This paper introduces fast algorithms for these calculations based on digital topology. This paper provides a complete process to deal with simulated and real data in order to obtain the topological invariants for 2D and 3D images. The algorithms are: (1) 2D hole counting, and (2) 3D boundary surface genus calculation.

One of the most difficult parts of real-world image processing is dealing with noise or pathological cases. This paper also gives detailed procedures for detecting such cases and will provide reasons for modifying the original image into an image where the mathematical formula could be applied.

The results of this paper have considerable potential to be used in big data processing, especially topological data processing [8, 10, 19].

© Springer Nature Switzerland AG 2021
M. Nguyen et al. (Eds.): ISGV 2021, CCIS 1386, pp. 203–218, 2021.
https://doi.org/10.1007/978-3-030-72073-5_16
2 Background Concepts of Digital Spaces

Digital topology was developed for image analysis in digital space, which is a discrete space where each point can be defined as an integer vector [13]. Some definitions of digital connectedness can also be found in [1,13]. The following figure provides some examples of digital space (Fig. 1).

Fig. 1. Examples of basic unit cells and their connections: (a) 0-cells, (b) 1-cells, (c) 2-cells, (d) 3-cells, (e) Point-connected 1-cells, (f) Point-connected 2-cells, and (f) Line-connected 2-cells.

Certain related theorems using Euler characteristics and the Gauss-Bonnet theorem have already been proven. The first is about simple closed digital curves. $C$ is a simple closed curve in direct (4-) adjacency where each element in $C$ is a point in $\Sigma_2$.

We use $IN_C$ to represent the internal portion of $C$. Since direct adjacency has the Jordan separation property, $\Sigma_2 - C$ will be disconnected.

We also call a point $p$ on $C$ a $CP_i$ point if $p$ has $i$ adjacent points in $IN_C \cup C$. In fact, $|CP_1| = 0$ and $|CP_i| = 0$ if $i > 4$ in $C$.

$CP_2$ contains outward corner points, $CP_3$ contains straight-line points, and $CP_4$ contains inward corner points.
For example, the following center point is an outward corner point in an array (see Fig. 2):

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 1 \\
0 & 1 & x
\end{array}
\]

However, in the next array, the center point is an inward corner point:

\[
\begin{array}{ccc}
0 & 1 & x \\
1 & 1 & x \\
x & x & x
\end{array}
\]

Fig. 2. Outward corner points and inward corner points

In [2], we used the Euler theorem to show these results for a simple closed curve \( C \):

**Lemma 1.**

\[ CP_2 = CP_4 + 4. \]  \( \text{(1)} \)

In 3D, let \( M \) be a closed (orientable) digital surface in 3D grid space with direct adjacency. We know that there are exactly 6 types of digital surface points (see Fig. 3), which was discovered by Chen and Zhang in [6]. Related definitions involving digital surfaces can be found in [6].

Assume \( M_i \) (\( M_3, M_4, M_5, M_6 \)) is the set of digital points with \( i \) neighbors in the surface. \( M_4 \) and \( M_6 \) have two different types, respectively.

As we know, the Gauss-Bonnet theorem states if \( M \) is a closed manifold, then

\[ \int_M K_G dA = 2\pi \chi(M) \]  \( \text{(2)} \)

where \( dA \) is a small element representing area and \( K_G \) is the Gaussian curvature. Its discrete form is
\[ \Sigma \{p \text{ is a point in } M\} K(p) = 2\pi \cdot (2 - 2g) \]  

where \( g \) is the genus of \( M \). Chen and Rong obtained the following [7]:

\[ g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8. \]

For a \( k \)-manifold, homology group \( H_i, i = 0, ..., k \) provides information on the number of holes in each \( i \)-skeleton of the manifold. When the genus of a closed surface is obtained, we can then calculate the homology groups corresponding to its 3D manifold [9, 11].

### 3 Hole Counting Algorithms in 2D

Hole counting in 2D describes the number of holes in a 2D image. Previous research can be found in [12, 17]. In this paper, we use a simple method to solve the hole counting problem.

A line or curve in the real world, for human interpretation, always has a degree of thickness regardless of how thin the structure is. However, a digital line may be interpreted differently. For example, Fig. 4 shows how similar digital objects can produce different results.

Images in the digital world have a great deal of differentiability than humans shown in Fig. 4. The difference between (1) and (2) is not the same as the difference between (2) and (3). This is because Fig. 4(1) can be interpreted as a (square-)dotted line for one of its legs in terms of direct adjacency or 4-adjacency. A dotted line is a collection of several disconnected objects in 4-adjacency. Figure 4(4) may have no hole, one hole, or two holes. In 8-adjacency, there is no hole, and in 4-adjacency, the points are not connected. The best way in image processing would be to use 8-adjacency for “1”s (foreground) and 4-adjacency for “0”s (background). Only Fig. 4(6) will give a result most similar to the human interpretation in that there are two holes.
Algorithm for Computing Topological Invariants in Digital Spaces 207

Fig. 4. Similar 2D digital objects with topological differences: (1) an angle with thickness = 1 using MS Paint Software, (2) an angle with thickness = 2, (3) an angle with thickness = 3, (4) an ellipse and a line with thickness = 1, (5) an ellipse and a line with varying thickness, and (6) an ellipse and a line with thickness = 2.

If we use 8-adjacency for “1”s and 4-adjacency for “0”s, this type of adjacency is called (8,4)-adjacency. This may cause other problems. For instance, let us assume we have two parallel “1” lines with one “0” line between them at a 45-degree angle. Each “0” point will be reviewed as a separated component, and they do not form a connected “0” line. This is also against human interpretation.

Our method assumes $C$ does not contain the following cases (if there is any, we will modify the original image to remove these scenarios, which is discussed later):

\[
\begin{align*}
1 & 0 \\
0 & 1 \\
\end{align*}
\]

and

\[
\begin{align*}
0 & 1 \\
1 & 0 \\
\end{align*}
\]

These two cases are called pathological cases (see Fig. 5).

It is obvious that the solution we present does not solve all problems or cover all cases and is strict in its use. However, the advantage with our method is to obtain a simple treatment.

Our algorithm fills or deletes certain points in the original image to avoid pathological cases. We also want to remove single points, whether they are black or white, and treat them as noise.

Topological invariants should maintain the Jordan property. We only allow direct adjacency in order to deal with topological invariants, at least in most cases.
Note that the number of holes in a connected component in 2D images is a basic invariant. With this, a simple formula can be proven using our previous results in digital topology [2].

3.1 The Simple Formula for the Number of Holes in S

An image segmentation method can extract a connected component. A connected component $S$ in a 2D digital image is often used to represent a real object. The identification of the object can be done first by determining how many holes are in the component. For example, letter “A” has one hole and “B” has two holes. In other words, if $S$ has $h$ holes, then the complement of $S$ has $h + 1$ connected components (if $S$ does not reach the boundary of the image).

**Theorem 1.** Let $S \subset \Sigma_2$ be a connected component and its boundary $B$ be a collection of simple closed curves without pathological cases. Then, the number of holes in $S$ is

$$h = 1 + (CP_4 - CP_2)/4$$  \hspace{1cm} (5)

$CP_4, CP_2 \subset B$.

In this section, we provide a proof using the 3D formula to get the theorem for holes.

**Proof:** Let $S \subset \Sigma_2$ be a connected component where its boundary does not have any pathological cases, something we can actually detect in linear time.

We can embed $S$ into $\Sigma_3$ to make a double $S$ in $\Sigma_3$. On the $z = 1$ plane, we have an $S$, denoted by $S_1$. We also have the exact same $S$ on the $z = 2$ plane, denoted by $S_2$.

Without loss generality, $S_1 \cup S_2$ is a solid object. (For simplicity, we omit some technical details for the strict definition of digital surfaces here.) Its boundary consists of closed digital surfaces with genus $g = h$. We know

$$g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8$$
It is easy to see that there will be no points in $M_6$ on the boundary of the 3D object bounded by $S_1 \cup S_2$. We might as well just using $S_1 \cup S_2$ to represent the solid object.

For each point $x$ in $CP_2$ of $C \subset S$ where $C$ is the boundary of $S$, we will have two points in $M_3$ in the solid $S_1 \cup S_2$. In the same way, if a point $y$ is inward in $CP_4 \in C$, then we will get two points in $M_5$ in the solid $S_1 \cup S_2$. Since there is no point in $M_6$, i.e., $|M_6| = 0$. Therefore, $2|C_2| = |M_3|$ and $2|C_4| = |M_5|$. We have

$$h = g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8 = 1 + (2|C_4| - 2|C_2|)/8$$

Thus,

$$h = 1 + (|C_4| - |C_2|)/4.$$  

We can also prove this theorem using Lemma 1 for digital curves: $CP_2 = CP_4 + 4$ for a simple closed curve. This theorem can also be proven intuitively [6].

We can see that this is a very simple formula for obtaining the number of holes (genus) for a 2D object and does not require any additional sophisticated algorithms. It only counts if the point is a corner point, inward point, or outward point.

However, we could not get a similar simple formula for triangulated representation of 2D objects since some of these special properties only hold in digital space.

### 3.2 Algorithms for Hole Counting

The key of the algorithm is to avoid all pathological cases. There are two types of such cases, and it may be difficult to decide whether we need to add or delete a point in order to avoid having a pathological case.

The first method is based on the original grid space. Sometimes, deleting a pixel to change a pathological case may result in another pathological case. In such scenarios, we add a pixel instead to change the original pathological case and vice versa.

In other scenarios where adding or deleting pixels will not eliminate the pathological case, we use delete since eventually it will complete the job. This is the example shown in Fig. 6.

The second method applies to the many instances of the image shown in Fig. 6. We need to consider a refinement method, e.g. using half-grid space to avoid the deletion and addition of a cell that will result in another pathological case. This method increases the amount of space needed. We use a half-size cell to fill the space or delete a half-size cell in order to remove the pathological case.
Algorithm 3.1. Algorithm for calculating the number of holes in 2D images.

**Step 1.** Get connected components using direct-adjacency (4-adjacency).

**Step 2.** Extract every component using the following: Fill all single “0” pixels (unit-square or 2-cell) in as “1” if all their indirect neighbors (8-adjacency) are “1”. Delete all single “1” pixels if all their indirect neighbors (8-adjacency) are “0”.

**Step 3.** Find all pathological cases. Delete or add a pixel if this action does not create a new pathological case. If both actions create a new pathological case, then use delete. Repeat this step until all pathological cases are removed.

**Step 4.** Count all inward edge points $CP_4$ and outward points $CP_2$.

**Step 5.** $h = 1 + (CP_4 - CP_2)/4$.

Note that if deleting a pixel means changing the single “0” pixel (unit-square or 2-cell) to a “1” pixel. This may delete a hole or reduce a noise pixel. Do a refinement may be better for this case. The above calculation is to obtain holes for each component. Algorithm 3.1 is an $O(\log n)$ space algorithm without Step 1 since it is only counting the number of point types. If we assume the number of pathological cases is constant, this algorithm will have linear time complexity.

4 Algorithms and Implementations for the Genus of Digital Surfaces in 3D

Basically, the topological properties of an object in 3D contains connected components, genus of its boundary surfaces, and other homologic and homotopic properties. In 3D, the problem of obtaining fundamental groups is decidable, but no practical algorithm has yet been found. Therefore, homology groups have played the most significant role [6,9].

Theoretical results show that there exist linear time algorithms for calculating genus and homology groups for 3D Objects in 3D space [9]. However,
the implementation of these algorithms is not simple due to the complexity of real data sampling. Most of the algorithms require triangulation of the input data since it is collected discretely. However, for most medical images, the data was sampled consecutively, meaning that every voxel in 3D space will contain data. In such cases, researchers use the marching-cubes algorithm to obtain the triangulation since it is a linear time algorithm [15]. In addition, the spatial requirements for such a treatment will be at least doubled by adding the surface-elements (sometimes called faces). Another defect of the marching-cubes algorithm is that it may generate a 3D object that is not a strict mathematical manifold even though the original triangulation is. Chen suggested using the so called convex-hull boundary method to complete such a task in digital geometry [5,6].

The theoretical work of calculating genus based on simple decomposition will turn into two different procedures: (1) finding the boundary of a 3D object and then using polygon mapping, also called polygonal schema, and (2) cell complex reductions where a special data structure will be needed.

In this paper, we look at a set of points in 3D digital space, and our purpose is to find the homology groups of the data set. The direct algorithm without utilizing triangulation was proposed by Chen and Rong in year 2008 [7]. However, this algorithm is based on the strict definition of digital surfaces. Many real 3D data sets may not satisfy this definition. In other words, a set of connected points may not be able to undergo such a process without further considerations using theoretical or practical methods.

In [7], we discuss the geometric and algebraic properties of manifolds in 3D digital spaces and the optimal algorithms for calculating these properties. We consider digital manifolds as defined in [6]. We presented a theoretical optimal algorithm with time complexity $O(n)$ to compute the genus and homology groups in 3D digital space, where $n$ is the size of the input data [7]. More information related to digital geometry and topology can be found in [13,14]. Now we present the theoretical algorithm in [7] below.

**Algorithm 4.1.** Let us assume we have a connected $M$ that is a 3D digital manifold in 3D.

**Step 1.** Track the boundary of $M$ and $\partial M$, which is a union of several closed surfaces. This algorithm only needs to scan though all the points in $M$ to see if the point is linked to any point outside of $M$, which will be a point on the boundary.

**Step 2.** Calculate the genus of each closed surface in $\partial M$ using the method described in Sect. 2. We just need to count the number of neighbors on a surface. and put them in $M_i$ using formula (4) to obtain $g$.

**Step 3.** Using theorem [7,9], we can get $H_0$, $H_1$, $H_2$, and $H_3$, where $H_0$ is $Z$. For $H_1$, we need to get $b_1(\partial M)$, which is the summation of the genus in all connected components of $\partial M$. (See [11] and [9].) $H_2$ is the number of components in $\partial M$, and $H_3$ is trivial.

**Lemma 2.** Algorithm 4.1 is a linear time algorithm.
Therefore, we can use linear time algorithms to calculate \( g \) and all homology groups for digital manifolds in 3D based on Lemma 2 and formula (4). We also have:

**Theorem 2.** There is a linear time algorithm to calculate all homology groups for each type of manifold in 3D.

However, this algorithm could not be directly used for real data set. We will explain next.

### 4.1 Practical Algorithms and Implementations

Above algorithm 4.1 is based on the condition that the a closed surface separates a 3D solid object into connected components. It is related to Jordan manifolds, meaning that a closed \((n - 1)\)-manifold will separate the \( n \)-manifold into two or more components. For such a case, only direct adjacency will be allowed since indirect adjacency will not generate Jordan cases. The following figure shows four cases that must be deleted before performing the algorithm. These cases are called pathological cases in 3D.

![Four pathological cases in 3D](image)

We can see that when a data set contains indirect adjacent voxels, we need a procedure to detect the situation and delete some voxels in order to preserve the homology groups.

Assuming we only have a set of points in 3D. We can digitize this set into 3D digital spaces. There are two ways of doing so: (1) by treating each point as a cube-unit, called the raster space, (2) by treating each point as a grid point, called the point space. These two are dual spaces. Using the algorithm described in [2], we can determine whether the digitized set forms a 3D manifold in 3D space in direct adjacency for connectivity. The algorithm can be completed in linear time.

In terms of algorithms for connected component search, Pavlidis realized that one can use Tarjan’s breadth-first-search (BFS) or depth-first-search (DFS) for images. The complexity of the algorithm is \( O(n) \) [16]. In 3D, we have 6-, 18-,
26-connectivity. Since real data contains noise, it is better to consider all of these connectivities. Therefore, we use 26-connectivity to get the connected components. Since such connected component of real processing is not strictly 6-connected. The topological theorem generated previously in [7] is no longer applicable. Therefore we want to transform a 26-connected component to a 6-connected component.

In theory, this type of transformation should be done using the careful adding or deleting process since optimization of the minimum number of changes could be an NP-hard problem. A similar problem was considered in [18] where a decision problem using adding voxels was proposed.

Now we present the practical algorithms as follows:

**Algorithm 4.2.** Assume the Input is 3D points or voxels in 3D. We can treat 3D points to voxels. Let \( B \) be the boundary of the 3D object \( M \).

**Step 1:** Separate the set of voxels to connected components using BST or DFS algorithm in 3D. We have a set of connected 3D objects. Let's working on a connected 3D object \( M \).

**Step 2:** Delete pathological cases:
The following rules (observations) are reasonable: In a neighborhood \( N_{27}(p) \) that contains 8 cubes and 27 grid points, use following procedure:

a) If a voxel only shares a 0-cell with another voxel, then this voxel can be deleted.

b) If a voxel only shares a 1-cell with another voxel, then this voxel can be deleted.

c) If a boundary voxel \( v \) shares a 0,1-cell with another voxel, assume \( v \) also shares a 2-cell with a voxel \( u \) on the boundary, then \( u \) must share a 0,1-cell with a voxel that is not in the object \( M \). Deleting \( v \) will not change the topological properties.

d) If, in a \( 2 \times 2 \times 2 \) cube, there are 6 boundary voxels and their complement (two zero-valued voxels) are the case in Fig.7(a). Add a voxel to this \( 2 \times 2 \times 2 \) cube such that the new voxel shares as many 2-cells in the set as possible. This means that we want the added voxel to be inside the object where possible. (Record if more 3D objects are created by this step and put them in different sets/memory locations).

**Step 3:** Track the boundary \( M \), said \( B \), which is a union of several closed surfaces. This algorithm only needs to scan though all the points in \( M \) to see if the point is linked to a point outside of \( M \). That point will be on boundary.

**Step 4:** Calculate the genus of each closed surface (e.g., \( B(1) \ldots, B(k) \)) in the boundary \( B \). The method is the following: since there are six different types of boundary points (on the boundary surface), two of them has 4 neighbors, and two of them has 6 neighbors in the surface (Fig. 3). We can use \( M_3, M_4, M_5, M_6 \) to denote the numbers of different types. \( M_i \) represents the number of points each of which has \( i \)-neighbors. The genus \( g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8 \).

**Step 5:** Get homology groups \( H_0, H_1, H_2 \), and \( H_3 \). \( H_0 \) is \( Z \). For \( H1 \), we need to get \( b_1(B) \) that is just the summation of the genus in all connected components
in \( B \). \( H_2 \) is the number of components in \( B \). \( H_3 \) is trivial. These are based on the Theorem in [7,9].

Therefore, we can use linear time algorithms to calculate \( g \) and all homology groups for digital manifolds in 3D.

**Theorem 3.** There is a linear time algorithm to calculate all homology groups for each type of manifold in 3D when using Algorithm 4.2.

Since this algorithm going through each point to test whether or not the neighborhood is on the boundary, it is a linear time and \( O(\log(n)) \) space algorithm.

We add the following details to give more explanation to Step 2 of Algorithm 4.2: (1) There are only two cases in cubical or digital space [2] where two voxels (3-cells) share a 0-cell or 1-cell. Therefore, the algorithm needs to modify the voxel set so two voxels share exactly a 2-cell (or are not adjacent), or there is a local path (in the neighborhood) where two adjacent voxels share a 2-cell [2,6]. A case was mentioned in [18], which is the complement of the case where two voxels share a 0-cell (see Fig. 7(a)). Such a case may create a tunnel or it could be filled. Here, the algorithm simplifies the problem by adding a voxel in a \( 2 \times 2 \times 2 \) cube. The similar case in point-space can also refer to the case in Fig. 7(a). We would like to point out that detection is easy but deleting certain points (and deleting the minimum number of points) to preserve homology groups seems to be a bigger issue.

![Fig. 8. Genus calculation for lab data.](image)

4.2 Implementations and Data Samples

We implemented Algorithm 4.2 in C++ where the program modifies the date to fit the theoretical definition of 3D manifolds and their boundary surfaces that are supposed to be digital surfaces defined in [2]. We have tested many cases.
using lab data. The results for the genus were correct. It means that detection processing (plus adding or deleting some data points) preserves the topology of the objects. Two examples are shown in Fig. 8. We also applied the program to real data samples. From our observations, the results were also correct, see Fig. 9.

![Fig. 9. Genus calculation for 3D real bone images.](image)

Note that when the object becomes more complex, it is possible for pathological situations to arise (Fig. 10).

![Fig. 10. Topological edge detection with different thresholds for 2D images.](image)
5 Remarks on Programming

Programming and image display differ from theoretical or conceptual drawing. For different purposes, we sometimes need to use raster space, meaning that we treat a 3D point (a voxel) as a 3-cell. In other cases, we need to treat a 3D point as a 0-cell. The refinement can be used to reduce some conflicts but will enlarge the memory space needed. For 2D cases, we also need a similar consideration. The following figure shows the 2D topological edge detection with different thresholds. The software was made by the author.

![Figure 11](image.png)

**Fig. 11.** Boundary tracking and display for different thresholds of 2D images.

In general, if the input data is in raster space, the boundary surface would be in point space. We must first make the translation from raster space to point space. Then, for each point on the surface, we count how many neighbors exist in order to determine its configuration category. The following figure shows the boundary tracking and display. The original data contains two holes inside. We track the boundary and display using thin-voxels as shown by the last image in Fig. 11.

6 Summary and Discussion

In this paper, we use digital topology to get a formula for calculating the number of holes in a connected component in 2D digital space. The formula is simple and can easily be implemented. The author does not know whether this formula has already been studied or obtained by other researchers. For 3D images,
we developed a practical way of calculating the genus by extracting boundary surfaces and removing pathological cases. Both algorithms are optimal in terms of time and space complexity (assuming the number of pathological cases are constant). In the future, we will make Python programming code to use the methods described in this paper to topological big data analysis.

This paper is modified based on a previously unpublished note [4] and an informally published conference paper [3]. The original version of this paper was posted in 2013 on arxiv.org at https://arxiv.org/abs/1309.4109. We did some modification in this version.

References

1. Brimkov, V.E., Klette, R.: Border and surface tracing. IEEE Trans. Pattern Anal. Mach. Intell. 30(4), 577–590 (2008)
2. Chen, L.: Discrete Surfaces and Manifolds. SP Computing, Rockville (2004)
3. Chen, L.: Genus computing for 3D digital objects: algorithm and implementations. In: Proceedings of the Workshop on Computational Topology in Image Context 2009, Austria, pp. 29–40 (2009)
4. Chen, L.: Determining the number of holes of a 2D digital component is easy, November 2012. http://arxiv.org/abs/1211.3812
5. Chen, L.: Digital Functions and Data Reconstruction. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-5638-4
6. Chen, L.: Digital and Discrete Geometry. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-12099-7
7. Chen, L., Rong, Y.: Linear time recognition algorithms for topological invariants in 3D. In: Proceedings of International Conference on Pattern Recognition (ICPR08) (2008)
8. Chen, L., Biswas, S.: Digital curvatures applied to 3D object analysis and recognition: a case study. In: Barneva, R.P., Brimkov, V.E., Aggarwal, J.K. (eds.) IWClA 2012. LNCS, vol. 7655, pp. 45–58. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34732-0_4
9. Dey, T.K., Guha, S.: Computing homology groups of simplicial complexes in $R^3$. J. ACM 45, 266–287 (1998)
10. Gholizadeh, S.: Topological data analysis in text processing. Doctoral Dissertation, The University of North Carolina at Charlotte (2020)
11. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
12. Lin, X., Yang, J., Summers, W.: Hold holes countable in binary images. Technical report, School of Computer Science, Colubus State University (2010). http://cs.columbusstate.edu/about_us/HolesCounting-Technical-Report.pdf
13. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Image Analysis. Morgan Kaufmann, Burlington (2004)
14. Kong, T.Y., Rosenfeld, A. (eds.): Topological Algorithms for Digital Image Processing. Elsevier, Amsterdam (2006)
15. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. Comput. Graph. 21(4), 163–169 (1987)
16. Pavlidis, T.: Theodosios Algorithms for Graphics and Image Processing. Computer Science Press (1982)
17. Qian, K., Bhattacharya, P.: Determining holes and connectivity in binary images. Comput. Graph. 16(3), 283–288 (1992)
18. Siqueira, M., Latecki, L.J., Tustison, N., Gallier, J., Gee, J.: Topological repairing of 3D digital images. J. Math. Imaging Vis. 30(3), 249–274 (2008)
19. Snášel, V., Nowaková, J., Xhafa, F., Barolli, L.: Geometrical and topological approaches to big data. Future Gener. Comput. Syst. 67, 286–296 (2017)