Chapter 6
Topological Data Analysis

Li M. Chen

Abstract Classical data processing uses pattern recognition methods such as classification for categorizing data. Such a method may involve a learning process. Modern data science also uses topological methods to find the structural features of data sets. In fact, topological methods should be the first step before the classification method is applied in most cases. Persistent homology is the most successful method for finding the topological structure of a discrete data set.

This chapter deals with topological data processing. We first introduce space triangulations and decompositions. Then, we discuss manifold learning and focus on persistent analysis. We give an overview of all topological methods but will focus on persistent data analysis.

6.1 Why Topology for Data Sets?

What is the difference between two data sets? This depends on what is the most important issue we care about these two data sets. When we talk about apples, the size or the number of apples is important. When we talk about automated car license plate recognition, we need to care about getting a good, clean image.

Netflix, an Internet based movie streaming and renting company, would want to predict revenues from a specific movie that targets a certain age group or demographic. We might ask: What is the volume (number of individuals) of people in an area? Can local internet servers in a city handle the movie stream? Is *city A* similar to *city B*?

We can classify the data into two categories for a certain movie: (1) people who are definitely interested and (2) people who are not interested.

We have discussed the NetFlix matrix completion problem in Chap. 2. Now the problem discussed here is more general especially in topological aspects. This is because that matrix completion is filling the holes using subspace, the completion of filling has so many options, a particular one might not reflect the actual data.

L.M. Chen (✉)
The University of the District of Columbia, Washington, DC, USA
e-mail: lchen@udc.edu

L.M. Chen et al., *Mathematical Problems in Data Science*,
DOI 10.1007/978-3-319-25127-1_6

On the other hand, holes in the matrix might be real. There are a group of people who will not watch a particular movie, we need to identify holes not to fill the holes.

Considering all the factors, Netflix will obtain a multidimensional cloud data set. Now, before Netflix purchases the rights to show a movie nationwide, they may test the movie in a small city. When the company has received the data for who or which families are watching the movie, the data will be translated into a set of vectors.

Even if similarities between cities exist, they may not be exactly the same. We can use deformation at the top of the training data to predict the topological structure of the large data set for a big city. Netflix can find the appropriate volume of people who are going to order the internet movie stream.

Another example is more practical and is used in medical imaging specifically bone data imaging. There are many holes in a particular part of a bone. The number of holes can indicate the strength of the bone. How do we calculate the correct number of holes [4, 5].

The third problem is related to wireless networking. We know we want to build several tower stations in a small city to handle the communication of cell phone users. Every stations will have a radio power that will cover certain area, usually circular area. We want to know if there is a hole where no station will reach it [6, 7]? These type of questions require us to find the structure or shape of the data. The topological structure is just the basic structure of these problems. We might be able to find valuable problems relating to topology in other business related data mining [30].

6.2 Concepts: Cloud Data, Decomposition, Simplex, Complex, and Topology

Cloud data, sometimes called scattered data, is a set of data points that are randomly arranged. They are usually dense, meaning that they have a lot of data points. Cloud computing, as well as cloud data computing, is highly related to networking.

A Problem of Wireless Networking Let us assume that we have a set of sites where each site has a tower station for network communication. When we drive a car from one location of a site to another, how do we handle the location change in terms of communication?

In fact, networking experts have already designed a way to switch a user (a cell-phone) from one tower station to another by measuring distances. We always select the nearest host station to the user to handle communication.

The method in mathematics is called the Voronoi diagram. The method pre-partitions the space into polygons. When a user enters in a specific polygon called Voronoi region, then the corresponding tower station takes over the service. This question is referred to in Chap. 1. Section 1.5.

The Voronoi Diagram The Voronoi diagram partitions a plane into polygons, where each polygon contains a sample point called a site; a point x is inside a specific polygon containing site p if x is closer to p than to any other site [4, 9, 35].

The Voronoi diagram method is particularly important to science and engineering. This partition is made based on the closest distance from the given site compared to other sites. The dual diagram of the Voronoi decomposition is a triangulation of the domain, which is called Delaunay triangulation.

Delaunay triangulation is the most popular form among different types of triangulation. See Fig. 3.4 from [4]. We will present the algorithm of Delaunay triangulation in the next section.

A space usually can be partitioned into smaller pieces. This is called a decomposition. Image segmentation is a type of decomposition. If these pieces are all triangles, we say this is a triangulation. We usually use polygons for more general decompositions. A triangle is called a 2D simplex (2-simplex). In a triangulation, two triangles share an edge (1-simplex), a point (0-simplex), or an empty set. A collection $\mathscr{K}$ of 2-simplexes, 1-simplexes, 0-simplexes (with the empty set) is called a simplicial complex if (a) the intersection of any two simplexes is also in $\mathscr{K}$, and (b) each edge or points of a simplex is an element in $\mathscr{K}$.

A topological space is a pair of $M = (X, \tau)$ where X is a set and τ is a collection of subsets of X. M satisfies (1) X and $\emptyset$ are in τ, (2) The union of any number of members of τ is in τ, and (3) The intersection of finite number of members of τ is in τ. We can see that a simplicial complex (usually contains finite number of simplexes) can be regarded as a topological space. We can see that a decomposition induces a simplicial complex.

In many other cases especially in digital images, we use squares to decompose a space. A square or other shape is called a cell, so we will have cell complexes. A triangulation of a space forms the simplicial complexes that is a foundation of Combinatorial Topology [4, 9, 13]. The formal definition of topology can be found in [4, 13].

6.3 Algorithmic Geometry and Topology

As we discussed in the last section, partitioning a 2D region into triangles is called triangulation. We can also partition a region into polygons where each polygon can be viewed as a cell. Therefore, we will have a cell complex. In higher dimensions, this is called a polyhedron. The polyhedron decompositions such as Voronoi diagrams are specific research areas in computational geometry also called algorithmic geometry. Cell complexes are in the research area of topology and have a long history in mathematics [13].

A complex that is usually a finite topology is a collection of cells in different dimensions. For the k-complex, M is the complex whose biggest dimension is k. If for each point (0-cell) in M, we have a group of k-cells containing this point that is homeomorphic to k-dimensional Euclidean space, then this complex is called a k-manifold.

Computational topology studies the problems related to manifolds in computing including algorithm design and practical applications [9, 34].

In this section, we discuss a classic problem of algorithmic geometry that is related to space decomposition and a relatively new problem that is called manifold learning.

6.3.1 Algorithms for Delaunay Triangulations and Voronoi Diagrams

Mathematically, the method of the most meaningful decompositions is the Voronoi Diagram: Given n points (called sites) on a plane, we partition a space into several regions. Any point in the region is closer to the site in its own region. This type of region is called the Voronoi region.

For a new point x, find the closest site to x. In other words, if we have a new point x and try to find the closest site to the new point, we only need to decide which Voronoi region contains x. This problem is called the nearest neighbor problem as we discussed in Chap. 2. That is also used in pattern recognition.

Fortune found an optimum algorithm for Voronoi diagrams with time complexity $O(nlogn)$, but it is difficult to implement [4]. Here, we first design an algorithm for Delaunay triangulation. This relatively simple algorithm is called Bowyer–Watson algorithm for the Delaunay triangulation .

Let P_i, $i = 1, \cdots, n$ be n sites. A Voronoi region is bounded by edges and vertices. The vertex of the region is called the Voronoi point. We know that: (1) An edge of the Voronoi region must have the property that each point on the edge must be equal distance to the two sites. (2) A Voronoi point must have the property of being equidistant to the three sites.

This means there must be a circle containing these three sites centered at each Voronoi point. This circle is the circumcircles of the triangle containing all three site points. Such a triangle is called a Delaunay triangle.

Another definition of Delaunay triangulation is as follows: no circumcircle of any triangle contains a site. The following Bowyer–Watson algorithm is designed based on this fact. The Bowyer–Watson algorithm is one of the most commonly used algorithms for this problem. This method can be used for computing the Delaunay triangulation of a finite set of points in any number of dimensions. After the Delaunay triangulation is completed, we can obtain a Voronoi diagram of these points by getting the dual graph of the Delaunay triangles. See Fig. 6.1 [4].

Algorithm 6.1. The Bowyer–Watson algorithm is incremental in that the algorithm works by adding one point at a time to a valid Delaunay triangulation of a subset of the desired points. Then, it works in the new subset, adding points to reconstruct the new Delaunay triangulation.

Step 1 Start with three points of the set. We make the first triangle by linking three points with three edges.

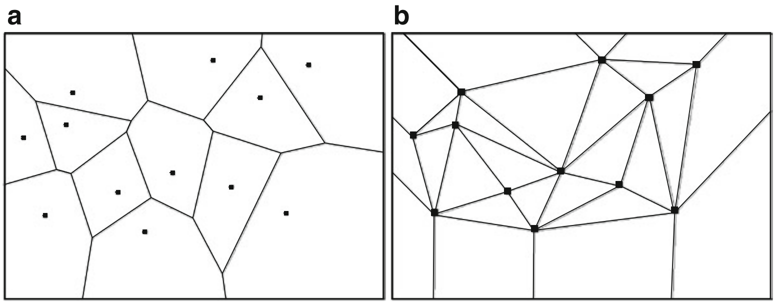


Fig. 6.1 Voronoi and Delaunay diagrams: (**a**) a Voronoi diagram; (**b**) Delaunay triangulation

Step 2 Insert a new point P. Draw the circumcircle of each existing triangle. If any of those circumcircles contains the new point P, the triangle will be marked as invalid.

Step 3 Remove all invalid triangles. This process will leave a convex polygon hole that contains the new point.

Step 4 Link the new point to each corner point of the convex polygon to form a new triangulation.

This algorithm is not the fastest one, and it runs in $O(n\sqrt{(}n))$ time. To get the Voronoi diagram from the Delaunay triangulation, the key is to link the centers of the circumcircles such that two corresponding triangles share an edge. For more details of this algorithm, see [4, 9, 35].

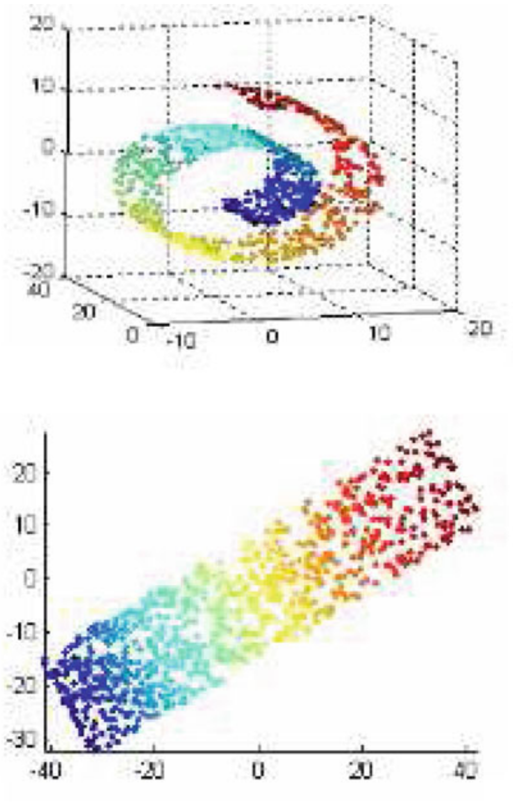
6.3.2 Manifold Learning on Cloud Data

Finding a surface from a 3D data volume is a very challenging job. However, if the data in the volume is continuous in terms of values, we can find an iso-surface, meaning that the value at each data point is the same. Then we can easily get the data set by searching for the same value in its neighborhood.

6.3.2.1 Real Problems Related to Manifold Learning

Isosurface Determination in High Dimensional Space In meteorology data processing, the researcher usually uses sensors to detect the temperatures or humidity data in the air. So reconstructing the shape from this type of sampling data points is not easy when we think the data is discrete points. In the past, we have some way that can draw a contour map. However, in such a case, we usually have a set of dense samples. We just need to extract the surface on which each point has the same or similar value called an isosurface. However if the data samples are shown

Fig. 6.2 Manifold learning on cloud data [4, 17]



like Fig. 6.2a, it is hard to know the 2D neighborhood of a point in a desired surface (e.g. an isosurface). Manifold learning is a way to find such a surface in a high dimensional space.

Object Deformation Sequencing in a Set of Images Given a set of images, see Fig. 6.3, can we determine a subset of images that are most likely in a deformation sequence? In this problem, we will transfer the images or objects to feature vectors, we want to find a smooth curve(s) in the space that holds these vectors (Fig. 6.4). We then extract the curves and corresponding images. Such a smooth curve is a (1D) manifold. This is also a good example in manifold learning.

Manifold learning here is to identify a lower dimensional manifold where most of the data will be located.

Manifold learning is always related to dimensionality reduction. For instance, we want to find a surface, a 2D object, from 3D space.

In Fig. 6.2, a 2D Swiss roll that was embedded in a 3D space [4, 17]. The question is how do we extract the information?

For this particular case, we want to know if the data points represent a sphere or a spiral shape [17, 32] (Manifold learning).

Fig. 6.3 Example of a set of facial images [32]

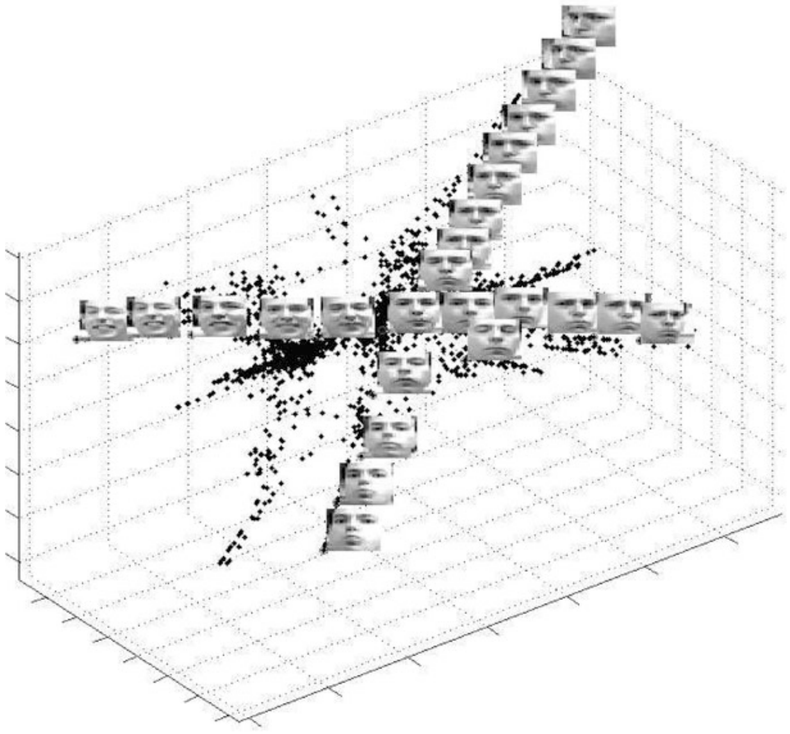


Fig. 6.4 Vector points of the set of facial images in 3D space [32]

Learning a data set that will most likely represent a random shape is a very difficult task. Therefore, there are many tasks to be studied in manifold learning [17, 32].

Two most popular methods for manifold learning are the Isomap and the kernel principal component analysis [4, 17]. The idea of the Isomap method is to use k-NN or MST to get the neighborhood information. Then, a graph is constructed with weight. After that, we use Dijkstra's algorithm to find the shortest path for each pair of points.

Let us present the Isomap algorithm below [4]:

Algorithm 6.2. The Isomap algorithm. Let S be a cloud data set.

Step 1 For each point $x \in S$, determine the neighbors of each point. We can use k-NN or MST to get the information.
Step 2 Construct the graph with the neighbors found in Step 1. The edge will be weighted by Euclidean distance.
Step 3 Calculate the shortest path between two vertices using Dijkstra's algorithm.
Step 4 Multidimensional scaling creates edges between two vertices. Cut off the data points beyond the clip level. Determine the local dimension of the data. This step is called lower-dimensional embedding.
Step 5 Compare the results from Step 4. Make a decision on the dimensions for all local neighborhoods. For instance, most local neighborhoods are 2D and so we make the 2D out put.

In Step 5, we can use principal component analysis to help us find the local dimensions [4, 17, 25]. In fast algorithm design, there may be some geometric data structures that can assist us to find new faster algorithms. The moving kernel will be determined by eigenvalues and eigenvectors. This method is referred to as the kernel principal component analysis.

In [32], a method called maximum variance unfolding (MVU) and the PCA method are used to find the image deforming sequence in Fig. 6.4. These results can be used to make better image reconstruction when we know the path of face change in deformation. See Fig. 6.5 [32]. More discussion can be found in [15]. This type of manifold learning has direct relationships to object tracking in videos. We have introduced it in Chap. 5. In manifold learning, we usually deal with very high dimensional data sets. For instance, we can treat a 32×32 picture as a $32 \times 32 = 1024$ dimensional vector. The calculation of this type is toward to a BigData type.

For massive images, cloud computing machines can be used to detect all possible directions of the pictures. This is also related to Bigdata analysis. Newly developed software, called SPARK, by researchers at Berkeley may play a significant role in this type of calculation [**?**]. In Chap. 8, Su et al. will discuss a special technique to manifold learning for data science.

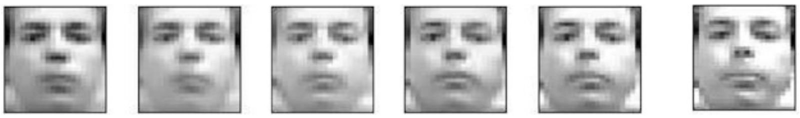


Fig. 6.5 An example of ideal image reconstruction after finding the deformation line in Fig. 6.4

6.4 Persistent Homology and Data Analysis

As we discussed in the beginning of this chapter, individual sample points do not have a topological structure other than the discretely located points in space. However, human's interpretation makes a set of discrete points to have some meanings. The "best" interpretation is the structure we are looking for. A technology called persistent homology analysis was proposed to solve this problem [2, 8, 12]. It can be used to find the topological structure of a data set.

Let us have a set of cloud data, each point is an independent component. We can interpret each data point as a sample of a point, a small disk, or a cubical volume. We can also use this point represents a relatively bigger area, a bigger disk. When the size of the disks increases, the shape of the data sets may change its topological properties. For example, the data points may connect to be a object (component). See Fig. 6.6. The intuitive meaning of persistence is the topological property such as the number of components or the number of holes does not change in a period of time as the radius of disks changes.

While changing the value of radius, the homology or the number of holes does not always change, so the fit with the most unchanged holes is referred to as the best fit. This method is called persistent homology analysis [13, 36]. This method is becoming one of the major developments in topological data analysis.

6.4.1 *Euler Characteristics and Homology Groups*

In topology, homology usually indicates the number of holes in each dimension. We start at the Euler characteristic that is an invariant to a topological manifold. For any finite simplicial or cell complex, the Euler characteristic can be defined as the alternating sum

$$\chi = k_0 - k_1 + k_2 - k_3 + k_4 - k_5 \cdots , \tag{6.1}$$

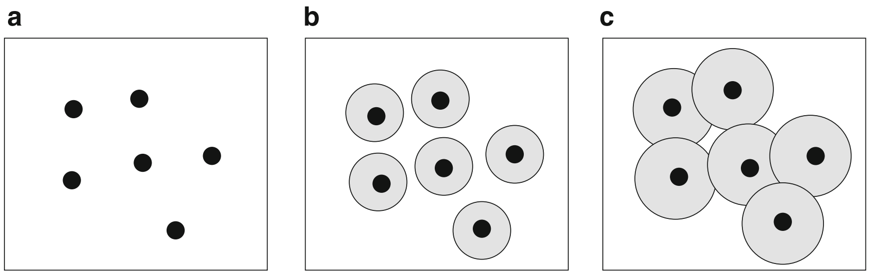


Fig. 6.6 Example of persistent analysis: (**a**) original data points, (**b**) the covered area by small disks centered at original points, and (**c**) the covered area by larger disks centered at original points; the topology is changed

where k_i denotes the number of i-cells in the complex. The detailed explanation of homology groups can be found in [13] and a concise introduction can be found in [4]. We only give a definition of homology groups here.

Let C_i be a group generated by i-cells in M. It is called a chain group where each element is in the form of $\Sigma_{A_i} n_i \cdot A_i$, where A_i is an i-cell and n_i is an integer. C_i is an Abelian group.

Let ∂_i be the boundary operator from C_i to C_{i-1}. The boundary operator only sends the i-cell to its boundary (with the direction already defined). We can see that the boundary of an i-cell is an "$(i-1)$-cycle." The combination of an i-cell, for instance $kA + k'B$, will map to an element in C_{i-1} by ∂_i.

The ith homology group, H_i, is defined as

$$H_i(X) := Kernal(\partial_i)/Image(\partial_{i+1}), \tag{6.2}$$

For example, let a square $A = (a, b, c, d)$ be a 2-cell. Then, we have 4 edges, $(a, b), (b, c), (c, d), (d, a)$. So $\partial_2(A) = (+1)(a, b) + (+1)(b, c) + (+1)(c, d) + (+1)(d, a) = (+1)(a, b) + (+1)(b, c) + (+1)(c, d) + (-1)(a, d)$. $\partial_2(A)$ is an element in C_1. $Kernal(\partial_i)$ means the set of elements in C_i that maps to 0, the identity element in the Abelian group. In fact, all cycles in C_i will map to 0 in C_{i-1}. (The boundary of a cycle is empty, which means 0). $Image(\partial_{i+1})$ is the image of mapping ∂_{i+1}. They are boundaries of elements in C_{i+1}. Every boundary (in C_i) is a "cycle." Therefore, $Image(\partial_{i+1})$ is a subgroup of $Kernal(\partial_i)$.

To further explain, C_i is a group that contains all combinations of i-cells (with connected-sum as we can intuitively say). $Kernel(\partial_i)$ sends all cycles to $0 \in C_{i-1}$. $Image(\partial_{i+1})$ is a set of cycles in C_i that has filled $(i + 1)$-cells in K_{i+1} (the $(i + 1)$-sectionskeleton of M). This means that for the boundary of grouped $i + 1$-cells, the boundaries are in C_i. We want to pull them out of $Kernel(\partial_i)$ and only leave empty cycles in C_i. The removal means $Kernal(\partial_i)/Image(\partial_{i+1})$ boundaries.

The ith Betti number b_i is the rank of H_i. For example, if Z^n is a free Abelian group, then rank$(Z^n) = n$. We also have

$$\chi = b_0 - b_1 + b_2 - b_3 + \cdots, . \tag{6.3}$$

The Betti numbers are topological invariants that indicate the connectivity of simplicial or cell complexes. Let x be n-dimensional manifold. If X is closed and oriented, then the Poincare duality holds:

$$b_k = b_{n-k}.$$

In 2D, b_1 describes the maximum number of 1-cuts needed to separate a surface into two pieces.

We also have $\chi = 2 - 2g$, where g is the genus. Intuitively, b_i refers to the number of i-dimensional holes on a topological surface. So (1) b_0 is the number of connected components, (2) b_1 is the number of different type of one-dimensional holes (of a cycle in terms of deformation or homeomorphism), and (3) b_2 is the number of two-dimensional tunnels (cavities).

For a sphere S^n, where $n > 1$, we have $b_0 = b_n = 1$ and all other $b_i = 0$. For example, a torus has one connected component ($b_0 = 1$), two circular holes (b_1, one in the center and the other in the middle of the "donut"), and one two-dimensional cavity or tunnel (b_2, the inside of the "donut"), which yields Betti numbers. There is no 3D-cell so $b_i = 0$ for all $i \geq 3$.

Understanding homology groups is not a simple task. After extensive consideration and practice, we suggest methods for interpreting homology groups using the following rules:

1. Let M be an n-manifold or cell complex in general where b_0 is always the number of connected components. Now M can only be connected.
2. Draw an i-cycle σ_i (a manifold that is homomorphic to an i-sphere, S^i) in M, where $i < n$. If every other i-cycle can be deformed to the original σ_i, then $b_i = 0$ (if H_i is the remaining element in the group, then the group no longer exists).
3. If M is a closed n-manifold without a boundary, then $b_n = 1$. The boundary is the only "cycle" or closed boundary in n-dimension. $H_n = Kernel(\partial_n)/\{e = 0\}$, $\{e = 0\}$ is the identity group. M will map to 0 in C_{n-1} and is the generator of the group. For the same reason, if M contains k closed n-manifolds (where each manifold is "minimal"), then $b_n = k$. For instance, if M is 2-sphere, S^2, then $b_0 = 1$ and $b_1 = 0$ since every cycle on S^2 is deformable to another cycle (called homotopic) and $b_2 = 1$.
4. As we have said in (2), i-cycle σ_i and its deformed i-cycle will be treated the same. If there is another cycle and we cannot deform σ_i to it, then $b_i > 1$. If the example is for a torus, then we have $b_1 = 2$, but $b_0 = 1$ and $b_2 = 1$.
5. If M is a disk (or 2-ball), then $b_2 = 0$ since there is no 2-cycle. There are 1-cycles, but they are all homotopic (deformable to each other), so $b_1 = 1$. In general, for n-ball B^n, $b_0 = 1$ $b_1 = 0, \cdots, b_n = 0$.

The two following examples are very practical in application.

Example 6.1. For a graph $G = (V, E)$, if it is connected then $b_0 = 1$ and b_1 is the number of "minimal" cycles, which are called generators of H_1. A cycle is not "minimal" if it can be constructed by two (or more) "minimal" cycles using connected-sum.

Example 6.2. Let F be a connected image or picture in 2D. According to Rule (1), $b_0 = 1$. Based on Rule (5), $b_2 = 0$. The boundary of a hole is a cycle, which is minimal. According to Rule (3), b_1 is just the number of holes. We can also verify this using formula (6.1) and (6.3). Since $b_0 - b_1 + b_2 = k_0 - k_1 + k_2$, we have $b_1 = 1 + k_1 - k_0 - k_2$. For example, in Hather's book, for X_2 we have $k_0 = 2$, $k_1 = 4$, and $k_2 = 1$. There are two holes since $b_1 = 1 + k_1 - k_0 - k_2 = 1 + 4 - 2 - 1 = 2$. We can conclude that there are exactly two holes in the Euclidean 2D plane as it is shown in [13].

When we deal with applications in the following sections, we discuss how we can calculate and use algorithms that cannot be solved with the human eye.

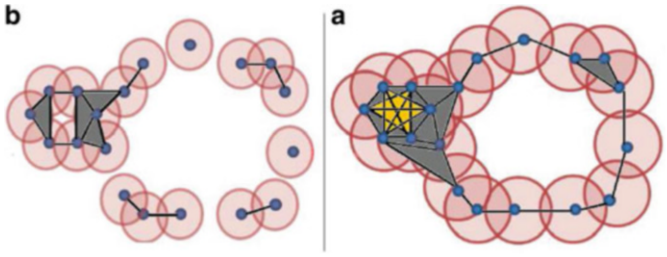


Fig. 6.7 Examples of Vietoris–Rips complex: (**a**), build VR complex using small disk, and (**b**) build VR complex using a relatively larger disk. [20]

6.4.2 *Data Analysis Using Topology*

Persistent data analysis has become more important in recent years. Many researchers have done some analysis on actual point cloud data sets. Algorithmically, it is not very efficient if we use small disks to cover the area and then do a homology calculation.

There is a method that is more practical called the Vietoris–Rips complex. It is related to the Cech complex that is made by the topological cover of $B(x, r)$.

Definition 6.1 (Vietoris–Rips Complex). Let X be a subset of a metric space with metric d. Choose a small real number $\epsilon > 0$. Construct a simplicial complex in the following way inductively:

(1) For each point in X, make it as a 0-simplex.
(2) For each pair $x_1, x_2 \in X$, make a 1-simplex $([x_1, x_2])$ if $d(x_1, x_2) \leq \epsilon$.
(3) For $x_1, x_2, \cdots, x_n \in X$, make an $(n-1)$-simplex with vertices $x_1, x_2, \cdots, x_n$. Then, $d(xi, xj) \leq \epsilon$ for all $0 \leq i, j \leq n$; that is, if all the points are within a distance of ϵ from each other.

This simplicial complex can also be denoted as $VR(X, \epsilon)$. The problem of this simplex is that we need extensive calculations to construct the complex, especially when X is a large set. It requires $O(2^{\{|X|\}})$ time complexity to determine such a simplex if we directly implement a simple algorithm as suggested by the definition, i.e. we check all possible simplexes. See Fig. 6.7 [20]. The construction process is called the Vietoris–Rips filtration. There has been software developed to build the complex [19, 20].

An algorithm for obtaining all possible Vietoris–Rips complex for each $r = \epsilon$ is not economic if the data set is very big. In the next section, we will find a very fast algorithm for 2D and 3D problems (Fig. 6.7).

6.5 Digital Topology Methods and Fast Implementation

Using a digital method, we can easily solve the problem of homology groups in 2D and 3D [4, 5]. This method is called digital geometry topology [4].

Digital geometry and topology was developed to provide a solid foundation for image processing and computer graphics. For 2D images, a connected component usually means an object in the image. A digital image can be viewed as an array; for simplicity, it can be viewed as a binary array with values only in $\{0, 1\}$. The value 0 indicates the background. Some theoretical development of digital topology related to digital surfaces are still under investigations [4, 16].

The topological structure of a connected component in 2D is essentially to find how many holes are in the component.

Thinking about the Vietoris–Rips or Cech complexes, these methods usually use $B(X, r)$ as the (open) covering set. If we use digital point plate (or square) $DB(X, r)$ to cover each point in X, then we will have the same topology. As we discussed above, for a digital component, we have $b_0 = 1$, $b_2 = 1$, and $b_1 = h$ where h is the number of holes in the component. If $DB(X, r)$ is a good cover, meaning that each intersection is simply connected (contractible), then the space made by $DB(X, r)$ has the same homology (groups) as the original manifold or complex sampled as the point set X.

The set $DB(X, r)$ refers to the digital cell complex. See [4] for details. The advantages of using $DB(X, r)$ instead of the Vietoris–Rips complex or $B(X, r)$ is that we can use the properties of digital topology to get a very fast algorithm in 2D and 3D.

To get $DB(X, r)$ algorithmically in a fast way is also interesting. We can use 6, 18, or 26 adjacency to determine whether the cubic cells are in $DB(X, r)$. For each r, we can obtain the homology groups or Betti numbers to get the persistent homology. In the following two subsections, we will introduce the digital methods.

For higher dimensional complexes, we still need to use the Vietoris–Rips complex. We would also need further research to archive the simplicity of computing.

6.5.1 2D Digital Holes and Betti Numbers

There is a very simple formula to get the number of holes in 2D digital space. 2D digital hole counting begins with the following.

Proposition 6.1. *For a connected component that has at least one 2-cell in 2D, we will have $b_0 = 1$, $b_2 = 1$ (at least one 2-cell in an image), and $b_1 =$ number of holes.*

This is because: (a) $H_0 = Z$ when the image is connected, (b) $H_2 = 0$ since ∂_3 does not exist, and (c) according to the definition, $\chi = \Sigma(-1)^i b_i = \Sigma(-1)^i k_i$. So $1 - b_1 + 0 = k_0 - k_1 + k_2$. For simplicity, let us use examples to verify this formula. See Fig. 6.5. We have 32 vertices, 48 edges, and 16 faces in Fig. 6.5a, so $k_0 = 32$, $k_1 = 48$, and $k_2 = 16$. Thus, $b_1 = 1 - (k_0 - k_1 + k_2) = 1$ in Fig. 6.5a.

In Fig. 6.5b, $k_0 = 32 + 21$, $k_1 = 48 + 34$, and $k_2 = 16 + 12$. Therefore, $b_1 = 2$ in Fig. 6.5b.

(Then, $H_1 = Z \times \cdots \times Z = Z^{b_1}$.)

Following $\chi = \Sigma(-1)^i b_i = \Sigma(-1)^i k_i$, this can be calculated as $b_1 = h$, where h is the number of holes (Fig. 6.8).

Therefore, using the digital method, we can easily find homology groups in 2D.

The number of holes and hole counting is important to determine whether two images are similar or completely different.

Let us look at the digital case of this problem. In Fig. 6.5, we defined In_p as the total number of corner points, each of which directs to the inside of the object. Likewise, Out_p is the number of total corner points, each of which directs to the outside of the object.

The outside boundary curve always has 4 more outward points than inward points. However, each of the inside cycle has 4 more inward (corner) points than outward points. We assume that M has h holes. Then, we will have $h + 1$ cycles, one of which would be outside boundary cycle B. Therefore,

Theorem 6.1.

$$h = 1 + (In_p - Out_p)/4. \tag{6.4}$$

The formal and topological proof requires more sophisticated knowledge in topology, see [4]. To examine the correctness of the formula, we can still use the examples shown in Fig. 6.5. In Fig. 6.5a, we have $In_p = 4$ and $Out_p = 4$.

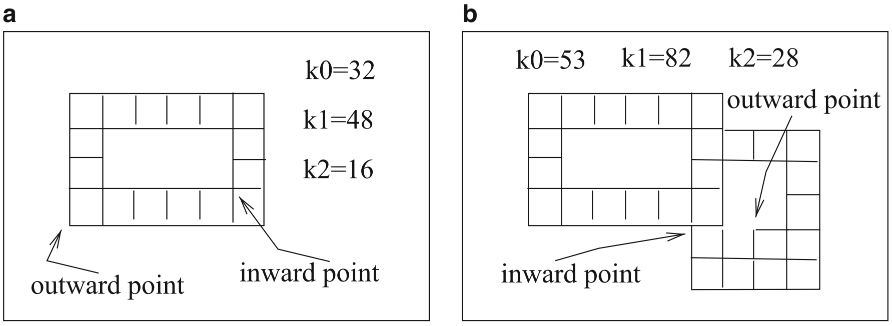


Fig. 6.8 Betti numbers and holes: (**a**), (**b**)

So $h = 1 + (In_p - Out_p)/4 = 1$. In Fig. 6.5b, $In_p = 11$ and $Out_p = 7$. So $h = 1 + (In_p - Out_p)/4 = 2$. We can see that formula (6.4) is much simpler than we just count k_0, k_1, and k_2.

In the image below, we can count the number of inward and outward points. We get $h = 1$.

6.5.2 3D Genus Computation

For details on 3D genus computation, see Chap. 14 in [4]. We will only present the formula and algorithm here [4, 5].

Theorem 6.2. *Let M be a closed 2D manifold in 3D space. The formula for genus is*

$$g = 1 + (|M_5| + 2 \cdot |M_6| - |M_3|)/8. \tag{6.5}$$

where M_i indicates the set of surface-points, each of which has i adjacent points on the surface.

This formula provides a type of topological invariants such as genus and homology groups for 3D image processing. We also design a linear time algorithm that determines such invariants for digital spaces in 3D.

Such computations have direct applications in medical imaging as they can be used to identify patterns in 3D imaging, especially for bone density calculation [4]. In [4], we discussed the implementation of the method and the applications of digital mean curvatures to 3D image classifications.

We now present a linear algorithm for finding homology groups in 3D. The actual real data calculation is shown in Figs. 6.9 and 6.10.

Algorithm 6.3. Let us assume that we have a connected set M that is a 3D digital manifold in 3D space.

Step 1. Track the boundary of M, $S = \partial M$, which is a union of several closed surfaces. This algorithm only needs to visit all the points in M to see if the point is linked to a point outside of M. This point would be on the boundary.

Step 2. Calculate the genus of each closed surface in ∂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 the formula (6.5) to obtain g.

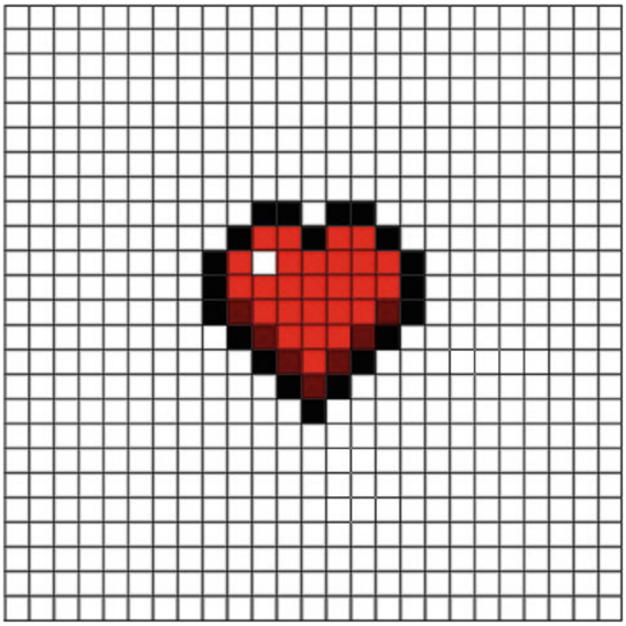


Fig. 6.9 Betti numbers and holes: $h = 1$ where $In_p = 18$ and $Out_p = 18$

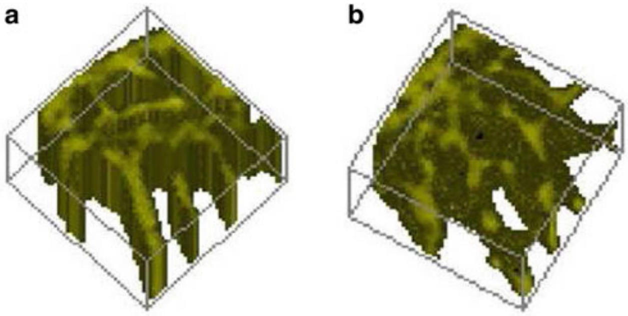


Fig. 6.10 3D genus calculation using a linear time algorithm in digital space: (**a**) g=6 and (**b**) g=10

Step 3. we also can get H_0, H_1, H_2, and H_3. H_0 is Z. For H_1, we need to get $b_1(\partial M)$, which is the summation of the genus in all of the connected components in ∂M. H_2 is the number of components in ∂M and. H_3 is trivial.

6.6 New Developments in Persistent Homology and Analysis

Persistent homology for data analysis has been studied by many researchers in mathematics and computer science, Carlsson [2], Edelsbrunner and Harer [8, 9], Ghrist [6, 7, 12], and Zomorodian [34, 36], just to name a few.

Besides playing an important role in shape analysis, image processing, and computer vision, in recent years, persistent homology has been applied to many areas of data science including biology and medical science [26, 27] and sensor networks [6, 7]. Statistical informational analysis is also used in homology [10, 11]. Persistent homology has also been applied to natural language processing [33]. Introductory articles can be found in [2, 12, 31, 33].

Some theoretical analysis using theoretical learning theory and probabilistic learning are particularly interesting to computer theorists [22, 23, 31]. Software systems developed for homology groups can be found in [19, 29]. The principle of the algorithm design can be found in [19, 20].

6.6.1 Coverage of Sensor Networks Using Persistent Homology

Sensor Networks are the popular research area in electrical engineering, computer science, and information technology [1]. A fantastic method to solving a sensor networks coverage problem was developed using persistent homology [6, 7]. This problem was first considered by Grist, Problem 6.1 in this book.

In this problem, we want to know whether or not a set of sensors can cover a given area without leaving holes. However, we do not know the exact location of the sensors. We have a set of testing locations (users who have cell phones) that will be able to tell if the user is within a certain distance ($\leq r$) to a sensor i.

This problem was first considered by Silva and Grist in [6]. Tahbaz-Salehi and A Jadbabaie proposed a solving method that uses integer programming with persistent homology [28].

They have even developed a distributed algorithm that can localize coverage holes in a network of sensors without any metric information. This means we can use a computing cloud to accomplish the job. This is exactly the right algorithm for BigData networking. The implementation under Hadoop or SPARK architecture should not be very difficult.

This algorithm uses Vietoris–Rips complexes and finds the tightest (smallest or minimal) cycle encircling a hole. Since this problem is mostly in 2D, the question we can ask is the following:

Problem 6.1. Can we use the digital method for hole finding described in the above section to solve this problem in an even faster way? What is the time complexity of the new algorithm?

6.6.2 Statistics and Machine Learning Combined with Topological Analysis

A computational and statistical learning method was developed by Niyogi, Smale, and Weinberger for noise data where the homology was concerned [22, 23]. More interestingly, their method is related to a spectral learning algorithm based on a combinatorial Laplacian (introduced in Chap. 5) that is referred to as a sampling data-based simplicial complex [25].

This work is of particular interest in the areas of theoretical computer science and discrete mathematics. However, researchers usually emphasize algorithm analysis and not practical implementations.

Another aspect of statistical methods used in topological data analysis was in [11]. They focused on the practical uses of the statistical method in shape analysis.

However, when both sides realize the advantages of the other, new and more effective methods will be developed in the near future.

The implementation is always difficult for people in computer science and information technology. Since the mathematics behind homology requires an advanced mathematical background, the methods are not easily accessible to the broader public [33].

For instance, understanding Betti numbers is much more difficult than the term number of holes. However, the concept of holes is not precise to the homeomorphism of cycles. Training data scientists in algebraic topology is an issue that must be resolved in order for them to use such great technology.

To be frank, the concept of homology groups is a hard topic even for some mathematicians. At the same time, the material and concept is one of the toughest for a professor to teach in a popular way. For instance, how do we explain why a triangle without filling in 2D has $b_1 = 1$, but a triangle with filling has $b_1 = 0$? Intuitively, we can see that the triangle without filling has a hole, but the other does not. Using the theory of homology to derive the proof would be a difficult job.

6.7 Remarks: Topological Computing and Applications in the Future

In this section, we provide an author's view to some possible applications of topological data processing in the future. Finding the topological structure of a massive data set has gained much attention in mathematics and engineering. Space data and methodological data analysis require topological solutions.

A set of individual data points (cloud data) does not have a topological structure mathematically. However, since they are sampled from the real world, we can only get a discrete set. So the human interpretation of each unique cloud data set means something different. How do we find the best interpretation that matches the original object or event [12, 34]?

We know that each point is an individual event in sampling. However, we can interpret each data point as an area or volume (usually a disk or a ball), but we would not know how big the area or volume is. When the area is very small, the object may still remain not connected, but when the coverage area of a point is very big, the total area will be connected and become one connected object. So what would the radius be such that the filled area is the closest to the real world? This is also related to "data reconstruction" since a different radius chosen is the simple learning process.

6.7.1 λ-Connectedness and Topological Data Analysis

In the persistent homology method, we make a sample point grow in its volume with a radius r. When r changes from 0 to a big number, the data will change from individual data points to a large volume until fills the entire space.

The persistent homology method calculates the homology groups (number of holes in each dimension) for each r. It would make some sense that the same topology (homology groups) that covers most of r will be the primary topological structure of M, the data set.

The minimum spanning tree(MST) that finds the minimum value for r, to make all data samples connected. This we call D_c. The value of the largest distance of the pair of points in the space M will make the set to be simply connected, denoted by D_M. The smallest value that makes the set simply connected is called D_s, so we have $D_c \leq D_s \leq D_M$.

How do we find D_s using a fast search such as divide and conquer? Another way to represent the problem of persistent analysis in λ- connectivity is to build a reduction in the following way:

Get the geometric distance for all pairs, which we do not need to calculate beforehand. We put the largest value on the site i, then we put the value $f(j) = f(i) - d(i,j)$. We can see that every point will be adjacent to other points.

We could also use MST with normalization: Let us put point i as a point we are looking at. i will easily be λ-connected to its closer points. For any point j, j will be easily λ-connected to point k if j and k are in the same circle or sphere with respect to the center at site i. This is because the potential values on j and k are determined only by the distance to i even though j and k are very far from each other. The good asset about this setting is that if j and k are not λ-connected, then j and k are not close to each other.

As we stand at site i, we can do a partition using λ-connectedness. So we calculate homology based on the partitioned data. H^0 will be the number of the component.

We can also calculate the value based on j, and we can use it to separate the classes partitioned by i. This is a refinement process with regard to the existing one that is centered at i.

The advantage is that we can build λ-connectivity for each node and pass it to a cloud computer to do a focused calculation and combine the results.

Mathematically, the results will be the same and the complexity is not increased. We can do a clustering using kNN to only focus on the center points (to build a λ-connectedness for approximation).

Such a process will save a lot of time and make the calculations possible in cloud computing [14, 15, 30]. We only consider the original data points and do not need to work on the actual filling using disks. If a component contains a hole, we can use Minkowski's sum.

Based on point locations and radius, we can use triangulation, simplex, or digital methods for space coverage. Then, we can calculate the homology group. We first want a λ-connected partition because we want to treat the smaller subsets.

Another application of λ-connectedness is to use other factors, such as the shape when D_M, the largest distance between two points in the set, is small. We can also use a hierarchy for detailed topological analysis.

A λ-connected component indicates the topology. The homology of the lambda-connected segmentation is another kind of the bar-code of persistent analysis. For image segmentation, when the value of λ changes from 0 to 1, we can obtain the homology of the partition by considering each λ-connected components, $H(\lambda)$. What is the relationship between $H(\lambda)$ and the maximum spanning tree for λ we discussed in Chap. 3?

6.7.2 Hierarchy of λ-Connectedness and Topological Data Analysis

Data has shape and shape contains meaningful information. Topological data analysis (TDA) tries to find the topology of the data, whereas λ-connectedness can attach the geometric and statistical information to topological data analysis.

Each data cluster has its own shape and pattern, but what the relationship is among them is what we try to find when we make a classification of the classified data (or treat the cluster as an element for next level classification). λ-connectedness can measure the similarity of two objects. Let H_r is a persistent homology group for radius r. Let the set of H_{r_i} to be vertices of a graph, the λ-connectedness can be implemented among those groups that represent data partitions.

Basic topology of data has meaning, but the detailed analysis would provide more than topological information. More importantly, λ-connectedness uses the information obtained from TDA that is not totally independent from the TDA.

Using λ-connectedness in topological data analysis provides an extra parameter for assisting the existing topological method.

The value of the data recombines the average location. The shape factor or pattern factor is not enough in the next level of partitioning the geometric data information on the distance metrics.

We can use λ-connectedness to combine classifications from different data sets in different data forms.

For image processing, λ-connectedness can still apply in terms of partitioning into a Big Pixel.

This can even be applied to the deformed image where the deformation is a continuous move.

6.7.3 Topological Computing in Cloud Computers

Some companies do not want to offer all the data information to outside user, they can provide some necessary information for others to evaluate the company. This just like the outlines and statistics of a company. The "door" or "gate" for exchange information. But cannot provide the whole information. In image segmentation, we may need to only hock the boundary in the quadtree (split-and-merge), there is no need to know the entire quadtree.

When a company only wants to know certain information such as boundary information or doors not inside of a building, the topological structure can be calculated based on the boundary and inside data (secret inside calculation).

Just like in quadtree analysis, we only know the boundary of lines, and each station would have to take care its own calculations.

Data sharing has limitations in that data may not be detailed or a computer station cannot store the whole volume of data. Topological data analysis would need search and matching on boundaries.

Learning in λ-connectedness includes learning the local lambda value, quadtree analysis, and the kernel method or dynamically moving kernel methods.

λ-connectedness has advantages in that it can work with both a search method with topological characteristics and a classification method related to geometric data classification.

It is a general methodology for multitasking, independent to variability in BigData, and a fast implementation method for computer programming due to Depth First Search and Breadth First Search technology.

To get the shape of a component in the data sets, we must consider other features, such as average value and shape type. In summary, connectivity can be applied to find the topological structure and λ-connectivity can be used to find classification. This classification in geometry is the shape of the object.

Learning in λ-connectedness includes learning the local lambda value, quadtree analysis, and the kernel method or dynamically moving kernel methods.

Homology can find topological structure, but we still need to find the geometric structure of data points. For this purpose, λ-connectedness is a good candidate for data science.

Topological analysis is needed to model data sets that may not fill the entire space [14]. For instance, the rainfall during a hurricane will contain a centric area that does not have a lot of precipitation. Data reconstruction should consider the topology of the data sets.

When we consider a specific problem, what features are the most important? Average value, shape, center location, or other. λ-connectedness will provide a general tool for these problems.

For any data set, with or without initial value, when a basic partition or classification is made, we can build a function related to each class. Further classification or decomposition of the original data will use this information. For instance, for the computation of homology groups in terms of persistent analysis or Morse theory, we can use the connectivity of the data on the previous partition/classification. Not only do we need the simple distance metric but we also require additional information, though the distance metric (used in most persistent methods) is important.

For instance, in circle packing, for the deformation of the data, we want to find the deformation trice. This trice will represent the customer's marketing tendencies.

The λ-connectedness method will provide more refined and detailed techniques to the modern problem related to BigData. Again, for existing techniques, the λ-connectedness method will be highly applicable to reexamine a practical problem related to data science in finding and modeling a detailed structure of the problem. It could bring us one step closer to the satisfactory or desired solution of the problem and can be used to articulate the problem.

The following example provides an explanation to this problem: (1) A company owns a large forest. In regular years, there is a fertilizer plan with designed distribution. Each small circle will give a certain amount of fertilizer for a given cost. This area contains mountains and lakes. This year, due to strong winds from a valley near the forest and a long winter, the trees need more fertilizer.

Then, the planned picture image will be combined with sensor monitoring on the ground.

We want to use the ground sensors to do the first partition, and then we want to find the deformation curve of how much fertilizer is required. How do we deal with this problem?

A standard classification method can be used to find categories. The persistent analysis can be used to find the lakes that do not need to be treated. To find the deformation curve, the λ-connectedness method can help. The data sets would still need to be monitored during the winter.

Problem 6.2. How do we calculate homology groups based on connected sum #? We know that $\chi(M \times N) = \chi(M) \cdot \chi(N)$ and $\chi(M_1 \# M_2) = \chi(M_1) + \chi(M_2) - \chi(S^n)$. How do we use the similar method in homology groups calculation if we know the homology group of parts?

Problem 6.3. For a polyhedra or polytope in very large scales, how do we decompose it to be convex hulls? This problem was discussed between Dr. David Mount in UMD and the author.

Problem 6.4. Given a cloud set (can be very large), we will want to define a region (moveable) R, find Delaunay decomposition in R. Move R to cover all Space, R can be an N-ball. How do we find the local Delaunay decomposition with the merge considering intersections (smoothly)? The solution of this question will also solve the persistent homology problem. This problem was discussed between Dr. Feng Luo in Rutgers University and the author.

References

1. H.B. Cheng, Y.D. Yao, Power adaptation for multi-hop networks with end-to-end BER requirements. IEEE Trans. Veh. Technol. **59**(7), 3545–3554 (2010)
2. G. Carlsson, Topology and data. Bull. Am. Math. Soc. **46**(2), 255–308 (2009)
3. G. Carlsson, T. Ishkhanov, V. De Silva, A. Zomorodian, On the local behavior of spaces of natural images. Int. J. Comput. Vis. **76**(1), 1–12 (2008)
4. L.M. Chen, Digital and Discrete Geometry: Theory and Algorithms, New York, Springer, 2014.
5. L. Chen, Y. Rong, Digital topological method for computing genus and the Betti numbers. Topol. Appl. **157**(12), 1931–1936 (2010)
6. V. de Silva, R. Ghrist, Coverage in sensor networks via persistent homology. Algebr. Geom. Topol. **7**, 339–358 (2007)
7. V. de Silva, R. Ghrist, Homological sensor networks. Not. Am. Math. Soc. **54**, pp. 10–17 (2007)
8. H. Edelsbrunner, J.L. Harer, Persistent homology: a survey, in surveys on discrete and computational geometry, in *Twenty Years Later: AMS-IMS-SIAM Joint Summer Research Conference, June 18–22, 2006, Snowbird, Utah 453* (American Mathematical Society, Providence, RI, 2008), p. 257
9. H. Edelsbrunner, J. Harer, *Computational Topology: An Introduction*. Applied Mathematics (American Mathematical Society, Providence, RI, 2010)
10. B.T. Fasy et al., Confidence sets for persistence diagrams. Ann. Stat. **42**(6), 2301–2339 (2014)
11. J. Gamble, G. Heo, Exploring uses of persistent homology for statistical analysis of landmark-based shape data. J. Multivar. Anal. **101**(9), 2184–2199 (2010)
12. R. Ghrist, Barcodes: the persistent topology of data. Bull. Am. Math. Soc. **45**(1), 61 (2008)
13. A. Hatcher, *Algebraic Topology*, 1st edn. (Cambridge University Press, Cambridge, MA, 2001)
14. A. Holzinger, On topological data mining, in *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*. Lecture Notes in Computer Science, Springer, vol. 8401 (2014), pp. 331–356
15. E. Kokiopoulou, J. Chen, Y. Saad, Trace optimization and eigen-problems in dimension reduction methods. Numer. Linear Algebra Appl. **18**, 565–602 (2011)
16. T.Y. Kong, Minimal non-deletable sets and minimal non-codeletable sets in binary images. Theoret. Comput. Sci. **406**, 97–118 (2008)
17. T. Lin, H. Zha, Riemannian manifold learning. IEEE Trans. Pattern Anal. Mach. Intell. **30**(5), 796–809 (2008)
18. E. Munch, M. Shapiro, J. Harer, Failure filtrations for fenced sensor networks. Int. J. Robot. Res. **31**(9), 1044–1056 (2012)
19. V. Nanda, The perseus software project for rapid computation of persistent homology. http://www.sas.upenn.edu/~vnanda/perseus/index.html, 2012
20. V. Nanda, R. Sazdanovic, Simplicial models and topological inference in biological systems, in *Discrete and Topological Models Molecular Biology*, ed. by N. Jonoska, M. Saito (Springer, Berlin, 2014), pp. 109–141
21. M. Nicolau, A.J. Levine, G. Carlsson, Topology-based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. Proc. Natl. Acad. Sci. **108**(17), 7265–7270 (2011)
22. P. Niyogi, S. Smale, S. Weinberger, Finding the homology of submanifolds with high confidence from random samples. Discrete Comput. Geom. **39**, 419–441 (2008)

23. P. Niyogi, S. Smale, S. Weinberger, A topological view of unsupervised learning from noisy data. SIAM J. Comput. **20**, 646–663 (2011)
24. B. Rieck, H. Leitte, Persistent homology for the evaluation of dimensionality reduction schemes, in *Eurographics Conference on Visualization (EuroVis) 2015*, ed. by H. Carr, K. -L. Ma, G. Santucci, vol. 34(3) (2015)
25. A. Singer, From graph to manifold Laplacian: the convergence rate. Appl. Comput. Harmon. Anal. **21**, 128–134 (2006)
26. G. Singh, F. Mmoli, G. Carlsson, Topological methods for the analysis of high-dimensional data sets and 3-D object recognition, in *Eurographics Association Symposium on Point-Based Graphics 22 (The Eurographics Association, 2007)*
27. G. Singh, F. Memoli, T. Ishkhanov, G. Sapiro, G. Carlsson, D.L. Ringach, Topological analysis of population activity in visual cortex. J. Vis. **8**(8), 1–18 (2008)
28. A. Tahbaz-Salehi, A. Jadbabaie, Distributed coverage verification in sensor networks without location information. IEEE Trans. Autom. Control **55**(8), 1837–1849 (2010)
29. A. Tausz, M. Vejdemo-Johansson, H. Adams, Javaplex: A research software package for persistent (co)homology. Software available at http://code.google.com/javaplex. (2011)
30. W. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (Springer, Berlin, 2011)
31. S. Weinberger, What is … Persistent Homology? Not. AMS 36–39 (2011)
32. K.Q. Weinberger, L.K. Saul, An introduction to nonlinear dimensionality reduction by maximum variance unfolding, in *Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI-06)*, Cambridge, MA (2006), pp. 1683–1686
33. X. Zhu, Persistent homology: an introduction and a new text representation for natural language processing, in *The 23rd International Joint Conference on Artificial Intelligence (IJCAI)* (2013), pp. 1953–1959
34. A.J. Zomorodian, Computing and comprehending topology: persistence and hierarchical Morse complexes. Ph.D. thesis, University of Illinois at Urbana-Champaign, 2001
35. A. Zomorodian, *Topology for Computing* (Cambridge University Press, Cambridge, MA, 2005)
36. A. Zomorodian, G. Carlsson, Computing persistent homology. Discrete Comput. Geom. **33**(2), 249–274 (2005)