UNIVERSITAT DE BARCELONA

**ADVANCED MATHEMATICS**
**MASTER'S FINAL PROJECT**

---

# Extended persistence and duality in cubical complexes

---

*Author:*
Junhan Cui

*Supervisor:*
Carles Casacuberta Vergés

**Facultat de Matemàtiques i Informàtica**

June 28, 2024

# Contents

## Abstract

In this work, I will continue to investigate the techniques of topological data analysis based on my bachelor thesis [4] and apply it to some image-type data sets. The tools and analysis methods we used are homology, cohomology, cubical complexes, persistent homology, barcode, extended persistence, super- and sub-level set, total persistence, persistence entropy, etc.

Our goal is to try to construct the cubical complex of the image through the above methods, and then study the topological features of two-dimensional and three-dimensional grayscale images, such as total persistence and persistence entropy. And try to combine it with statistical methods to determine the attribution of the image.

We found some kind of duality in the bachelor thesis of Marina Anguas [1]. She performed sub-level set filtration from bottom to top and super-level set from top to bottom on a 3D image and calculated extended persistence. Then she found some pairs of results they were similar. So we hope to study the relationship between cycles of different dimensions such as $H_0$, $H_1$ and $H_2$ by studying a duality analogous to Poincaré duality or Lefschetz duality, and simplify the calculation through the relationship between them.



**Keywords:** Algebraic topology, Homology, Cohomology, Cubical complexes, Extended persistence, Duality, Digital image.

2020 Mathematics Subject Classification. 55N31, 62R40, 68T09

# Chapter 1

# Introduction

## 1.1 Introduction of topological data analysis

Before we get into the main body of this article, we would like to introduce some basic concepts about topological data analysis (TDA). Topological data analysis is a discipline that uses topological tools to analyze and understand the structure of data. TDA focuses on the shape and spatial structure of data, especially the high-dimensional characteristics and overall structure of data, rather than relying solely on local features of traditional statistics.

TDA constructs simplicial complexes (or other complexes, such as singular complexes and cubical complexes, which are used in this article) associated with the data and infers qualitative features of the set from the homology of the complexes. These features can quantify the complex topological shapes and geometric structures in the data to answer questions about the data set. The data is usually represented as a point cloud in Euclidean or more general metric space. We will use a common topological tool, persistence graph, to reflect the connectivity of data points to represent loops and holes in space.

TDA is a new field that emerged in the 2000s from various works in algebraic topology and computational geometry. Although the history of data analysis via geometric methods goes back a long way, TDA was actually started by Edelsbrunner et al. [5] in the field of topological persistence.

This marked the real beginning of TDA. Later persistent homology of Zomorodian and Carlsson [10] made TDA a very powerful technique.

Although the underlying principles of topological data analysis are not easy to understand, thanks to the existing topological data analysis codes in various programming languages, even novices can complete the analysis. This work uses the ultra-fast C++ Ripser package as the core computing engine, and uses a module named Cripser.py developed based on Python's built-in Ripser.py module to

implement the analysis of cubical complexes.

The Python program used later in this article (i.e., Cripser.py) was developed by Takeki Sudo and Kazushi Ahara from Meiji University in 2018, and modified by Shizuo Kaji from Kyushu University in 2019. The paper related to this program is [7].

Next we will briefly introduce the basic concepts of TDA. For a more detailed introduction, please refer to Chapter 3 of my bachelor thesis [4].

## 1.2 Basic concepts of TDA

First of all, we need to define some basic concepts in algebraic topology as well as related theorems and conclusions.

We divide these concepts into two parts:

1. Simplicial complexes, cubical complexes and their filtrations.
2. Homology groups.

According to the definition, in affine space the difference between two points is a vector, and the addition of a point and a vector yields another point, although addition between points cannot be done.

From this definition, we can deduce the following fact. There are $n+1$ affinely independent points in a $k$-dimensional Euclidean space ($k \geq n$) if and only if there is no $(n-1)$-dimensional hyperplane that contains $n+1$ points. The hyperplane that contains $n+1$ points needs to be at least $n$-dimensional.

### 1.2.1 Simplicial complexes and cubical complexes

**Definition 1.1.** An *n-simplex* is a $n$-dimensional polytope which is the convex hull of $n+1$ points in a Euclidean space $\mathbb{R}^d$, called *vertices*. Thus, an $n$-simplex is a subset of $\mathbb{R}^d$ of the following form:

$$C = \{\theta_0 P_0 + \cdots + \theta_n P_n \mid \sum_{i=0}^{k} \theta_i = 1,\ \theta_i \geq 0,\ i = 0, \ldots, n\},$$

where $\{P_0, \ldots, P_n\}$ are $n+1$ affinely independent points.

Thus, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, a 3-simplex is a solid tetrahedron, etc.

**Definition 1.2.** The convex hull of any nonempty subset of the vertices of an $n$-simplex is called a *face* of the simplex. Faces are also simplices themselves.

**Example 1.3.** The faces of a 3-simplex (tetrahedron) are its triangular faces, which can be viewed as 2-simplices. The faces of a 2-simplex (triangle) are its sides. The faces of a 1-simplex are its vertices.

If we wish to study more complex structures, it is not enough to rely on one simplex. We want to study the graphics formed by simplices according to certain rules. And those combinations of simplices ar called simplicial complexes. In mathematics, a simplicial complex is a set composed of points, line segments, triangles, and their $n$-dimensional analogues.

**Definition 1.4.** A *cubical complex* (also called cubical set and Cartesian complex) is a set composed of points, line segments, squares, cubes, and their $n$-dimensional counterparts.

Hence, a cubical complex is a collection of multiple "$n$-dimensional cubes" connected by faces, edges, and vertices. Each "cube" can be of any dimension, such as a point (0 dimension), a line segment (1 dimension), a square (2 dimension), a cube (3 dimension), etc.

We can understand simplicial complexes as a combination of $n$-dimensional "triangles" and cubical complexes as a combination of $n$-dimensional "cubes".

Cubical complexes are used analogously to simplicial complexes in the computation of the homology of topological spaces.

**Definition 1.5.** A *simplicial complex* $K$ is a set of simplices that satisfies the following conditions:

1. Every face of a simplex from $K$ is also an element (a simplex) in $K$.
2. Each non-empty intersection of any two simplices $\sigma_1, \sigma_2 \in K$ is a face of both $\sigma_1$ and $\sigma_2$.

Note that the common faces of cubical complexes are vertices or sides of cubes.

The reason we define a simplicial complex is that we need to create a simplicial complex based on a data cloud. Each data point in a data point cloud is usually considered as a vertex, so we can consider the data point cloud as a set of vertices.

Similarly, the reason we define a cubical complex is that we need to create a cubical complex based on image-type data. We can transform a binary image (consisting of white and black) into a numerical matrix with grayscale number in each element of this matrix. Obviously, if we need to study the topological features of a digital image, the best way is not to consider each pixel (or voxel) as a vertex in a point cloud, but to consider each pixel (or voxel) and consider it as a different element according to different construction methods, which may be a vertex or a square.

We want to create cubical complexes relating voxels of the images. The idea is to consider subsets of the numerical matrix based on the image and then find out the possible topological structures of cubical complexes.

### 1.2.2 Filtration

To study the possible changes of a cubic complex throughout the creation from a voxel to the entire image, we need to use a concept in set theory which is a *filtration*.

**Definition 1.6.** Given a cubical complex $C$, a *filtration* about $C$ is a family of indexed subcomplexes $F = \{C_i \subseteq C\}_{i \in I}$, where $F$ is indexed on an ordered set $I$ such that if $i \leq j$ then $C_i \subseteq C_j$ and also $\varnothing, C \in F$.

There exist $i_0, i_1, \ldots, i_{n-1}, i_n \in I$, $i_0 \leq i_1 \leq \cdots \leq i_{n-1} \leq i_n$ such that

$$\varnothing = C_{i_0} \subseteq C_{i_1} \subseteq \cdots C_{i_{n-1}} \subseteq C_{i_n} = C.$$

For example, a filtration of a cubical complex can be based on the value of grayscale of each voxel of an image, so that every $C_{i_n}$ is the set which contains all elements with grayscale values less than or equal to $i_n$.

**Example 1.7.** Suppose that we have a $2 \times 2$ photo that we can transform into a grayscale matrix. If we build the filtration step by step from small to large numbers in the matrix, we will get this filtration. In the first step, we only consider the elements with value 1 and color them; in the second step, we consider the numbers after that, that is, the elements with value 4, and finally we consider larger values, that is, color the grid with value 7.

$$\begin{pmatrix} 1 & 4 \\ 4 & 7 \end{pmatrix}$$

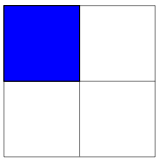


Figure 1.1: Step of the value = 1

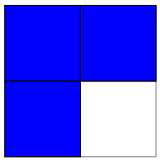


Figure 1.2: Step of the value = 4

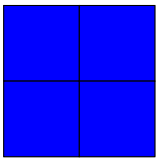


Figure 1.3: Step of the value = 7

If we construct the filtration from the other direction, then we start from the biggest element to the smallest element.

### 1.2.3 Homology group and $n$-cells

Algebraic topology has two important tools: Homotopy and Homology. We explain a little bit these concepts. When two continuous functions from one topological space to another are called homotopic if one can be "continuously changed" into the other, such a deformation will be called a *homotopy* between these two functions. While the existence of homotopies is difficult to determine in higher dimensions, there is a more computable alternative than homotopy groups: the homology groups $H_n(X)$.

Before defining homology groups, it is necessary to know some preliminary concepts such as $n$-cells, $CW$-complexes and boundary functions.

For the part of CW-complexes and $n$-cells, we can see the theory in [6]. We are not going to explain it here, because CW-complexes are not the core part in this work. We just need it to define homology groups.

An *n-cell* is a space homeomorphic to an $n$-ball in a Euclidean space. Hence, a 0-cell is a point; a 1-cell is an open arc; a 2-cell is an open disk, etc.

*Attaching an n-cell* to a space $X$ consists of gluing a closed $n$-ball $D^n$ to $X$ by means of a continuous map $f\colon \partial D^n \to X$, by identifying each point $z \in \partial D^n$ with its image $f(z) \in X$.

A complex formed by attaching $n$-cells for various values of $n$ is called a *cell complex* or *CW-complex*.

**Example 1.8.** If we attach a 1-cell to a 0-cell, we obtain a circle. If we attach a 2-cell to a 0-cell, we obtain a sphere (the surface of a hollow ball) because we glued the boundary of a 2-cell with the 0-cell. These are examples of CW-complexes.

Given a CW-complex $X$, we denote by $C_n(X)$, or by $C_n$ for shortness, the free abelian group with basis the set of $n$-cells of $X$. There is a group homomorphism (called *boundary*) $\partial_i\colon C_i \to C_{i-1}$ which sends $i$-cells to sums of $(i-1)$-cells using the information given by the attaching map of each $i$-cell; see [6] for details.

We call this homomorphism a *boundary* because $\partial_i$ sends an $i$-cell to a sum of $(i-1)$-cells which form the boundary of the given $i$-cell.

There is a sequence of homomorphisms as follows:

$$\cdots C_i \xrightarrow{\partial_i} C_{i-1} \xrightarrow{\partial_{i-1}} \cdots \longrightarrow C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0.$$

It is shown in [6] that $\partial_i \circ \partial_{i+1} = 0$ for all $i$. Therefore, $Im\,\partial_{i+1} \subseteq Ker\,\partial_i$ for all $i$.

For example, $Ker\,\partial_0$ contains all vertices because we cannot find lower dimensional elements than them; $Ker\,\partial_1$ is generated by "cycles" formed by concatenation of 1-cells (edges); $Ker\,\partial_2$ is generated by "cycles" formed by assembling 2-cells into surfaces that enclose cavities, and so on. In general, $Ker\,\partial_i$ is generated by

"cycles" formed by assembling $i$-cells which wrap (in any dimension) an $(i+1)$-dimensional void space.

So there is a difference between $Ker\,\partial_i$ and $Im\,\partial_{i+1}$. In $Ker\,\partial_i$ we find closed concatenations of $i$-cells and it does not matter if such a concatenation wraps something of higher dimension or not. But in $Im\,\partial_{i+1}$ we encounter concatenations which do have cells of higher dimension inside. In this sense, we visualize the fact that $Im\,\partial_{i+1}$ is contained in $Ker\,\partial_i$.

Now we have introduced all we need to define homology groups.

**Definition 1.9.** The *n-dimensional homology group* $H_n(X)$ of a CW-complex $X$ is a quotient group defined as

$$H_n(X) := \ Ker\,\partial_n / Im\,\partial_{n+1}.$$

From this definition we can observe that elements of a homology group $H_n$ can be viewed as concatenations of $n$-cells that do not wrap anything inside. And the concatenations of $n$-cells which wrap some $(n+1)$-cell have been eliminated by the quotient group operation.

In the following summary, we explain the interpretation of $H_n$ from the previous definition 1.9.

1. The 0-dimensional holes ($H_0$) are the connected components.
2. The 1-dimensional holes ($H_1$) are the holes formed by linear combinations of 1-cells. Each such hole is a cycle which is formed by a concatenation of edges (1-cells).
3. The 2-dimensional holes ($H_2$) are the holes formed by linear combinations of 2-cells. The interior of a hollow sphere is the most common example.

### 1.2.4 Persistent homology

In this section, we define the core conception of TDA by referring to the article [5] by Edelsbrunner et al.

As we have already defined *filtration* in Definition 1.6, we have $\varnothing = C_{i_0} \subseteq C_{i_1} \subseteq \cdots C_{n_1} \subseteq C_n = C$, we apply the homology functor, which for each space gives a vector space and for each inclusion gives a linear map:

$$0 = H(C_0) \to H(C_1) \to \cdots \to H(C_n) = H(C)$$

referring to this sequence as a *persistence module*.

As we defined above in 1.9 $H_n$ is the $n$-th homology group with $n$ the dimension. We assume coefficients in a field $F$, so that $H_n = F \bigoplus F \bigoplus \cdots \bigoplus F = F^{\beta_n}$ is

a vector space over $F$, with $\beta_n$ = rank $H_n$ known as the $n$-th *Betti number*. (The $n$-th Betti number refers to the number of $n$-dimensional holes on a topological surface.)

It is instructive to split the module into *indecomposable summands* of the form

$$0 \to F \to \cdots \to F \to 0.$$

There is a unique such decomposition whose direct sum gives the original module. Each summand can be interpreted as the *birth* of a homology class at its first non-zero term and the *death* of the same class right after its last non-zero term.

In other words, we can define it using mathematical language more precisely.

**Definition 1.10.** Given a simplicial complex $C$ with $F_C = \{C_i \subseteq C\}_{i \in I}$, every $h \in H_n(C_i)$ is an $n$-dimensional hole of the homology group in the subspace $C_i$.

The *birth time* of hole $h$ is the first time $j$ in which $h$ appears as an $n$-dimensional hole. We define the homomorphism $f_{i,j} : H(C_i) \to H(C_j)$ so that

$$T_{birth}(h) := inf\{j \in I \mid h \in Imf_{j,i}\}.$$

The *death time* of hole $h$ is the first time $j$ in which $h$ disappears and no longer is an $n$-dimensional hole, so that

$$T_{death}(h) := inf\{j \in I \mid h \notin Imf_{i,j}\}.$$

Hence the *persistent homology* is the homology where the *persistence* of a hole refers to the time between its birth and its death. And we use persistence to assess the variation of holes along the process of the related filtration.

For cubical complex, it will be convenient to define creator and destroy cells to locate the cycles.

**Definition 1.11.** The *creator* of a cycle is the unit that produces the cycle. And the *destroyer* of a cycle is the unit that destroy the cycle.

For example, the voxel of a $n$-dimensional image with the lowest filter value in a connected component creates a 0-dimensional cycle, while a voxel connecting two independent connected components destroys the component with a higher birth time.

The creator and destroyer units are not uniquely identified, as it is possible for multiple destroyer units to destroy the same cycle at the same time, but they still provide useful information to locate cycles.

**Example 1.12.** If we have a 2-dimensional image and we transform it into a matrix of their grayscale values. Then we can observe the filtration from the lowest value to highest value.

$$\begin{pmatrix} 1 & 4 & 9 \\ 4 & 7 & 3 \\ 6 & 8 & 5 \end{pmatrix}$$

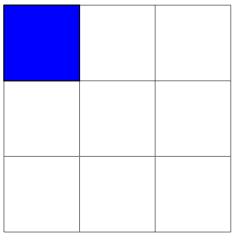


Figure 1.4: Step of the value = 1

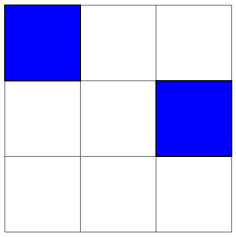


Figure 1.5: Step of the value = 3

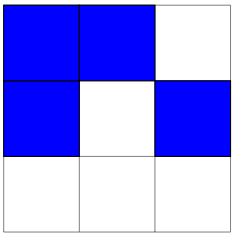


Figure 1.6: Step of the value = 4

We observe that in the first step of the filtration when we filled the element with value = 1 and in the second step we filled the element with value = 3. Until here we have two separated connected components, i.e. two $H_0$. And in the third step we filled the element with value = 4 and the connected component created by the element of in the top left corner and the other created by the right block they both have been destroyed because of the generation of two blocks filled in the third step. So we can call these two blocks with value = 4 are destroyer cells of these two $H_0$ generated until the second step.

### 1.2.5 Total persistence

One of the disadvantages of TDA is the difficulty to interpret and explain the meaning of the results we obtained. For example, once we made the persistence diagram, we cannot find out a conclusion directly from this diagram.

So we need a method to quantify the diagram with a certain number that allows us to do a numerical analysis and what we found is a intrinsic topological property called *total persistence*. We used the concept defined in the work of M. Rucco [8].

**Definition 1.13.** *Total persistence* is a numerical property of the persistence diagram defined by

$$Total\ Persistence\ := \sum_{i \in I} l_i \qquad with\ l_i = death_i - birth_i,$$

where $I$ is the set of hole indices.

So the total persistence is simply the sum of the persistence time of all holes once generated during the process of filtration. It is obviously a real positive number. So we can consider it as a numerical descriptor of a persistence diagram.

### 1.2.6 Persistence entropy

In information theory, the entropy of a random variable is the average level of "information" inherent in the variable's possible outcomes. And this average information level is determined according to the unexpectedness and uncertainty of the event.

We need to notice that events with a smaller probability will provide more information entropy, because an ordinary event will not make people feel surprised, nor will it make people feel that something special has happened. And once an unusual event occurs, it is natural to notice possible changes, and this provides us with more information.

According to this idea, Claude Shannon defined the information entropy in his paper 'A mathematical theory of communication' [9] as a reasonable measure of information contained.

**Definition 1.14.** Given a discrete random variable $X$ with the distribution according to $p : X \to [0, 1]$ (i.e., the probability), the *Information entropy* is defined as

$$H(X) := -\sum_{x \in X} p(x) \log(p(x)) = \mathbb{E}[-\log(p(X))].$$

The base of logarithm can be defined depending on the situation we study.

By the definition of information entropy, we can see that the event with lower probability will give a greater information.

Scholars who use TDA for data analysis are inspired by the definition of information entropy (defined in 1.14), thus defining a new entropy based on total persistence and persistence of holes (defined in 1.10) to measure the amount of topological information. This entropy is called *persistence entropy* and we will use the definition in the work of M. Rucco [8].

In the case of persistent topology, we define the probability of $n$-hole.

**Definition 1.15.** The parameter for the distribution is defined as:

$$p_i := \frac{death_i - birth_i}{Total\ persistence}.$$

Before the definition of persistence entropy, we would like to see that this parameter we defined above is a probability.

It is easy to see that:

(1) $p_i \geq 0$ because the death time will never be earlier than the birth time.

(2) $\sum_{i \in I} p_i = \frac{\textit{Total persistence}}{\textit{Total persistence}} = 1$.

(3) $P(\bigcup_{i=1}^{n} h_i) \ = \ P(h_1 \bigcup \cdots \bigcup h_n) \ = \ P(h_1) + \cdots + P(h_n) \ = \ \sum_{i=1}^{n} P(h_i)$.

**Definition 1.16.** *Persistence entropy* is defined as

$$\textit{Persistence entropy}(X) := -\sum_{x \in X} p(x) \log_2(p(x)) = \mathbb{E}[-\log_2(p(X))],$$

where $X$ is a set of holes exist in a certain persistence diagram and $x \in X$ are the holes. The logarithm is with base 2: $\log_2$.

## 1.3 Purpose of this work

The advantage of TDA is that it can effectively process high-dimensional and complex data, capture the global topological structure in the data, and provide a deep understanding of the shape of the data. However, compared with traditional statistical methods, TDA has some disadvantages, that is, the computational complexity is too high, especially for the computation of $H_2$ or $H_n$ the homology groups with higher dimensions. The processing of large-scale data sets may require efficient algorithms and computing resources, and the interpretation of topological features may require professional knowledge.

As we have said in the abstract, we found some relations between the numerical values of the total persistence time of $H_0$, $H_1$, $H_2$. So we want to study the possible dual relations between the homology groups with different dimensions.

In the practical usage of TDA, we usually calculate the homology groups with dimensions less than 3, vertex $H_0$, cycle $H_1$, cavity $H_2$. Because they are the most easy to imagine and to understand. And in the most cases, these $H_n$ are sufficiently to study the variation of topological structure of an object.

The problem is the complexity of computation of $H_2$. It is natural that the difficulty to compute the $H_n$ will be raised as the dimension raised. To compute $H_0$, we just need to count the number of connected components, that is the simplest case. To compute $H_1$, we need to do a little bit more works to count the cycles formed. And to compute $H_2$, we need to consider the 3-dimensional space and it will cause very high computational cost. Generally, we cannot do the analysis just using $H_0$ and $H_1$ especially for 3-dimensional objects. Although the computation for $H_2$ will be expensive, it is necessarily to be analyzed.

The main purpose of this work is finding a way using duality relations between $H_0$, $H_1$, $H_2$ to compute in a easier way the homology with higher dimension. For example, we can calculate $H_2$ as easy as calculate $H_0$ using some duality

between them. This will improve a lot the efficiency of algorithm. And we can also generalize the case to higher dimensions to calculate $H_i$ as easy as calculate $H_j$ with $j \leq i$.

And one of the other objective with less importance is the discriminant classification of images using their topological features.

# Chapter 2

# Extended persistence for manifolds

Before we start to study cubic complexes, we need to understand the concept of extended persistence for simplicial complexes. Because we need extended persistence to use duality to more easily calculate homology groups in different dimensions.

Generally speaking, ordinary TDA does not use extended persistence, but only uses filtration built from one direction to study the possible changes in the topological structure of the object when the filtration step changes. For example, if we cut a three-dimensional object through countless cross sections, and then we build subsets of this object little by little from the bottom up, we can get the results we need.

It should be noted that in actual calculations, there will be two different filtrations from top to bottom or from bottom to top, which will lead to completely different results. Although there is a certain duality between the filtrations built in these two directions, it does not mean that we can directly regard these two directions as the same.

## 2.1 Extended persistence

First, let us introduce extended persistence for simplicial complexes (it should be noted that although cubical complexes also have their corresponding extended persistence, the two are not connected, but their ideas have certain similarities).

**Example 2.1.** Here we use a picture that is often used when introducing extended persistence. This picture is a smooth closed manifold without boundaries embedded in three-dimensional space, with an axis describing the height attached to the

side, on which the positions of all critical points are marked.

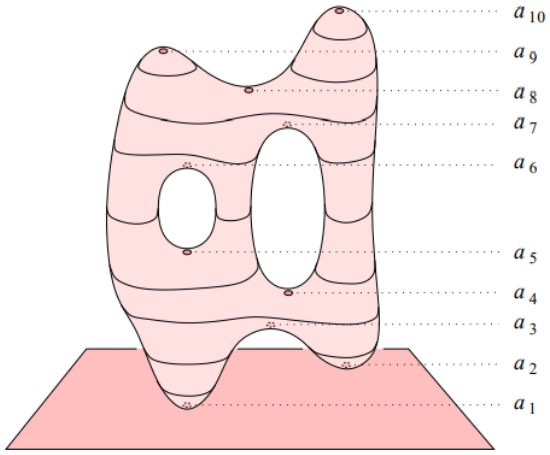
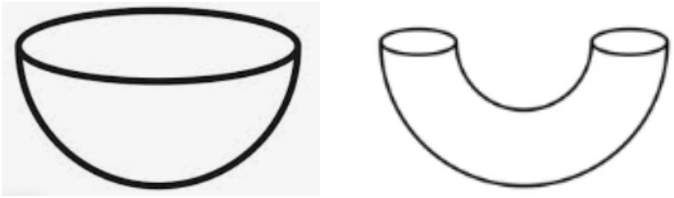



Figure 2.1: A closed smooth manifold.

If we decompose this image into countless cross-sections and study the shapes of each cross-section, we will find that the shape of the cross-section is different every time we encounter a critical point. To be more precise, the shapes of the cross-section are not homeomorphic in different intervals formed by critical points.

For example, the cross section in the interval $(a_1, a_2)$ is a circle on a plane. After passing the critical point $a_2$, when entering the interval $(a_2, a_3)$, the cross section becomes two unconnected circles on a plane. Obviously, the shape of this cross section is not isomorphic to the previous one. After passing the point $a_3$, the two unconnected circles on the cross section merge into a larger circle.

In addition, if we consider not just the cross section, but the entire part below point $a_4$, then we will find that this part (subset) is homeomorphic to a hemisphere. If we pass through point $a_4$ and are in the interval $(a_4, a_5)$, then the subset consisting of all parts below point $a_5$ is not homeomorphic to a hemisphere, but to a semi-torus.



(a) Half-sphere (b) Half-torus

Figure 2.2: Subspaces that are not homeomorphic.

And if we start from the critical point $a_{10}$ and do the same in the other direction from up to down, then we can have different results than the previous one. This is an example of extended persistence, "extended" means another way.

## 2.2 Morse functions and critical points

We first define Morse functions on smooth manifolds.

**Definition 2.2.** For a smooth function $f\colon \mathbb{M} \to \mathbb{R}$, where $\mathbb{M}$ is a smooth manifold, a *critical point* of $f$ is a point $p \in \mathbb{M}$ such that $\frac{\partial f}{\partial x_i}(p) = 0$ for all $i = 1, \ldots, n$, where $(x_1, \ldots, x_n)$ are local coordinates in any chart around $p$.

If $p$ is a critical point of $f$, then $f(p)$ is a *critical value*.

**Definition 2.3.** A smooth function $f\colon \mathbb{M} \to \mathbb{R}$ defined on a smooth manifold $\mathbb{M}$ is called a *Morse function* if its Hessian matrix $\frac{\partial^2 f}{\partial x_i \partial x_j}$ is non-degenerate at every critical point $p \in \mathbb{M}$.

Morse functions were originally used to describe the geographical environment in the real world. Given a location, the function value is equal to the altitude at this point. The critical points are the top of the mountain, the bottom of the valley, and other places in the real world (i.e., the saddle point). Generally speaking, the graph of this type of function is usually accompanied by contour lines, just like the picture in the previous example.

As the Hessian matrix of a Morse function is non-degenerate at critical points, this implies that critical points of this type of function are isolated and therefore there are only finitely many critical points, assuming that $\mathbb{M}$ is compact.

Based on Morse functions, we can now define the sub-level sets and super-level sets which will be used in the extended filtration in both directions.

**Definition 2.4.** For a Morse function $f\colon \mathbb{M} \to \mathbb{R}$, denote, for each $h \in \mathbb{R}$,

1. $L_h(f) = \{x \in \mathbb{M} \mid f(x) \leq h\}$. Note that $L_s(f) \subseteq L_t(f)$ if $s \leq t$. These are called *sub-level sets* of $f$.

   We have that $L_h(f) = \emptyset$ if $h < inf(f)$ and $L_h(f) = \mathbb{M}$ if $h \geq sup(f)$.

2. $L^h(f) = \{x \in \mathbb{M} \mid f(x) \geq h\}$. Note that $L^t(f) \subseteq L^s(f)$ if $s \leq t$. These are called *super-level sets* of $f$.

   We have that $L^h(f) = \mathbb{M}$ if $h < inf(f)$ and $L^h(f) = \emptyset$ if $h \geq sup(f)$.

Sub-level sets will be used in the case of the filtration from bottom to top, and the super-level sets will be used in the case of the filtration from top to bottom.

Based on these concepts, we can construct a chain complex of homomorphisms using sub-level sets

$$0 = H_m(L_{i_0}) \longrightarrow H_m(L_{i_1}) \longrightarrow \cdots \longrightarrow H_m(L_{i_n}) = H_m(\mathbb{M})$$

for interleaved values $i_0 < \cdots < i_n$ and each homological dimension $m$. The *interleaved values* are the values between the extremes of intervals formed by the critical points of the manifold, that means these $\{i_0, \ldots, i_n\}$ have no element equal to some critical points. Then we do not need to consider the case if we reached exactly the thresholds.

Note that we are now constructing the filtration from bottom to top, using sub-level sets to construct each subset that needs to calculate the homology group. Obviously, we cannot add the previously defined boundary functions on the arrow, because boundary functions only allow the input of high-dimensional space to be mapped to a lower-dimensional space, so we are just building a chain between homology groups, which is irrelevant to boundary functions. Obviously, $i_0$ must be a value less than inf(f), and $i_n$ must be a value greater than sup(f). According to the previous definition, we can traverse all possible homology groups and their variations from the empty set to the entire $\mathbb{M}$ in this way.

## 2.3 Cohomology and extended filtration

A natural problem is if we can construct from up to bottom using super-level sets. We can do that but to study the extended persistence and filtration in both directions, we need to introduce the concept of *cohomology*.

In homology theory and algebraic topology, cohomology is a general term for a sequence of abelian groups, usually associated with a topological space, often defined from a *cochain complex* (we will define it later). Cohomology can be viewed as a method of assigning richer algebraic invariants to a space than homology. Some versions of cohomology arise by dualizing the construction of homology; actually, we construct it in this work dualizing the homology and also dualizing sub-level sets.

### 2.3.1 Singular cohomology

Singular cohomology is a powerful invariant in topology that relates commutative rings to arbitrary topological spaces. Every continuous mapping $f : X \to Y$ determines a homomorphism from the cohomology ring of Y to the cohomology ring of X; this imposes strict restrictions on the possible mappings from $X$ to $Y$.

Unlike topological invariants such as homotopy groups, cohomology rings are often computable in practice for the space of interest and this is also the reason why we use homology instead of homotopy to investigate their topological features.

For a topological space $X$, singular cohomology is based on the *singular chain complex:*

$$\cdots C_i \xrightarrow{\partial_i} C_{i-1} \xrightarrow{\partial_{i-1}} \cdots \longrightarrow C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0.$$

Here $C_i$ is the free abelian group on the set of continuous maps from the standard $i$-simplex into $X$, and $\partial_i$ is the corresponding $i$-th boundary homomorphism. The groups $C_i$ are zero for $i$ negative.

Now fix an abelian group $A$ and replace each free abelian group $C_i$ by its dual group defined as $C^i := Hom(C_i, A)$ where, $Hom$ denotes the set of all possible group homomorphisms from $C_i$ to $A$ and it is also an abelian group.

On the other side, we also need to replace the boundary homomorphisms $\partial_i$ by their dual homomorphisms $\partial^i \colon C^i \to C^{i+1}$. Since we reversed the arrows of the boundary homomorphisms, we can also reverse all arrows of the previous chain complex and we obtain a *cochain complex:*

$$\cdots C^i \xleftarrow{\partial^i} C^{i-1} \xleftarrow{\partial^{i-1}} \cdots \longleftarrow C^2 \xleftarrow{\partial^2} C^1 \xleftarrow{\partial^1} C^0.$$

For each integer $i$, the $i$-th *cohomology group* of $X$ with coefficients in $A$ is defined as

$$H^i(X; A) := Ker\, \partial^i / Im\, \partial^{i-1}.$$

The group $H^i(X; A)$ is zero for $i$ negative and the elements of $C^i$ are called *singular i-cochains* with coefficients in the group $A$.

As we reversed all the arrows in the sequence of homomorphisms and added a "co" prefix to indicate the other direction, we can also call the elements of $Ker\, \partial$ *cocycles* and those of $Im\, \partial$ *coboundaries*.

### 2.3.2 Extended filtration

Let $a_1 < a_2 < \cdots < a_n$ be the homological critical values of the Morse function $f : \mathcal{M} \to \mathbb{R}$ as defined before. From the interleaved values $i_0 < i_1 < \cdots < i_n$ we saw before contained in the interior of intervals formed by the critical values as extremes. We can get sub-level sets $L_{i_j} = f^{-1}(-\infty, i_j]$ which are 2-manifolds (figure 2.1) with boundary (as they are subsets of whole manifold without boundary). Symmetrically, we define super-level sets $L^{i_j} = [i_j, \infty)$ which are also 2-manifolds with boundary. That means we define the free abelian groups $A$ in the chain of cohomology as we saw previously.

In fact, we are now constructing the cohomology using one of all versions that is dualizing the construction of homology, dualize the sub-level set to super-level

set, dualize the direction up to down to the reverse direction bottom to up, and dualize homology group to cohomology group.

Then we can construct a sequence (with $p$ fixed) of homology groups going up from bottom and a sequence of relative homology groups coming back down from the top:

$$0 = H_p(L_{i_0}) \to H_p(L_{i_1}) \to \cdots \to H_p(L_{i_n})$$
$$= H_p(\mathbb{M}, L^{i_n}) \to H_p(\mathbb{M}, L^{i_{n-1}}) \to \cdots \to H_p(\mathbb{M}, L^{i_0}) = 0.$$

where relative homology groups are defined as $H_n(X, Y) := Ker\,\partial'_n / Im\,\partial'_{n+1}$ with $\partial'_n \colon C_n(X)/C_n(Y) \to C_{n-1}(X)/C_{n-1}(Y)$ with $Y$ some subspace of $X$.

We know that for modulo 2 arithmetic the homology groups are isomorphic to the cohomology groups in complementary dimensions, so that Lefschetz duality implies that

$$H^p(\mathbb{M}) \cong H_{d-p}(\mathbb{M}, L^i)$$

where $d$ is the dimension of $\mathbb{M}$. This result suggests the idea to prove the duality in chapter 4 for cubical complexes.

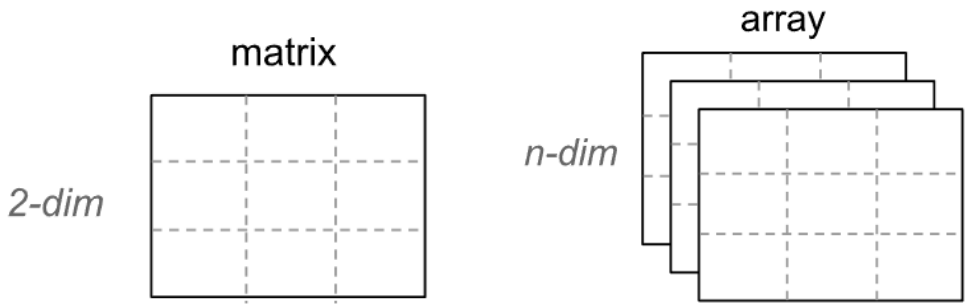
# Chapter 3

# Construction for cubical complexes

Obviously, simplicial complexes is different with cubical complexes. Simplicial complexes consist of combinations and concatenations of $n$-dimensional simplices as we defined in 1.1. They are points, segments, triangles, tetrahedra and their $n$-dimensional counterparts. Cubical complexes consist of combinations and concatenations of points, line segments, squares, cubes, and their $n$-dimensional counterparts.

Visual shape is not the most important difference between them, there are something more such as simplicial complex triangulate object and usually be used in the case that investigate a smooth manifold, that means the object is continuous. And cubical complex will usually be used in the case that investigate a object that can be transformed into matrix or multidimensional array of numbers, that means they can be considered as a discrete object.

In this work, we will focus on cubical complexes. This is because our research object this time is image-type files, such as two-dimensional or three-dimensional electronic scanned images. The study of image types is not the same as the study of data point clouds. Because data point clouds can usually form a manifold, and two-dimensional images are regarded as (or converted into) a numerical matrix composed of the multiplication of the length and width of the image (calculated in pixels). Three-dimensional images are regarded as a collection of countless cross-sectional slices. If each slice is regarded as a two-dimensional image (that is, a numerical matrix), then a three-dimensional image is a multidimensional numerical array similar to a Rubik Cube as the following figures.




(a) 2D-image (b) 3D-image

Figure 3.1: Transformation of images into numerical type data

Obviously, cubical complex is more appropriate to the image-type data.

One more detail about the transformation of images into numerical data is that we only consider the grayscale images because in such case we only need to save one numerical value which represents the magnitude of grayscale in the place of a voxel. But for coloured images, there will be more values to save.

## 3.1 Cubical complexes

**Definition 3.1.** An *elementary interval* is a subset $I \subset \mathbb{R}$ of the form

$$I = [l, l+1] \quad or \quad I = [l, l]$$

for some $l \in \mathbb{Z}$. An *elementary cube* $Q$ is the finite product of elementary intervals, i.e.,

$$Q = I_1 \times I_2 \times \cdots \times I_n \subset \mathbb{R}^n$$

where $I_1, I_2, \ldots, I_d$ are elementary intervals. Equivalently, an elementary cube is any translate of a unit cube $[0,1]^n$ embedded in Euclidean space $\mathbb{R}^d$ (for some $n, d \in \mathbb{N} \cup \{0\}$ with $n \leq d$). A set $X \subseteq \mathbb{R}^d$ is a *cubical complex* (or *cubical set*) if it can be written as a union of elementary cubes (or possibly, is homeomorphic to such a set).

**Example 3.2.** Every vertex is a unit cube $[0,1]^0$ and every edge is homeomorphic to a interval [0,1]. Then we can say that all graphs are homeomorphic to a 1-dimensional cubical complex.

Another example can be the set of 2-dimensional cubes in a plane like we have seen in example 1.12.

## 3.2 Direct and indirect adjacency

The fundamental principles of the following part is mainly came from the paper of [2], the theorems, figures, proofs.... We will not cite it very frequently along the following part. This chapter will be a stepping stone for the next chapter which is the most theoretical of the book and is the center of this work. This and the next chapter will be a mixture of my interpretations and notes with the content of the article [2].

There are two ways to construct a cubical complex from an image $I$: The V-construction $V(I)$ represents voxels by vertices and the T-construction $T(I)$ represents voxels by top-dimensional cubes.

In fact, V-construction and T-construction correspond respectively to two different voxel connectivity methods.

The V construction corresponds to the direct adjacency in image analysis, where voxels are connected if and only if their grid positions differ by 1, so each voxel has 2d neighbors. For d = 2, the pixels are 4-connected, with direct neighbors to the left and right and above and below.

The T-construction corresponds to indirect adjacency, where voxels are also connected diagonally, each voxel has $3d - 1$ neighbors, and the pixels are 8-connected.

It is well known that the choice of direct or indirect adjacency affects the overall topology of binary images and the critical points of grayscale image functions. This effect is particularly significant when the structure of the image is similar to the length scale of a digital grid.

This figure was came from [2].

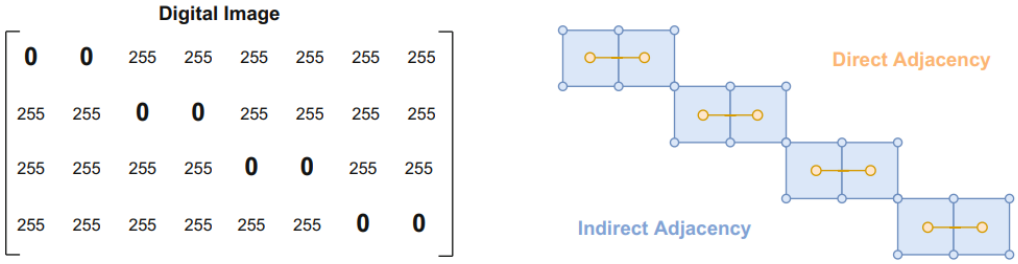



Figure 3.2: Smooth closed manifold without boundaries

As shown in Figure 3.2, this is a non-square matrix, but we still found a diagonal line where all elements are equal to 0. If we use direct connectivity, we only consider the four directions of the element directly above, directly below, directly to the left, and directly to the right, without considering the connectivity in the

diagonal direction. In this case, the yellow direct connection will only connect the two 0s in each row, and will not connect to the 0s diagonally above the left and diagonally below the right. In this way, four connected components are finally constructed.

If we use indirect connectivity, we will consider the connectivity of this element in all possible eight directions around the two-dimensional nine-square grid, including the diagonal direction. In this way, the blue indirect connection will connect the two 0s in each row and connect them to the 0s above and below the left and right diagonally. In this way, a connected component is finally constructed.

Similarly, it is obvious that direct connection does not consider diagonal connectivity, so it can be equivalent to treating each element as a vertex, because vertices do not have diagonal connectivity, so it is equivalent to V-construction. Indirect connection considers all eight directions, which is equivalent to replacing each element with a square, and the four sides and four vertices of the square can be connected to other sides and vertices, so it is equivalent to T-construction.

Using different connectivity methods will result in changes in the number of homology groups and their respective birth and death times.

## 3.3 T-construction and V-construction

In this section, we are not only define the V- and T-construction (informally, see the next chapter for formal definition start from def 4.18), we also make an example to compute the persistence of their homology groups $H_0$ and $H_1$. To compare their difference and figure out why this will happen.

**Definition 3.3.** If we transform all voxels of the image into a vertex (i.e., 0-dimensional unit cubes) and only consider four connectivity directions of the element directly above, below, to the left, to the right and connect them only use edges (i.e., homeomorphic to the interval which is 1-dimensional unit cubes) then we call this type of construction as *V-Construction*. (In some sense, this is a graph.)

And if we transform all voxels into a cube (i.e., 2-dimensional unit cubes) and consider all eight connectivity directions of the element allowing the connections through the four vertices and four sides of the cube. And do not allow the connection just using vertices or edges. Then we call this type of construction as *T-Construction*.

We are so concerned with the different construction ways because this is also a huge difference between simplicial complexes and cubical complexes. Although in simplicial complexes there are various ways to construct the filtration such as *Čech complex* and *Vietoris-Rips complex* (for more details please see my bachelor

thesis **??**) but it just differs in the counting distance between the vertices (all points are vertices in simplicial complex no matter the construction) and the criterion to make them connected. In the case of V- and T-construction, the voxels are not equal, it will be vertex or cubes depending on different construction, and the way of connecting are also different will be just edges or vertices&edges of a cube.

So that we can easily apply the same method for a Čech complex or a Vietoris-Rips complex and then get some different results without dual relation between them. In the following sections, we can see the dual relation between the homology groups of V- and T- constructions using respectively sub-level set and super-level set for cubical complexes.

But for now, we see first an example that compare the different homology groups because of two different constructions. Note that here we only use one filtration based on sub-level set which is similar to what we defined before but not using the height but grayscale value.

**Example 3.4.** Note that this figure and the following barcode for homology groups, the padded image also came from the paper of [2].

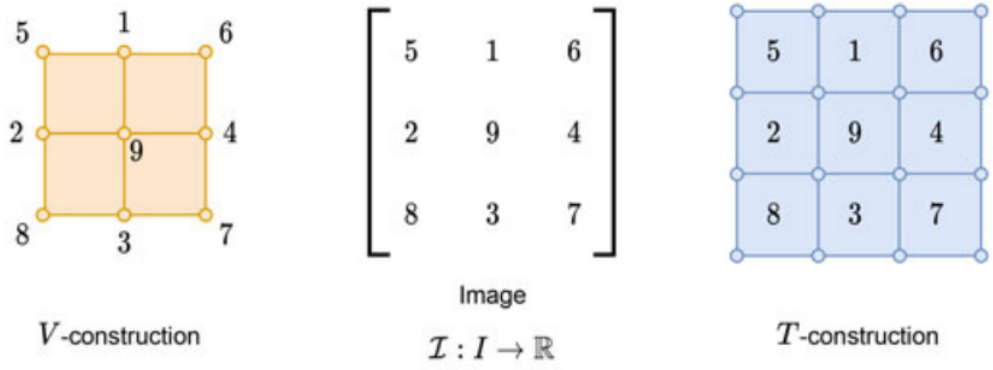



Figure 3.3: V- and T- construction for an image

This figure is cited from the paper [2]. We transformed the original $2D-$ image into a numerical grayscale matrix and then we used the V- and T- construction for this image. Note that from V-construction we get a $2 \times 2$ cubical complex and from T-construction we get a $3 \times 3$ cubical complex equal to the original image. This detail will be used in the definition 4.18.

We put the birth and death time of homology groups with dimensions 0 and 1 in the form of barcode.

We use sub-level sets based on the grayscale values starts from 1. As V-construction do not consider the connectivity in diagonal directions so that in the

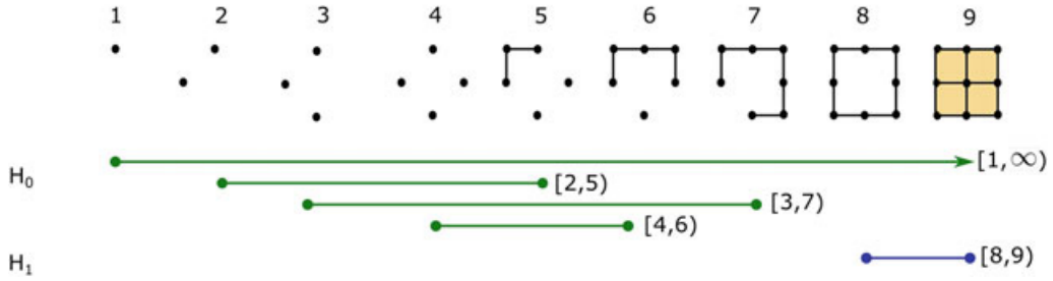



Figure 3.4: Barcode for $H_0$ and $H_1$ of V-construction

first four steps we only have four separated vertices, four connected components $H_0$ and from the step with value equal to 5 we destroyed a $H_0$ with birth time equal to 2 because the criterion for destroying the $n$-cycles (said in def 1.11) is always choosing the one with larger birth time, so we chose the connected component generated by the vertex with value 2.

Then we repeat the same process until the step with value equal to 8 when a 1-cycle (simply cycle) was born and then it died in the next step because the cycle has been filled by the last vertex.

Now we are going to see the case using T-construction.

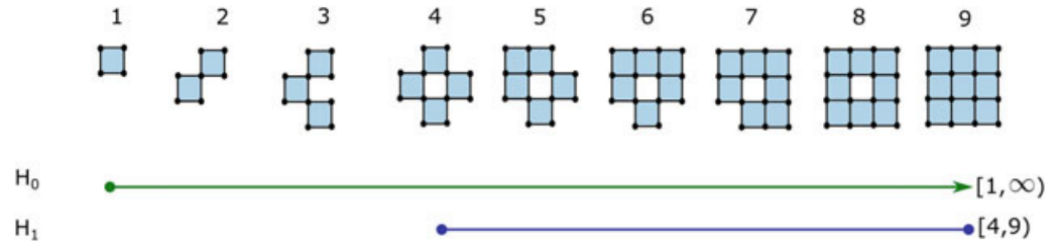



Figure 3.5: Barcode for $H_0$ and $H_1$ of T-construction

It is easy to see we just have one connected component because in each step the cube can be connected diagonally or directly by its side so that this $H_0$ persists forever. And the unique cycle $H_1$ was born in the step with value equal to 4 because the cubes enclose an empty space and this space died in the last step filling by the last vertex with value 9.

Obviously, if we use the same filtration and the same direction without any modification for the original image, the persistence for these homology groups are different so that it cannot help us to find the possible dual relation between them.

## 3.4 Modification for duality and padded T-construction

As we have seen in the previous section, we need to do some modification to make the dual relation between to be established.

The method we will use is called *Padding* and it will be defined formally in def 4.20.

Here we are just looking an example to compare the relation between them.

**Example 3.5.** First we padded an outer layer that enclose the matrix with values $N$ which are larger then the maximal value of the image.

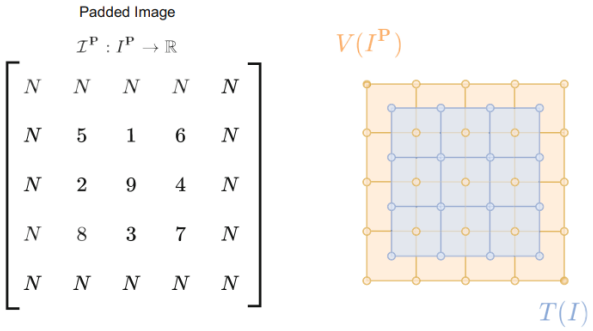



Figure 3.6: Padded image

Then we can choose the V-construction using padded image $I^P$ or T-construction use it. In the figure 3.6 we used V-construction considering all boundary elements as vertices same to the interior number.

And now we see another figure that shows a dual relation exist in the case we used padded image and two reversal directions.

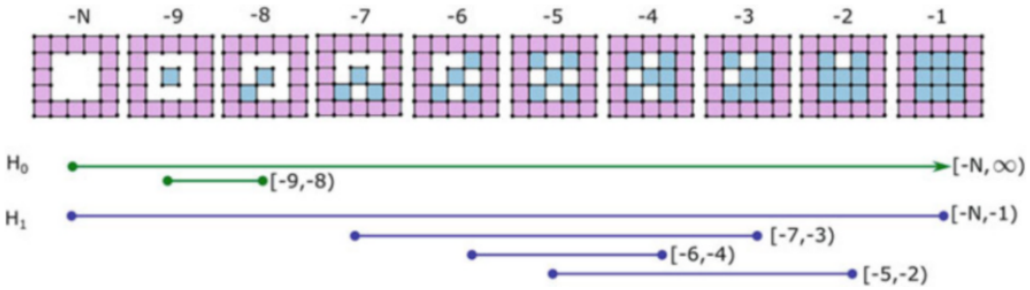



Figure 3.7: Padded image with T-construction in negative direction

In this case we also used the sub-level set started from $-N$ with $N > max\mathcal{I}^P$ but the idea is actually given by super-level set.

We can see that there are two $H_0$ with persistence $[-N, \infty)$ and $[-9, -8)$. In practical cases we usually eliminated the persistence which contains $\infty$ so we only consider $[-9, -8)$. And it is exactly (with exchange the extremes and symbols) the result of $H_1$ of V-construction using non-padded image and positive direction filtration with persistence $[8, 9)$. And the rest $H_1$ are also exchanging the extremes and symbols from the persistence of $H_0$ in the figure 3.4. Note that $N$ can be seen as $\infty$ so that $[-N, -1)$ corresponds to $[1, \infty)$.

So that it is possible to calculate $H_1$ using $H_0$ with this dual phenomenon and it will be more efficient because $H_0$ is easier to compute.

# Chapter 4

# Duality for cubical complexes

In this chapter, we will prove the dual relation between the T- and V-constructions using sub-level sets in two directional filtrations.

## 4.1 Basic definitions and notations

**Definition 4.1.** A CW-complex is *regular* if the closure of each $k$-cell is homeomorphic to the closed $k$-dimensional ball $D^k$. In this work, we call every finite regular CW-complex as a *cell complex*. And the dimension of a cell complex $C$ is the maximum dimension of cells in $C$, they are defined formally as $dim(C) = max\{dim(c) \mid c \in C\}$.

Let $C$ be a cell complex with cells $\tau$ and $\sigma$ with different dimensions.

1. If $\tau \subseteq \overline{\sigma}$ then $\tau$ is called a *face* of $\sigma$, and $\sigma$ is then called a *coface* of $\tau$, denoted by $\tau \preceq \sigma$. (In this paper the symbol $\preceq$ do not related to partially ordered sets.)

2. The *codimension* of a pair of cells $\tau \preceq \sigma$ is the difference of their dimensions, $codim(\tau, \sigma) := dim(\sigma) - dim(\tau)$. If $codim(\tau, \sigma) = 1$, we call $\tau$ is a *facet* of $\sigma$, and it will be denoted by $\tau \lhd \sigma$. For example, $\sigma$ is a triangle and a side of it $\tau$ is its facet.

3. A function $f : C \to \mathbb{R}$ on the cells of $C$ is *monotonic* if it increases with the dimension, that is, $f(\tau) \leq f(\sigma)$ whenever $\tau \preceq \sigma$.

## 4.2 Dual filtered cell complexes and persistence pair

**Definition 4.2.** Two $d$-dimensional cell complexes $C$ and $C^*$ are *combinatorially dual* if there is a bijection $C \to C^*$ and $\sigma \mapsto \sigma^*$ between the sets of cells $\sigma, \sigma^*$ such that

1. *Dimension complementary*: $\dim(\sigma^*) = d - \dim\sigma$ for all $\sigma \in C$.

2. *Face reversal* $\sigma \preceq \tau \iff \tau^* \preceq \sigma^*$ for all $\sigma, \tau \in C$.

**Definition 4.3.** A *filtered cell complex* $(C, f)$ is ordered pair with a cell complex $C$ together with a monotonic function $f : C \to \mathbb{R}$. Consider a sequence of the cells $\sigma_0, \sigma_1, \ldots, \sigma_n$ in $C$, such that $\sigma_i \preceq \sigma_j$ implies $i \leq j$.

We call this sequence of cells is *compatible* with the function $f$ if it satisfies

$$f(\sigma_0) \leq f(\sigma_1) \leq \cdots \leq f(\sigma_n).$$

We can define sub-level sets with respect to a monotonic function in a similar way the sub-level sets and super-level sets as above with respect to Morse function.

The sub-level set $C_r := f^{-1}(\infty, r]$ is a subcomplex of $C$.

Unlike the previous Morse function, the $f(\sigma)$ here reflects when a cell $\sigma$ enters the filtration given by this sequence of subcomplexes. For example, $f(\sigma) = r$ means the cell $\sigma$ has been contained in the subcomplex until the filtration reached step $r$.

This result also implies that we can construct a filtration based on sub-level sets using monotonic function.

**Definition 4.4.** Two filtered complexes $(C, f)$ and $(C^*, g)$ are *dual filtered complexes* if $C$ and $C^*$ are combinatorially dual to one another and if there exists a sequence $\sigma_0, \sigma_1, \ldots, \sigma_n$ of the cells in $C$ that is compatible with $f$ and its dual sequence $\sigma_n^*, \sigma_{n-1}^*, \ldots, \sigma_0^*$ of the cells in $C^*$ is compatible with $g$.

**Proposition 4.5.** Suppose two functions $f : C \to \mathbb{R}$ and $f^* : C^* \to \mathbb{R}$ satisfy $f^*(\sigma^*) = -f(\sigma)$. Then $(C, f)$ and $(C^*, f^*)$ are dual filtered complexes.

We have defined the concept of persistence of homology groups in the chapter of introduction, the definition 1.10. Based on the persistence, we can define a multiset of persistence.

**Definition 4.6.** Given a filtered complex $(C, f)$, we obtain inclusions $f^{-1}((-\infty, r]) \to f^{-1}((-\infty, s])$ of sub-level sets for $r \leq s$. We construct the filtration using this inclusion then calculate the $n$-dimensional homology groups $H_n$ and also their persistence.

The *n-dimensional persistence set* of $f$ is the multiset defined as

$$Per_n(f) := \{[birth_i, death_i) | i \in I\}$$

where $I$ is the index set with cardinality equal to the number of the corresponding $n$-dimensional homology groups $H_n$, and $Per_n(f)$ can be considered a set which contains all birth and death time of $H_n$.

The multiset of persistence pairs for all dimensions is defined as

$$Per(f) = \bigcup_{n=0}^{\dim(C)} \mathrm{Per}_n(f).$$

It is possible that some homology groups have death time at $\infty$, so we also classify the finite and infinite persistence. Writing $\mathrm{Per}_{\mathbb{F}}(f)$ for the multiset of finite intervals with death time $< \infty$, and $\mathrm{Per}_{\infty}(f)$ for the remaining cases. We obtain $\mathrm{Per}(f) = \mathrm{Per}_{\mathbb{F}}(f) \cup \mathrm{Per}_{\infty}(f)$.

To compute the persistence of $f$, $\mathrm{Per}(f)$, we choose an ordering $\sigma_0, \sigma_1, \ldots, \sigma_n$ of the cells in $C$ that is compatible with $f$.

We say that cells $\sigma_i$ and $\sigma_j$ appear at the same step of filtration generated by the sequence of sub-level sets $(f^{-1}((-\infty, r]))_{r\in\mathbb{R}}$ if $f(\sigma_i) = f(\sigma_j)$ (as we defined previously, the value of f indicates when the cell enter the filtration). We must add exactly one cell at every step to make the computation. It is similar to the construction of filtration for simplicial complexes but more careful because each time we do not add subcomplex but just add exactly one cell.

Each $\{\sigma_0, \ldots, \sigma_i\}$ is a union of cells and it also is a subcomplex.

$$\emptyset \subset \{\sigma_0\} \subset \{\sigma_0, \sigma_1\} \subset \cdots \subset \{\sigma_0, \sigma_1, \ldots, \sigma_{n-1}\} \subset \{\sigma_0, \sigma_1, \ldots, \sigma_n\} = C.$$

Every sub-level set $f^{-1}((-\infty, r])$ appears somewhere in this sequence:

$$f^{-1}((-\infty, r]) = f^{-1}((-\infty, f(\sigma_i)] = \{\sigma_0, \sigma_1, \ldots, \sigma_i\}$$

for $i = \max\{i = 0, \ldots, n \mid f(\sigma_i) \leq r\}$.

Based on this type of filtration using the sub-level sets generated by the anti-image of $f$, we can calculate the birth and death time using the value of $f$.

**Definition 4.7.** A pair $(\sigma_i, \sigma_j)$ of cells where $\sigma_j$ kills the homology group created by $\sigma_i$ is called a *persistence pair*. That means $\sigma_i$ is the creator and $\sigma_j$ is the destroyer of the homology group as we defined in the definition 1.11.

A persistence pair $(\sigma_i, \sigma_j)$ corresponds to the interval $[f(\sigma_i), f(\sigma_j)) \in \mathrm{Per}_{\mathbb{F}}(f)$. If $f(\sigma_i) = f(\sigma_j)$ then this interval can be empty.

A creator cell $\sigma_i$ with no corresponding destroyer cell is called *essential*, and corresponds to the interval $[f(\sigma_i), \infty) \in \mathrm{Per}_{\infty}(f)$. For example, we usually get an essential cell in the case of $H_0$. The creation of the first cell and this connected

component will persist until the whole filtered cell complex has been constructed in the last step of filtration.

To investigate the boundary of subcomplex consist of cells, we need to define a matrix that contains total information of their boundaries. Similar to the adjacency matrix in graph theory.

## 4.3 Rank values for total boundary matrix

**Definition 4.8.** In persistent homology, we work with the $\mathbb{Z}/2\mathbb{Z}$ (i.e. binary matrix with just 0 and 1) *total boundary matrix* $D$ which is defined by $D_{i,j} = 1$ if $\sigma_i \lhd \sigma_j$ and 0 otherwise.

That means if $\sigma_i$ is a facet of $\sigma_j$ then we indicate it with 1. Obviously the boundary of $n$-cell are $n-1$-cells which have common faces with it and the codimension between them is equal to 1. So the one facet can be seen as a subset of the boundary.

**Definition 4.9.** Define a rank value for total boundary matrix and its sub-matrix

$$r_D(i,j) = \text{rank } D_i^j - \text{rank } D_i^{j-1} - \text{rank } D_{i+1}^j + \text{rank } D_{i+1}^{j-1}$$

where $D_i^j = D[i:n, 0:j]$ is the *lower-left sub-matrix* of $D$ attained by deleting the first rows up to $i-1$ and the last columns starting from $j+1$.

Obviously this ranking value $r_D(i,j)$ consider the order of the indexes because $r_D(i,j) \neq r_D(j,i)$, deleting the first rows up to $j-1$ and the last columns starting from $i+1$ yields different matrices.

**Theorem 4.10.** *Pairing Uniqueness theorem*

This theorem is proven in [3]. Given a linear ordering of the cells $\sigma_0, \sigma_1, \ldots, \sigma_n$ in a filtered cell complex $C$, $(\sigma_i, \sigma_j)$ is a persistence pair (as defined in definition 4.7) if and only if $r_D(i,j) = 1$.

**Corollary 4.11.** If $r_D(i,j) \neq 1$ and $r_D(j,i) \neq 1$ for all $j$ then the cell $\sigma_i$ is essential.

*Proof.* Pairing Uniqueness theorem told us that if $(\sigma_i, \sigma_j)$ is a persistence pair (means they are creator and destroyer of $H_n$) then $r_D(i,j) = 1$.

So that $(\sigma_i, \sigma_j)$ can not be a persistence pair because $r_D(i,j) \neq 1$, neither $(\sigma_j, \sigma_i)$ because of $r_D(j,i) \neq 1$.

These two cells are neither a creator nor a destroyer, that implies that $\sigma_i$ must be an unpaired cell. It is possible to have a cell without death but it is impossible

that we have a cell with only death without birth. So this $\sigma_i$ only can be a cell with certain birth time and persist forever, which means it is essential.

□

We have defined persistence pair consist of creator and destroyer cells in order. If cells have their own dual, then it is also possible to find dual persistence pair and it will help us to study dual cells and homology groups.

The anti-transpose matrix is to flip the matrix about the secondary diagonal (the diagonal from the upper right to the lower left). The elements on the secondary diagonal remain unchanged, and the other elements are mirrored along the diagonal.

For a matrix $A$ of order $n$, then its $A^{AT}$ will put each $a_{i,j}$ in $a_{n-j+1,n-i+1}$.

**Example 4.12.**

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

$$A^{AT} = \begin{pmatrix} i & f & c \\ h & e & b \\ g & d & a \end{pmatrix}$$

Thus, we have the anti-transpose of $A^{AT}$.

We denote the anti-transpose of the matrix $D$ by $D^{AT}$ and then $D_{i,j}$ delete the first rows up to $i-1$ and the last columns starting from $j+1$ becomes to delete rows up to $n-j+1-1$ and the last columns starting from $n-i+1-1$, i.e. $D^{AT}_{i,j} = D_{n-j,n-i}$. Anti-transposition is also the composition of standard matrix transposition with a reversal of the order of the columns and of the rows.

For $(C, f)$ and $(C^*, g)$ dual filtered cell complexes with $n+1$ cells. Suppose there are a sequence $\sigma_0, \sigma_1, \ldots, \sigma_n$ of the cells in $C$ that is compatible with the monotonic function $f$ of filtered cell complex $(C, f)$, and that $\sigma^*_n, \sigma^*_{n-1}, \ldots, \sigma^*_0$ is the dual sequence that is compatible with $g$. And we can construct a bijection $\sigma^* \mapsto \sigma$ with $\sigma_i$ the $i$-th cell in the sequence and $\sigma^*$ is the $n-i$-th cell of the dual sequence.

As these filtered cell complexes are dual, then satisfies the property of dimension complemantary which means $dim(\sigma^*) = n - dim(\sigma)$.

Let $D$ be the total boundary matrix of $C$ and $D^*$ be the total boundary matrix of $C^*$ with their respective orderings.

**Lemma 4.13.** The dual total boundary matrix $D^*$ is the anti-transpose $D^{AT}$ of $D$, that is,

$$D^*_{i,j} = D_{n-j,n-i} = D^{AT}_{i,j}.$$

*Proof.* To prove the equality between matrices then we can prove it using aribitrary element of them.

If we can prove for all $i, j$ arbitrary, the $D^{AT}_{i,j} = D^*_{i,j}$ then we make the proof.

Remember the definition ($D_{n-j,n-i} = 1$ if $\sigma_i \lhd \sigma_j$) and the face reversal property by dual filtered cell complexes.

$$D^{AT}_{i,j} = 1 \iff D_{n-j,n-i} = 1 \iff \sigma_{n-j} \lhd \sigma_{n-i} \iff \sigma^*_{n-i} \lhd \sigma^*_{n-j} \iff D^*_{i,j} = 1$$

□

**Lemma 4.14.** The sub-matrices defined in 4.9 $D^j_i = D[i : n, 0 : j]$ satisfies

$$(D^j_i)^{AT} = (D^{AT})^{n-i}_{n-j}$$

and thus

$$rankD^j_i = rank(D^{AT})^{n-i}_{n-j}$$

and

$$r_D(i, j) = r_{D^{AT}}(n - j, n - i).$$

*Proof.* The first statement follows from

$$(D^j_i)^{AT} \underset{def 4.9}{=} (D[i : n, 0 : j])^{AT} \underset{Lemma 4.13}{=} D^{AT}[(n - j) : n, 0 : (n - i)] \underset{def 4.9}{=} (D^{AT})^{n-j}_{n-i}.$$

From the first statement we know $(D^j_i)^{AT} = (D^{AT})^{n-i}_{n-j}$, then the rank value will be the same $rank(D^j_i)^{AT} = rank(D^{AT})^{n-i}_{n-j}$. And the anti-transposition operation is rank preserving, so the second statement can be proved.

By the definition of rank value (def 4.9)

$$r_D(i, j) = rankD^j_i - rankD^{j-1}_i - rankD^j_{i+1} + rankD^{j-1}_{i+1}$$

$$\underset{rankD^j_i = rank(D^{AT})^{n-i}_{n-j}}{=} rank(D^{AT})^{n-j}_{n-i} - rank(D^{AT})^{n-j+1}_{n-i} - rank(D^{AT})^{n-j}_{n-i+1} + rank(D^{AT})^{n-j+1}_{n-i+1}$$

$$= r_{D^\perp}(n - j, n - i).$$

□

**Theorem 4.15.** *Persistence of Dual Filtrations*

Let $(C, f)$ and $(C^*, g)$ be dual filtered complexes with compatible ordering sequence $\sigma_0, \sigma_1, \ldots, \sigma_n$. Then

1. $(\sigma_i, \sigma_j)$ is a persistence pair in the filtered cell complex $(C, f)$ if and only if $(\sigma_j^*, \sigma_i^*)$ is a persistence pair in $(C^*, g)$.

2. $\sigma_i$ is essential in $(C, f)$ if and only if $\sigma_i^*$ is essential in $(C^*, g)$.

*Proof.* From the Lemma 4.13 we know that $D^*_{i,j} = D^{AT}_{i,j}$ and the third statement of lemma 4.14 implies that $r_D(i,j) = r_{D^{AT}}(n-j, n-i)$. Therefore, $r_D(i,j) = r_{D^*(n-j,n-i)}$, and then

$$r_D(i,j) = 1 \iff r_{D^*}(n-j, n-i) = 1.$$

By the Pairing Uniqueness Theorem 4.10, the above implies that $(\sigma_i, \sigma_j)$ is a persistence pair meanwhile the $(n-j)$-th cell of the dual filtration $(X^*, g)$ is paired with the $(n-i)$-th. As we said before, $\sigma_j^*$ is the $(n-j)$-th cell of the ordering sequence $\sigma_n^*, \sigma_{n-1}^*, \ldots, \sigma_0^*$. So that $(\sigma_j^*, \sigma_i^*)$ is a persistence pair in $(C^*, g)$.

For the second statement, we can also see the third statement of Lemma 4.14 that tells us that the following two statements are equivalent:

- Both $r_D(i,j) \neq 1$ and $r_D(j,i) \neq 1$ for all $j$.
- Both $r_{D^*}(n-j, n-i) \neq 1$ and $r_{D^*}(n-i, n-j) \neq 1$ for all $n-j$.

(Because $r_D(i,j) = r_{D^*(n-j,n-i)}$)

By Corollary 4.11, this means that $\sigma_i$ is an essential cell in $(C, f)$ if and only if the $(n-i)$-th cell $\sigma_i^*$ is essential in the dual filtration $(C^*, g)$.

□

We have proved all above theorems and propositions to build a relation between dual filtered complexes and persistence of a monotonic function $f$. We focus on $Per(f)$ because persistence is deeply related to homology groups.

And the following corollary made this:

**Corollary 4.16.** Let $(C, f)$ and $(C^*, g)$ be dual filtered complexes. Then

1.
$$[f(\sigma_i), f(\sigma_j)) \in \mathrm{Per}_{n,\mathbb{F}}(f) \iff [g(\sigma_j^*), g(\sigma_i^*)) \in \mathrm{Per}_{d-n-1,\mathbb{F}}(g).$$

2.
$$[f(\sigma_i), \infty) \in \mathrm{Per}_{n,\infty}(f) \iff [g(\sigma_i^*), \infty) \in \mathrm{Per}_{d-n,\infty}(g).$$

*Proof.* A persistence pair $(\sigma_i, \sigma_j)$ of an ordering sequence compatible with the function $f$, the birth value is $f(\sigma_i)$ and the death value is $f(\sigma_j)$ as we said after the definition 4.7. The result then follows directly from respectively the statements 1, 2 of Theorem 4.15. □

## 4.4 Formal definition for V- and T-constructions

Now we are going to define more formally the V- and T-constructions than the definition in chapter 3.

As we said in the chapter 3 that we can transform a digital scanned $2D$ or $3D$ images into a numerical matrix (or array) with grayscale value for each pixel (or voxel). Then we can say that for a $d$-dimensional image of size $(n_1, \ldots, n_d)$ is a real-valued function that returns a numerical array defined as

$$\mathcal{I} : I = [[1, n_1]] \times \cdots \times [[1, n_d]] \to \mathbb{R}$$

where each $e_i$ is the size of a dimension and each interval $[[1, n_i]]$ is a closed interval of natural numbers because the size of a certain dimension of the image cannot be non-integer or negative numbers.

For $d = 2$, then each element (point) $p \in I$ is called *pixel* and if $d > 2$ then it will be called *voxel*.

In the definition 3.1, we defined the concept of elementary interval $I = [l, l+1]$ or $I = [l, l]$ and elementary cube $Q = I_1 \times \cdots \times I_n \subset \mathbb{R}^n$. We go one step further defining *elementary k-cube*.

**Definition 4.17.** An *elementary k-cube* $\gamma \subset \mathbb{R}^d$ is the product of $d$ elementary intervals (note that the elementary intervals defined here is almost the same to the $I_i$)

$$\gamma = e_1 \times e_2 \times \cdots \times e_d.$$

The difference is the dimension, if we have a lower dimension $k < d$ of the intervals have the form $e_i = [l_i, l_i + 1]$ and the rest $d - k$ are degenerate, $e_i = [l_i, l_i]$.

Note that as what we defined previously a cubical complex $C \subset \mathbb{R}^d$ is a cell complex consisting of a set of elementary $k$-cubes with all faces of $\gamma \in C$ are also in $C$.

Based on this new definition of cubical complex we have the formal definition for V- and T-constructions.

**Definition 4.18.** Given a $d$-dimensional grayscale digital image $\mathcal{I} : I \to \mathbb{R}$, of size $(n_1, n_2, \ldots, n_d)$, the *V-construction* is a filtered cell complex $(V(I), V(\mathcal{I}))$ (note that $I$ is the image and $\mathcal{I}$ is the grayscale function on $I$) defined as follows.

1. $V(I)$ is a cubical complex built from an array of $(n_1 - 1) \times \cdots \times (n_d - 1)$ elementary $d$-cubes as the figure 3.3 displayed in the previous example and all their faces.

2. The vertices $v^{(0)} \in V(I)$ (here we using the notation (0) because we consider the order for V start from 0-cells, i.e. vertices) are indexed exactly by the elements $p \in I$, and we define the function $V(\mathcal{I})$ firstly on these vertices as,

$$V(\mathcal{I})(v^{(0)}) = \mathcal{I}(p).$$

Then for an elementary $k$-cube $\gamma$ which can be seen as a subcube of $I$, the function takes the maximal value of grayscale for its vertices.

$$V(\mathcal{I})(\gamma) = \max_{v^{(0)} \preceq \gamma} V(\mathcal{I})(v^{(0)}).$$

And then the function $V(\mathcal{I})$ is also monotonic by the definition.

**Definition 4.19.** Given the same conditions as above. The *T-construction* is a filtered cell complex $(T(I), T(\mathcal{I}))$ defined as follows.

1. $T(\mathcal{I})$ is a cubical complex built from the array of $n_1 \times \cdots \times n_d$ elementary $d$-cubes and all their faces. Note that here is a difference with $V-construction$, we replace all vertices of $I$ by an elementary $d$-cube so there is no need to minus 1.

2. The $d$-cells $\tau^{(d)} \in T(I)$ are indexed exactly by the elements $p \in I$, and we define the function $T(\mathcal{I})$ firstly on these top-dimensional cells as,

$$T(\mathcal{I})(\tau^{(d)}) = \mathcal{I}(p).$$

Note that for $T(\mathcal{I})$ we use $d$-cells instead of vertices in the case of $V(\mathcal{I})$.

Then for an elementary $k$-cube $\gamma$, the function takes the smallest value of grayscale of all its adjacent (by sides and vertices) $d$-cubes,

$$T(\mathcal{I})(\gamma) = \min_{\gamma \preceq \tau^{(d)}} T(\mathcal{I})(\tau^{(d)}).$$

This ensures that $T(\mathcal{I})$ is monotonic with respect to the face relation on $T(I)$.

**Definition 4.20.** The *padded image* $\mathcal{I}^P : I^P \to \mathbb{R}$ has image domain $I^P = [0, n_1 + 1] \times \cdots \times [0, n_d + 1]$ and image function.

$$\mathcal{I}^P(p) = \begin{cases} \mathcal{I}(p), & \text{for } p \in I \\ N, & \text{for } p \in I^P \setminus I. \end{cases}$$

with $N > \max_{p \in I} \mathcal{I}(p)$ and in practical cases we consider $N$ as $\infty$.

So that the padded image can be compatible perfectly with V- and T- constructions.

## 4.5 Duality between deformed constructions

If we identify the padded boundary vertices of V($I^P$) then we can transform this cubical complex into a sphere just like the construction of *Riemann sphere* identifying the $\infty$ as a point.

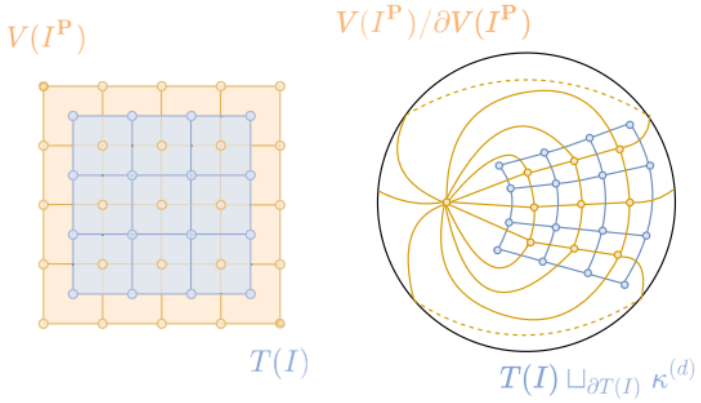



Figure 4.1: Sphere formed by identifying boundary vertices

As the padded V-construction has been transformed into a sphere so the image created using T-construction will be also deformed into a cursive manifold with boundary. So we consider a new deformed figure obtained by attaching a $d$-cell $\kappa^(d)$ along the boundary of $T(I)$ and denote it by $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$.

We can see some dual relations between these two deformed figures, it seems like a version of Alexander duality because we deformed padded V-image into a compact, locally contractible subspace of the sphere $S^2$. But the formal demonstration is not here, it is just an idea that may implies the existence of duality.

**Lemma 4.21.** Given an image $I$, the quotient of the padded V-construction modulo its boundary, $V(I^P)/\partial V(I^P)$, is combinatorially dual (defined in def 4.2) to $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$ T-construction no padded attaching boundary with $d$-cell.

*Proof.* Each elementary $k$-cube, $\gamma \in V(I^P)$ takes the form

$$\gamma = e_1 \times \cdots \times e_d, \quad e_i = [l_i, l_i + 1] \text{ or } e_i = [p_i, p_i]$$

where $k$ of the elementary intervals are non-degenerate with $l_i \in \{0, \ldots, n_i\}$ with the size until $n_i$ and the maximal $k$-cube will be $[n_i, n_i + 1]$ and $(d - k)$ are degenerate with $p_i \in \{0, \ldots, n_i + 1\}$. Note that $\gamma \in \partial V(I^P)$ if this cube has at least one degenerate interval with $p_i = 0$ or $(n_i + 1)$ because two extreme values represent

the boundary of both sides. Now we consider the following cell constructed from $\gamma$:

$$\gamma^* = e_1^* \times \cdots \times e_d^*, \quad e_i^* = [l_i + \frac{1}{2}, l_i + \frac{1}{2}] \text{ or } [p_i - \frac{1}{2}, p_i + \frac{1}{2}]$$

with $l_i$ and $p_i$ as defined above. This cell has $k$ degenerate intervals because we replace these $k$ non-degenerate intervals into degenerate ones and similarly it has $(d-k)$ non-degenerate ones so $\gamma^*$ is an elementary $(d-k)$-cube, i.e. we constructed a dual elementary cube with respect to $\gamma$.

If we suppose that $\gamma \notin \partial V(I^P)$, then we see that $p_i$ cannot be neither 0 nor $n_i + 1$ which formed degenerate intervals in boundary then $p_i \in \{1, \ldots, n_i\}$, and the degenerate coordinate values $(l_i + \frac{1}{2}) \in \{\frac{1}{2}, \frac{3}{2}, \ldots, (n_i + \frac{1}{2})\}$ for $l_i \in \{0, \ldots, n_i\}$. We can see the dimension complementary property for $\gamma$ and $\gamma^*$ and then we obtain a bijection between $k$-cells $\gamma$ in $V(I^P) \setminus \partial V(I^P)$ and $(d-k)$-cells $\gamma^*$ in $T(I)$.

And then consider the $\gamma \in V(I^P)/\partial V(I^P)$, the vertices in the boundary are 0-cells so that their dual cells are $d$-cells such that the boundary $\partial T(I)$ attached in it. So the mapping from 0-cell $[\partial V(I^P)] \in V(I^P)/\partial V(I^P)$ to the $d$-cell attached to $\partial T(I)$ yields with previous paragraph a dimension complementary bijection between all cells of $V(I^P)/\partial V(I^P)$ and those of $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$ then the *dimension complementary* property of def 4.2 is done.

The next step is to confirm that the face relations between cells in $V(I^P)/\partial V(I^P)$ are mapped to coface relations in $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$ to prove the *face reversal* property. By the construction above, all interior (out of boundary) face relations for $V(I^P)$ map to coface relations for $T(I)$ because of the bijection between interior cells and their dual.

The last thing we need to check is that the (identified) vertex $[\partial V(I^P)]$ in $V(I^P)/\partial V(I^P)$ has dual face relations to the $d$-cell $\kappa^{(d)}$ attached to the boundary of $T(I)$ in $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$.

From *Face reversal* $\sigma \preceq \tau \iff \tau^* \preceq \sigma^*$ for all $\sigma, \tau \in C$ we have the equivalent statement as the following:

$$[\partial V(I^P)] \preceq \gamma \text{ in } V(I^P)/\partial V(I^P) \iff \gamma^* \preceq \kappa^{(d)} \text{ in } T(I) \sqcup_{\partial T(I)} \kappa^{(d)}.$$

We start from the direction left to right.If $\gamma^* \in \partial T(I)$ is a cell of the boundary, then it must be a face (or maybe a facet) of $\kappa^{(d)}$ because the whole boundary is attached in a same $d$-cell $\kappa^{(d)}$. And in the other direction, if $\gamma^*$ is a face of $\kappa^{(d)}$ then it also must belongs to the boundary $\partial T(I)$ as it was attached with the $d$-cell.

$$\gamma^* \preceq \kappa^{(d)} \iff \gamma^* \in \partial T(I)$$

This equivalence means that there is at least one of the degenerate elementary of all possible $k$ degenerate intervals of $\gamma^*$ has $0 + \frac{1}{2} = \frac{1}{2}$ or $(n_i + \frac{1}{2})$ with extreme

values $l_i = 0$ and $n_i$ as $\gamma^*$ belongs to the boundary $\partial T(I)$. So the corresponding dual elementary interval in the dual cell $\gamma$ is $e_i = [l_i, l_i + 1] = [0, 1]$ or $[n_i, n_i + 1]$.

Note that $\gamma \in \partial V(I^P)$ if this cube has at least one degenerate interval $e_i = [p_i, p_i]$ with $p_i = 0$ or $(n_i + 1)$ then this fact means the dual cell $\gamma$ must be contained in the boundary $\partial V(I^P)$ and then the vertex $[\partial V(I^P)]$ is a face of the elementary $k$-cube.

The converse implication follows in the same manner just replace $\gamma^*$ by the vertex $[\partial V(I^P)]$ and $\kappa^{(d)}$ by the dual cell $\gamma$ then do the same, and we are done with the proof.

□

**Lemma 4.22.** Given an image, $I$, the quotient of the padded T-construction modulo its boundary, $T(I^P)/\partial T(I^P)$, is the combinatorially dual of $V(I) \sqcup_{\partial V(I)} \kappa^{(d)}$.

As we have seen in the previous two lemmas, the padded image identifying the boundary of one construction is always combinatorially dual with the other, and the ordering cells sequence $\sigma_0, \sigma_1, \ldots, \sigma_n$ and $\sigma_n^*, \sigma_{n-1}^*, \ldots, \sigma_0^*$ are also reversal dual because of the two properties of combinatorially dual in def 4.2.

So that it is natural to consider using the idea of sub- and super-level sets in two directions to construct dual filtrations, in the next step we will define it formarlly.

If we multiply $-1$ to all elements of $I$, then if we start from the minimum value (considering negatives) is equal to start from the largest value of $I$, means we reversed the direction.

By the definition of the padded image, the functions $V(\mathcal{I}^P)$ and $V(-\mathcal{I}^P)$ (also true for $T$) are constant on the boundary with the value $N$ or $-N$ respectively. This fact induces a function defined as

$$\widetilde{V}(-\mathcal{I}^P) : V(I^P)/\partial V(I^P) \to \mathbb{R}$$

identifying all boundary vertices because they are constants with respect to the function $V(-\mathcal{I}^P)$.

And particularly

$$\widetilde{V}(-\mathcal{I}^P)([\partial V(I^P)]) = -N.$$

We used $-N$ instead of 0 or *NULL* of other quotient group functions and this $N$ is actually $\infty$ in practical cases for essential cells in both directions.

For other $k$-cubes $p \in I^P$ out of the boundary,

$$V(-\mathcal{I}^P)(p) = \max_{v^{(0)} \preceq p} V(-\mathcal{I}^{\mathcal{P}})(v^{(0)}).$$

Note that the negative symbol do not mean numerically be negative but the direction of filtration so that $V(-\mathcal{I}^P)$ is almost the same to $V(\mathcal{I}^P)$ just differing the order.

Similarly, the function $T(\mathcal{I})$ extends to a function $\widehat{T}(\mathcal{I})$ on $T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$

$$\widehat{T}(\mathcal{I}) : T(I)/\kappa^{(d)} \to \mathbb{R}$$

with $\widehat{T}(\mathcal{I})(\kappa^{(d)}) = N$. As identifying the boundary cubes in $T(I)$ to a same $d$-cell induces this function. Note that we are doing for $I$ not $-I$ in a positive direction so the value is positive for $N$.

For the reversal case, we have $\widetilde{T}(-\mathcal{I}^{\mathcal{P}}) : T(-I^P)/\partial T(-I^P) \to \mathbb{R}$ with padded modification in negative direction and $\widehat{V}(\mathcal{I}) : V(I)/\kappa^{(d)} \to \mathbb{R}$ without padded boundary in positive direction.

**Lemma 4.23.** For each $\gamma \in T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$ and dual cell $\gamma^* \in V(I^P)/\partial V(I^P)$ as we have defined and used in lemma 4.21

$$-\widehat{T}(\mathcal{I})(\gamma) = \widetilde{V}(-\mathcal{I}^P)(\gamma^*).$$

*Proof.* Firstly, suppose $\dim \gamma = d$ and $\gamma \neq \kappa^{(d)}$ means that $\gamma$ is a $d$-cell different from the one which is attached along the boundary $\partial T(I)$. Suppose $p \in I$ is the corresponding element of the image so that by the definition 4.19 $\mathcal{I}(p) = \widehat{T}(\mathcal{I})(\gamma)$. The dual cell $\gamma^* \in V(I^P)/\partial V(I^P)$ corresponds to the same voxel, as $\gamma \neq \kappa^{(d)}$ then $\gamma^* \neq [\partial V(-I^P)]$ so that $\widetilde{V}(-\mathcal{I}^P)(\sigma^*) \neq -N$ and by the definition the result is given as following

$$\widetilde{V}(-\mathcal{I}^P)(\gamma^*) = -I(p) = -\widehat{T}(\mathcal{I})(\gamma).$$

Note that the value is $-I(p)$ because of the reverse direction by the negative symbol.

Then the first case with $\gamma \neq \kappa^{(d)}$ is done then we consider the particular case.

For $\gamma = \kappa^{(d)}$, with its dual $[\partial V(I^P)]^*$, as we defined $\widehat{T}(\mathcal{I}) : T(I)/\kappa^{(d)} \to \mathbb{R}$ and $\widehat{T}(\mathcal{I})(\kappa^{(d)}) = N$ the function values satisfy the following

$$-\widehat{T}(\mathcal{I})(\kappa^{(d)}) = -N = \widetilde{V}(-\mathcal{I}^P)([\partial V(I^P)]).$$

The previous proof based on the case $\dim\gamma = d$ so we classified if it is equal to $\kappa^{(d)}$. Now we suppose $\gamma \in T(I) \sqcup_{\partial T(I)} \kappa^{(d)}$ and $\dim \gamma < d$. As we said above, for the cases when $\gamma \neq \kappa^{(d)}$ and $\gamma^* \neq [\partial V(I^P)]^*$ then $\widehat{T}(\mathcal{I})$ and $\widetilde{V}(\mathcal{I})$ are the same forms to $T(\mathcal{I})$ and $V(\mathcal{I})$. So by the definitions 4.18 and 4.19, we have the following:

$$-\widehat{T}(\mathcal{I})(\mathcal{I}) \underset{4.19}{=} -\min_{\tau^{(d)} \succeq \gamma} \widehat{T}(\mathcal{I})(\tau^{(d)}) = \max_{\tau^{(d)} \succeq \gamma} -\widehat{T}(\mathcal{I})(\tau^{(d)})$$

Because of the face reversal property of definition 4.2 $\sigma \preceq \tau \iff \tau^* \preceq \sigma^*$ for all $\sigma, \tau \in C$, we have

$$\tau^{(d)} \succeq \gamma \iff v^{(0)} \preceq \gamma^*$$

and

$$-\widehat{T}(\mathcal{I})(\tau^{(d)}) = -\mathcal{I}(p) = \widetilde{V}(-\mathcal{I}^P)(v^{(0)})$$

then

$$\max_{\tau^{(d)} \succeq \gamma} -\widehat{T}(\mathcal{I})(\tau^{(d)}) = \max_{v^{(0)} \preceq \gamma^*} \widetilde{V}(-\mathcal{I}^P)(v^{(0)}) = \widetilde{V}(-\mathcal{I}^P)(\gamma^*)$$

as required. □

Now we then define the functions $\widetilde{T}(-\mathcal{I}^P)$ on $T(I^P)/\partial T(I^P)$ and $\widehat{V}(\mathcal{I}^P)$ on $V(I^P) \sqcup_{\partial V(I^P)} \kappa^{(d)}$ similarly to those above.

**Lemma 4.24.** For each $\gamma \in V(I^P) \sqcup_{\partial V(I^P)} \kappa^{(d)}$ and its dual cell $\gamma^* \in T(I^P)/\partial T(I^P)$ we have

$$-\widehat{V}(\mathcal{I}^P)(\gamma) = \widetilde{T}(-\mathcal{I}^P)(\gamma^*).$$

**Corollary 4.25.** For a grayscale digital image $\mathcal{I} : I \to \mathbb{R}$

1. The filtered complexes $(T(I) \sqcup_{\partial T(I)} \kappa^{(d)}, \widehat{T}(I))$ and $(V(I^P)/\partial V(I^P), \widetilde{V}(-I^P))$ are dual.
2. The filtered complexes $(V(I^P) \sqcup_{\partial V(I^P)} \kappa^{(d)}, \tilde{V}(I^P))$ and $(T(I^P)/\partial T(I^P), \widetilde{T}(-\mathcal{I}^P))$ are dual.

*Proof.* The key of this proof is the previous lemmas and the Theorem 4.5, these results yield directly this corollary.

□

## 4.6 Persistence of deformed dual constructions

In the previous lemmas, we have studied the effect of padding a extra layer with value $N$ to their persistence, the operations we used are below:

1. Padding the image with a layer of voxels with value $N$.
2. Attach the boundary of $V(I^P)$ or $T(I)$ to a $d$-cell $\kappa^{(d)}$. Means the creation of functions we defined previously with hat symbol.

3. Identify the boundary in a negative directional padded filtration. Means the creation of functions we defined previously with tilde symbol and $-\mathcal{I}^{\mathcal{P}}$.

The first operation (padding the image) does not change the persistence and the second operation (attaching a $d$-cell) only creates an essential $d$-cycle ($H_d$) with birth an $N$ because the positive directional filtration create that $d$-cycle when it reached the value equal to $N > max(\mathcal{I})$ and then persist forever.

Formally we can state them in the following proposition.

**Proposition 4.26.** For an image $\mathcal{I} : I \to \mathbb{R}$ we have the following equalities between persistence of padded and original images

1. From the first operation we have

$$\mathrm{Per}(V(\mathcal{I}^P)) = \mathrm{Per}(V(\mathcal{I})) \quad \textit{and} \quad \mathrm{Per}(T(\mathcal{I}^P)) = \mathrm{Per}(T(\mathcal{I}))$$

2. From the second operation we have

$$\mathrm{Per}(\hat{V}(\mathcal{I})) = \mathrm{Per}(V(\mathcal{I})) \cup \{[N, \infty]_d\}$$

and

$$\mathrm{Per}(\hat{T}(\mathcal{I})) = \mathrm{Per}(T(\mathcal{I})) \cup \{[N, \infty]_d\}$$

We want to study the effect of taking the quotient of a padded image modulo the boundary (the third operation) to figure out the explicit relation with the persistence (mentioned in def 4.6) of $f$ which is the key to do the fast computation of homology groups.

As we did the transformation of image to padded construction and deformed the cubical complex to closed, locally compact sphere identifying the boundary with value $N$ and deform the other to a cursive manifold attaching a $d$-cell along the boundary. So we can see that now the deformed sphere is homeomorphic to a closed disc $D^d$ and we can compute the persistence $Per(f)$ with $f$ a monotonic function (definition 4.6) of this sphere using the following lemma.

**Lemma 4.27.** Let $C$ be a cell complex and take a monotonic function $f : C \cong D^d \to \mathbb{R}$ with

$$\gamma \in \partial C \Rightarrow f(\gamma) = -N = \min f$$

and induced quotient map $\tilde{f} : C/\partial C \to \mathbb{R}$. Then

$$\mathrm{Per}(\tilde{f}) = (\mathrm{Per}(f) \setminus \{[-N, \max f)_{d-1}\}) \cup \{[\max f, \infty]_d\}.$$

The whole proof is too long and we can see that in [2] page 19. Obviously, the definition of $\widetilde{f}$ is similar to $\widetilde{T}$ and $\widetilde{V}$ with $-\mathcal{I}^{\mathcal{P}}$ and then the result of $Per(\tilde{f})$ can help us to calculate *Per* for padded construction with negative direction.

**Corollary 4.28.** *For a d-dimensional image* $\mathcal{I} : I \to \mathbb{R}$

$$\mathrm{Per}(\widetilde{V}(-\mathcal{I}^P)) = \mathrm{Per}(V(-\mathcal{I}^P)) \setminus \{[-N, -\min I]_{d-1}\} \cup \{[-\min I, \infty]_d\}$$

*and*

$$\mathrm{Per}(\widetilde{T}(-\mathcal{I}^P)) = \mathrm{Per}(T(-\mathcal{I}^P)) \setminus \{[-N, -\min I]_{d-1}\} \cup \{[-\min I, \infty]_d\}$$

*Proof.* This corollary is the direct result of the previous lemma replacing $f$ by $V(-\mathcal{I}^P)$ and $T(-\mathcal{I}^P)$ and using $\max V(-\mathcal{I}^P) = -\min \mathcal{I}$ and $\max T(-\mathcal{I}^P) = -\min \mathcal{I}$. □

Finally, after all results proved above, we can reach the core result of this work, the equality between persistence of padded T-construction with negative directional filtration and V-construction with original image in positive directional filtration and this result is also true vice versa.

**Theorem 4.29.** For a grayscale image $\mathcal{I} : I \to \mathbb{R}$ the persistence of the $V$- and $T$-constructions satisfy

$$\mathrm{Per}_{\mathbb{F}}(V(\mathcal{I})) = \{[-q, -p)_{d-k-1} \mid [p, q)_k \in \mathrm{Per}_{\mathbb{F}}(T(-\mathcal{I}^P))\} \setminus \{[\min \mathcal{I}, N)_0\}.$$

Note that $\{[\min \mathcal{I}\}$ are all persistence of homology group with dimension 0 which are connected component, i.e., $H_0$ and

$$\mathrm{Per}_{\infty}(V(\mathcal{I})) = \{[\min \mathcal{I}, \infty)_0\}$$

only consists of essential cell which is a connected components $H_0$ that persist forever.

*Proof.* The infinite persistence for the function $V(\mathcal{I})$ only formed by $H_0$ and no other dimensions no matter what is the value of the dimension of $\mathcal{I}$. This result follows from the fact that $V(\mathcal{I}) \cong D^{(d)}$ for example for a $2D-$dimensional image with four pixels, V-construction will construct a square border that is homeomorphic to a $2D-$disc.

There will be a $H_0$ generated by the first element with minimal value added into the filtration, this connected component will be growing along the filtration

adding more elements with higher values and will never get killed that persists as a subset until when it reached the $maxV(\mathcal{I})$ and keep living until $\infty$ as whole set homeomorphic to a disc $D^{(d)}$ which is also a $H_0$. So we always have only one essential homology group $\mathrm{Per}_\infty(V(\mathcal{I})) = \{[\min \mathcal{I}, \infty)_0\}$.

For the finite case:

$$\mathrm{Per}_{\mathbb{F}}(V(\mathcal{I})) \underset{4.26}{=} \mathrm{Per}_{\mathbb{F}}(V(\mathcal{I}^{\mathcal{P}})) \underset{4.26, finite}{=} \mathrm{Per}_{\mathbb{F}}(\widehat{V}(\mathcal{I}^{\mathcal{P}}))$$

$$\underset{4.16(1)}{=} \{[-q,-p)_{d-k-1} \mid [p,q)_k \in \mathrm{Per}_{\mathbb{F}}(\widetilde{T}(-\mathcal{I}^P))\}$$

$$\underset{4.28}{=} \{[-q,-p)_{d-k-1} \mid [p,q)_k \in \mathrm{Per}_{\mathbb{F}}(T(-\mathcal{I}^P))\} \setminus \{[-N,-\min \mathcal{I})_{d-1}\}\}$$

$$= \{[-q,-p)_{d-k-1} \mid [p,q)_k \in \mathrm{Per}_{\mathbb{F}}(T(-\mathcal{I}^P))\} \setminus \{[\min \mathcal{I}, N)_0\}\}$$

And the last equality is deduced from Poincaré duality and the symbol just represent the direction so they are still isomorphic. □

This part is purely theoretical and now we have reached the result that allow us to compute the persistence of homology groups using their own dual. This will be a powerful tool to simplify the calculation and then we will do some practical examples in the next chapter.

# Chapter 5

# Practical experiment using codes

In this chapter we will use some Python codes to accomplish several objectives related to our purpose of the work.

1. To check the correctness of the duality theorem we proved in the last section of chapter 4 with some practical examples.

2. First, we use one construction method and one filtration method to calculate $H_0$, $H_1$, and $H_2$. Then, we use the duality theory to calculate $H_0$, $H_1$ and use another construction (with padding function) and another direction of filtration to calculate $H_0$ to calculate $H_2$. Finally, we compare the efficiency and speed of these two calculation methods.

3. Use the more efficient algorithm obtained above to calculate the topological features of the image, and then try to use simple statistical methods to classify and distinguish based on these results.

The Python codes we use to achieve these objectives are created by Takeki Sudo and Kazushi Ahara, modified by Shizuo Kaji. More details can be found in their paper [7].

The fundamental of this algorithm is highly related to the paper [2] which is also the theoretical basis of this work.

## 5.1 Some examples using Python codes

We will check the correctness of the duality theorem with some practical examples.

The first example is easy to see and can be proved manually, just a variation of the figure 3.3. We consider an image with numerical grayscale matrix

$$\begin{pmatrix} 1 & 4 & 5 \\ 3 & 2 & 9 \\ 2 & 4 & 7 \end{pmatrix}$$

The homology groups barcodes are the following in the order V-construction, T-construction both in positive filtration and padded T-construction in negative filtration.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 3 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 2 | 3 | 2 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 1.79769e+308 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 5.1: V-construction

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1.79769e+308 | 1 | 1 | 0 | 0 | 0 | 0 |

Figure 5.2: T-construction

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | -1.79769e+308 | 1.79769e+308 | 4 | 4 | 0 | 4.29497e+09 | 4.29497e+09 | 0 |
| 1 | 1 | -3 | -2 | 1 | 0 | 0 | 1 | 1 | 0 |
| 2 | 1 | -3 | -2 | 1 | 0 | 0 | 2 | 0 | 0 |
| 3 | 1 | -1.79769e+308 | -1 | 4.29497e+09 | 0 | 0 | 0 | 0 | 0 |

Figure 5.3: Padded T-construction

The first column indicates the dimension of homology group, the second and third represent birth and death. The columns $3, 4, 5$ and $6, 7, 8$ are the coordinates of creator and destroyer cells in the matrix (or array). So that we have

1. V-construction: $H_0$: [2,3), [2,3) and [1,$\infty$)
2. T-construction: $H_0$: [1,$\infty$)
3. Padded T-construction: $H_0$: $[-\infty, \infty)$; $H_1$: [-3,2), [-3.2) and [-$\infty$,-1)

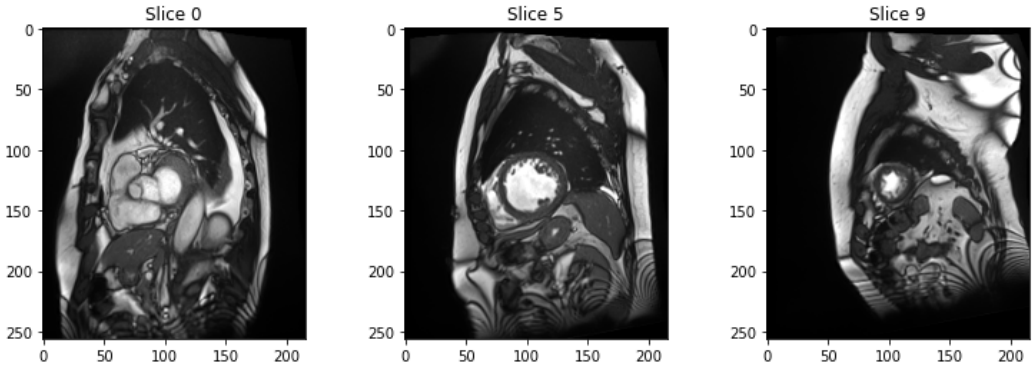
We display this example in order to point out two things. The first is that the two-dimensional coordinates of creator and destroyer cells are exchanged in the coordinates of their dual if just have one unique creator cell and one unique destroyer cell. The second is the elimination of essential homology groups. For the reason we can see the last row of padded T-construction has creator cell with coordinate $(-\infty, 0, 0)$, in fact is equal to $(-N, 0, 0)$. But the corresponding dual is the last row of V-construction with the same coordinates for creator and death cells (both are $(0, 0, 0)$). So that this dual pair of homology group must be omitted

in the analysis. The other reason to take off them because it will not be possible to process homology groups with $\infty$ birth and death, specially for computing persistence entropy and total persistence as we introduced in 5.1

The second example came from the data set *Data for Automated Cardiac Diagnosis Challenge (ACDC) [Segmentation Task]* of MIT, it is a set of digitally scanned three-dimensional figures, for more details you can see their page in kaggle.

As they are $3D$ graphs, we need to slice them into pieces and transform each of them into a grayscale numerical matrix then construct a graph with volume combining all its slices.

There is no need to display all slices of a graph, here we just submit three slices of a $3D$ graph of a patient.




(a) (b) (c)

Figure 5.4: Three slices of a 3D graph

We used the code displayed in appendix and calculated the number of $H_0$ is 17226, $H_1$ is 19829, $H_2$ is 3520 eliminating the essential ones. The dual relation is still correct and we will not put all these results.

## 5.2 Velocity of computation

Then we will compare the computation velocity between the

1. The algorithm $(1)$ computes all $H_0, H_1, H_2$ that only use one filtration without padding
2. The algorithm $(2)$ that computes first $H_0, H_1$ using a V-construction then computes $H_0$ using padded T-construction with padding.

If we just compare the difference of the costed time for one patient, the sample is too restricted to analyze. So we compute the time for all hundred patients and

compute for several times and take the average to compensate for the instability of the running speed of a computer because the efficiency also depend on the status of computer.

We have tried five times for each algorithm, and the final result is

1. The algorithm (1) use averagely 290.6901940345764 seconds for computing all homology groups in one time.
2. The algorithm (2) use averagely 278.4182926177979 seconds for computing separately homology groups using duality.

So that the second algorithm using duality is more efficient than the first one, the speed of computation has accelerated by early 4.4%. (In the best time also with five tries but not recorded, we have reached 8% but it is still too small.)

The overall speed improvement is not as much as expected. For simplicial complexes, calculating $H_2$ is very expansive and slow, requiring a lot of computing power, while calculating $H_0$ and $H_1$ is much simpler. However, the current speed improvement is obviously not so extreme. This may be because the calculation of cubic complexes itself is relatively simple, so there is not such a big difference. Another possible reason is that each of our cross-section slice is too small, most of them are less than $300 \times 300$ pixels.

## 5.3 Discriminant classification using topological features

We are going to use the the data set *Brain Tumor Classification (MRI)* with page in kaggle

https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri.

This dataset consists of a series of two-dimensional grayscale digital scanned images. These images record the brains of normal people and patients with brain tumors. The entire dataset has four categories of data, including normal brain images, brain images with meningioma tumor, glioma tumor or pituitary tumor.

We did not use the entire dataset, but selected 234 images from it, including 117 normal brain images, from image(14).jpg to image(66).jpg. Of course, these two images are also included.

The remaining 117 images were selected from brain images of patients with meningioma tumors. I selected 117 images from m(2).jpg to m(118).jpg (including the first and last two images).

As mentioned at the beginning of this chapter, our third goal is to use the more efficient (although only 4% improvement) algorithm found in the second goal

to calculate the homology group relatively easily. Then, we can calculate more topological features through the persistence of the calculated homology group, such as the total persistence and persistence entropy mentioned in the introduction of Chapter 1.

Our plan is to first calculate the $H_0$ total persistence, $H_1$ total persistence, $H_0$ persistence entropy and $H_1$ persistence entropy of each $2D$-image, and in addition to the four columns representing the topological features, give the classification of each image in the fifth column. For example, the value of the last column of a normal human brain image is equal to zero, while the value of the last column of a brain with a tumor is equal to one. This is also the standard for our classification judgment through the logistic regression model later.

When we calculated $H_0$ and $H_1$ for each image, we used the duality principle, that is, we only used V-construction to calculate $H_0$ and then calculated $H_0$ by Padded T-constructions to get the dual of $H_1$. For such a large number of images, using duality still makes the calculation faster.

We have included all the detailed calculation process and the codes used in the appendix starting from the page 53. Here we will briefly introduce the logistic regression and why I use it here.

**Definition 5.1.** *Logistic regression*, in statistics, is a log-odds model. The probability of an event occurring is modeled by making the log-odd of the event a linear combination of one or more independent variables.

Formally, in binary logistic regression, there is a binary dependent variable, encoded by an indicator variable labeled "0" and "1" (as our case), and the independent variables can each be a binary discrete variable (two classes, represented by the indicator variable) or a continuous variable (with any real value) represents the probability of happen.

And the logistic function is given as

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}} \tag{5.1}$$

We used logistic regression for discriminative classification because our situation is binary separable, and this regression model is not a black box algorithm like other machine learning methods. Each step of our process can be clearly defined and explained. In the end, after we found the topological features of all 234 images and classified them, we split them into two-thirds of the training set and the remaining one-third of the test set. We then used it to train the logistic model and compared our predicted classification results with the actual classification in the test set, and found that the accuracy of our model reached an astonishing 99%.

The specific calculation results can be restored using the code in the appendix and the 234 pictures we just mentioned, or you can directly look at the results at the end of the appendix. What we want to show is that the topological characteristics of a graph (or other object) contain a lot of information, which can enable us to obtain more accurate results than traditional statistical methods. Therefore, it is particularly necessary to find an algorithm for quickly calculating homology groups, which is one of the purposes of this article.

# Chapter 6

# Conclusion

This work has many purposes, but the core is to deform the original electronic scanned image so that we can construct filtration from two opposite directions, and finally obtain the duality relation about homology group and its persistence at the end of Chapter 4. This duality relation allows us to skip some more complicated calculations and perform simpler and faster operations through duality. Although we finally found that in most cases, especially for image datasets with smaller size and fewer images, the optimization of speed by duality calculation is not very significant.

Another goal of this paper is to obtain faster and more accurate results than traditional statistical methods based on the fast calculation of homology groups and their persistence. In the last section, we successfully used topological features based on $H_0$ and $H_1$ and obtained an amazing accuracy of nearly 99% through non-black box logistic statistical modeling, which is extremely rare in traditional statistical analysis. This fact once again proves the advantages of topological data analysis and the necessity of finding more efficient algorithms.

# Bibliography


[1] Marina Anguas Escobar, *Integrating topological features to enhance cardiac disease diagnosis from 3d cmr images*, (2023).

[2] Bea Bleile, Adélie Garin, Teresa Heiss, Kelly Maggs, and Vanessa Robins, *The persistent homology of dual digital image constructions*, Research in Computational Topology 2, Springer, 2022, pp. 1–26.

[3] David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov, *Vines and vineyards by updating persistence in linear time*, Proceedings of the twenty-second annual symposium on Computational geometry, 2006, pp. 119–126.

[4] Junhan Cui, *Dimensionality reduction based on persistence entropy*, (2023).

[5] Edelsbrunner, Letscher, and Zomorodian, *Topological persistence and simplification*, Discrete & Computational Geometry **28** (2002), 511–533.

[6] Allen Hatcher, *Algebraic Topology*, 2005.

[7] Shizuo Kaji, Takeki Sudo, and Kazushi Ahara, *Cubical ripser: Software for computing persistent homology of image and volume data*, arXiv preprint arXiv:2005.12692 (2020).

[8] Matteo Rucco, Filippo Castiglione, Emanuela Merelli, and Marco Pettini, *Characterisation of the idiotypic immune network through persistent entropy*, Proceedings of ECCS 2014: European Conference on Complex Systems, Springer, 2016, pp. 117–128.

[9] Claude Elwood Shannon, *A mathematical theory of communication*, The Bell system technical journal **27** (1948), no. 3, 379–423.

[10] Afra Zomorodian and Gunnar Carlsson, *Computing persistent homology*, Proceedings of the twentieth annual symposium on computational geometry, 2004, pp. 347–356.

# Python codes we used and some calculation results

```
1
2
3  import cripser, tcripser
4  import numpy as np
5
6  #To compute the first example with a matrix 3 * 3.
7
8  a= np.matrix([[1,4,5],[3,2,9],[2,4,7]])
9  pd = cripser.computePH(a,maxdim=2)
10 pd_t = tcripser.computePH(a,maxdim=2)
11 pd_t_padded=tcripser.computePH(a,maxdim=2,embedded=True)
12
13
14 import h5py
15
16 ##To visualize the slices of a 3D graph in h5 format.
17
18 def visualize_h5(file_name, dataset_name):
19     with h5py.File(file_name, 'r') as f:
20         if dataset_name in f:
21             data = f[dataset_name][...]
22         else:
23             print(f"Dataset {dataset_name} not found in the
                file.")
24             return
25
26     print(f"Data shape: {data.shape}")
27
28     # If dim == 3 with three components (num_slices, height,
       width)
29
```

```
    # We just need three slices of them

    if data.ndim == 3:
        num_slices = data.shape[0]
        print(f"Number of slices: {num_slices}")

        plt.imshow(data[0], cmap='gray')
        plt.title('Slice 0')
        plt.show()

        mid_slice = num_slices // 2
        plt.imshow(data[mid_slice], cmap='gray')
        plt.title(f'Slice {mid_slice}')
        plt.show()

        plt.imshow(data[-1], cmap='gray')
        plt.title(f'Slice {num_slices - 1}')
        plt.show()

    ##If the image is two-dimensional then we just have this.

    elif data.ndim == 2:
        plt.imshow(data, cmap='gray')
        plt.title('2D Image')
        plt.show()

#To extract the image data set from h5 file f.

def image_data_h5(file_name, dataset_name):
    with h5py.File(file_name, 'r') as f:
        if dataset_name in f:
            data = f[dataset_name][...]
            return data
        else:
            print(f"Dataset {dataset_name} not found in the
                file.")
            return


#To visualize the slices of the frame 01 of the first patient.

file_name = 'patient001_frame01.h5'
dataset_name = 'image'
visualize_h5(file_name, dataset_name)
```

```
73
74
75  #To compute the velocity of computation we need to calculate
        the image data for all patients.
76
77  import timeit,time
78  import os
79
80  folder_path = 'D:\TFM\ACDC_preprocessed\ACDC_training_volumes'
81  file_list = [os.path.join(file) for file in os.listdir(
        folder_path) if os.path.isfile(os.path.join(folder_path,
        file))]
82
83  all_image_data = []
84
85  for file_name in file_list:
86
87      image_data= image_data_h5(file_name, 'image')
88      all_image_data.append(image_data)
89
90
91
92  #To compute the time for the first algorithm for hundred
        patients and each one has two frames.
93
94  start_time = time.time()
95
96  for image_data in all_image_data:
97
98      #To compute all homology groups using V-construction
            without padding.
99
100     results = cripser.computePH(image_data,maxdim=2)
101
102 end_time = time.time()
103
104 # To compute the running time.
105
106 elapsed_time_no_efficient = end_time - start_time
107 print(elapsed_time_no_efficient)
108
109
110
111 #Times for five tries.
```

```
112
113 291.20752716064453+292.75494146347046+286.75698947906494
114 +293.9577124118805+288.77379965782166
115
116
117
118
119 #To compute the time for the second algorithm for hundred
        patients and each one has two frames.
120
121 start_time = time.time()
122
123 for image_data in all_image_data:
124
125     ##To compute homology groups without H2 using V-
            construction without padding.
126
127     results_v_no2 = cripser.computePH(image_data,maxdim=1)
128
129     ##To compute H0 using padded T-construction.
130
131     results_tp_0 = tcripser.computePH(image_data,maxdim=0,
            embedded=True)
132
133 end_time = time.time()
134
135 #To compute the running time.
136
137 elapsed_time_efficient = end_time - start_time
138 print(elapsed_time_efficient)
139
140 #Times for five tries.
141
142 279.1985194683075+278.0140290260315+279.47413992881775
143 +279.53665685653687+275.86811780929565
144
145
146
147
148 # For the last section discriminant classification using
        topological features.
149
150 from PIL import Image
151 import matplotlib.pyplot as plt
```

```
import os

folder_path = 'D:\\TFM\\brain_tumor_classification\\normal'

file_list = [os.path.join(file) for file in os.listdir(
    folder_path) if os.path.isfile(os.path.join(folder_path,
    file))]

image_normal=[]
image_data_normal=[]

for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)

    image = Image.open(file_path)

    gray_image = image.convert('L')


    image_normal.append(gray_image)
    image_data_normal.append(np.array(gray_image))









folder_path = 'D:\\TFM\\brain_tumor_classification\\
    meningioma_tumor'

file_list = [os.path.join(file) for file in os.listdir(
    folder_path) if os.path.isfile(os.path.join(folder_path,
    file))]

image_tumor=[]
image_data_tumor=[]

for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)

    image = Image.open(file_path)

    gray_image = image.convert('L')
```

```
191
192     image_tumor.append(gray_image)
193     image_data_tumor.append(np.array(gray_image))
194
195
196
197 #####The following is for normal#########################
198
199 # Here are the H0 and H1 of each picture.
200
201 result_normal_0 = []
202
203 result_normal_1 = []
204
205
206 for img_array in image_data_normal:
207
208     normal_0 = cripser.computePH(img_array,maxdim=0)
209     normal_1 = tcripser.computePH(img_array,maxdim=0,embedded=
            True)
210
211     result_normal_0.append(normal_0)
212     result_normal_1.append(normal_1)
213
214
215 # Eliminate rows with infinite
216
217 result_normal_cleaned_0 = []
218
219
220 for array in result_normal_0:
221     # Use boolean indexing to remove rows that contain values
            â â greater than 255 or less than 0 in the first and
            second columns
222
223     valid_rows = (array[:, 1] <= 255) & (array[:, 1] >= 0) & (
            array[:, 2] <= 255) & (array[:, 2] >= 0)
224     cleaned_array = array[valid_rows]
225
226     result_normal_cleaned_0.append(cleaned_array)
227
228
229
230
```

```
231
232 # First convert all negative signs to positive and then swap
        the positions of birth and death
233
234 for array in result_normal_1:
235
236     array[:, 1] *= -1
237     array[:, 2] *= -1
238
239     array[:, [1, 2]] = array[:, [2, 1]]
240
241 result_normal_cleaned_1 = []
242
243
244 for array in result_normal_1:
245
246     valid_rows = (array[:, 1] <= 255) & (array[:, 1] >= 0) & (
            array[:, 2] <= 255) & (array[:, 2] >= 0)
247     cleaned_array = array[valid_rows]
248
249     result_normal_cleaned_1.append(cleaned_array)
250
251
252
253
254 ###############Total persistence and persistence entropy of H0
        for normal images
255
256 # Create an empty column matrix with 117 rows and 1 column,
        using np.zeros
257
258 total_persistence_matrix_0 = np.zeros((117, 1))
259
260
261 for i in range(117):
262 # Calculate the difference between the third column and the
        second column of each row
263
264     differences = result_normal_cleaned_0[i][:, 2] -
            result_normal_cleaned_0[i][:, 1]
265
266 # Add these differences to get a number
267
268     result = np.sum(differences)
```

```
269
270     total_persistence_matrix_0[i]=result
271
272
273 persistence_entropy_matrix_0 = np.zeros((117, 1))
274
275 import math
276 from math import log2
277
278 for i in range(117):
279
280     differences=0
281
282     for j in range(result_normal_cleaned_0[i].shape[0]):
283
284         differences += (-1)*((result_normal_cleaned_0[i][j, 2]
                - result_normal_cleaned_0[i][j, 1])/
                total_persistence_matrix_0[i,0]) * log2(((
                result_normal_cleaned_0[i][j, 2] -
                result_normal_cleaned_0[i][j, 1])/
                total_persistence_matrix_0[i,0]))
285
286
287     persistence_entropy_matrix_0[i]=differences
288
289
290 ###############Total persistence and persistence entropy of H1
    for normal images
291
292
293 total_persistence_matrix_1 = np.zeros((117, 1))
294
295
296 for i in range(117):
297
298     differences = result_normal_cleaned_1[i][:, 2] -
          result_normal_cleaned_1[i][:, 1]
299
300     result = np.sum(differences)
301     total_persistence_matrix_1[i]=result
302
303
304 persistence_entropy_matrix_1 = np.zeros((117, 1))
305
```

```
306
307
308 for i in range(117):
309
310     differences=0
311
312     for j in range(result_normal_cleaned_1[i].shape[0]):
313
314         differences += (-1)*((result_normal_cleaned_1[i][j, 2]
                - result_normal_cleaned_1[i][j, 1])/
                total_persistence_matrix_1[i,0]) * log2(((
                result_normal_cleaned_1[i][j, 2] -
                result_normal_cleaned_1[i][j, 1])/
                total_persistence_matrix_1[i,0]))
315
316
317     persistence_entropy_matrix_1[i]=differences
318
319
320
321 ##Create an identification column matrix, the column for normal
        graphs is all 0.
322
323 class_normal_matrix = np.zeros((117, 1))
324
325 ##Merge the first four columns, that is, four different
       topological features, and the last column together.
326
327
328 topo_features_normal_matrix = np.hstack((
       total_persistence_matrix_0, total_persistence_matrix_1,
       persistence_entropy_matrix_0, persistence_entropy_matrix_1,
       class_normal_matrix))
329
330
331
332
333
334 #####The following is for tumor#########################
335
336 # We can actually do the same thing just replacing 'normal' by
       'tumor' in the names.
337
338
```

```
result_tumor_0 = []

result_tumor_1 = []


for img_array in image_data_tumor:

    tumor_0 = cripser.computePH(img_array,maxdim=0)
    tumor_1 = tcripser.computePH(img_array,maxdim=0,embedded=
        True)

    result_tumor_0.append(tumor_0)
    result_tumor_1.append(tumor_1)



result_tumor_cleaned_0 = []


for array in result_tumor_0:
    valid_rows = (array[:, 1] <= 255) & (array[:, 1] >= 0) & (
        array[:, 2] <= 255) & (array[:, 2] >= 0)
    cleaned_array = array[valid_rows]

    result_tumor_cleaned_0.append(cleaned_array)







for array in result_tumor_1:
    array[:, 1] *= -1
    array[:, 2] *= -1
    array[:, [1, 2]] = array[:, [2, 1]]

result_tumor_cleaned_1 = []


for array in result_tumor_1:
    valid_rows = (array[:, 1] <= 255) & (array[:, 1] >= 0) & (
        array[:, 2] <= 255) & (array[:, 2] >= 0)
    cleaned_array = array[valid_rows]

```

```
    result_tumor_cleaned_1.append(cleaned_array)




###############Total persistence and persistence entropy of H0
    for tumor images


total_persistence_tumor_matrix_0 = np.zeros((117, 1))


for i in range(117):
    differences = result_tumor_cleaned_0[i][:, 2] -
        result_tumor_cleaned_0[i][:, 1]
    result = np.sum(differences)
    total_persistence_tumor_matrix_0[i]=result


persistence_entropy_tumor_matrix_0 = np.zeros((117, 1))

import math
from math import log2

for i in range(117):

    differences=0

    for j in range(result_tumor_cleaned_0[i].shape[0]):

        differences += (-1)*((result_tumor_cleaned_0[i][j, 2] -
            result_tumor_cleaned_0[i][j, 1])/
           total_persistence_tumor_matrix_0[i,0]) * log2(((
           result_tumor_cleaned_0[i][j, 2] -
           result_tumor_cleaned_0[i][j, 1])/
           total_persistence_tumor_matrix_0[i,0]))


    persistence_entropy_tumor_matrix_0[i]=differences


###############Total persistence and persistence entropy for H1
    of the tumor image.

```

```
416
417 total_persistence_tumor_matrix_1 = np.zeros((117, 1))
418
419
420 for i in range(117):
421     differences = result_tumor_cleaned_1[i][:, 2] -
           result_tumor_cleaned_1[i][:, 1]
422     result = np.sum(differences)
423     total_persistence_tumor_matrix_1[i]=result
424
425
426 persistence_entropy_tumor_matrix_1 = np.zeros((117, 1))
427
428
429
430 for i in range(117):
431
432     differences=0
433
434     for j in range(result_tumor_cleaned_1[i].shape[0]):
435
436         differences += (-1)*((result_tumor_cleaned_1[i][j, 2] -
               result_tumor_cleaned_1[i][j, 1])/
              total_persistence_tumor_matrix_1[i,0]) * log2(((
              result_tumor_cleaned_1[i][j, 2] -
              result_tumor_cleaned_1[i][j, 1])/
              total_persistence_tumor_matrix_1[i,0]))
437
438
439     persistence_entropy_tumor_matrix_1[i]=differences
440
441
442
443 ##Create an identification column matrix, the column for tumor
    images is all 1.
444
445 class_tumor_matrix = np.ones((117, 1))
446
447 ##Merge the first four columns, that is, four different
    topological features, and the last column together.
448
449 topo_features_tumor_matrix = np.hstack((
    total_persistence_tumor_matrix_0,
    total_persistence_tumor_matrix_1,
```

```
    persistence_entropy_tumor_matrix_0,
    persistence_entropy_tumor_matrix_1, class_tumor_matrix))
450
451
452
453 ##Now I merge the topological feature matrices of these two
    categories together to form a data set of all images and
    their topological features.
454
455
456 topo_features_all_matrix = np.vstack((
    topo_features_normal_matrix, topo_features_tumor_matrix))
457
458
459
460
461
462
463 ## Now we use logistic regression
464
465 from sklearn.linear_model import LogisticRegression
466 from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report
467 from sklearn.model_selection import train_test_split
468
469
470 labels=topo_features_all_matrix[:,4]
471
472 # Use the train_test_split function to randomly split the data
    into training and test sets
473
474 train_set, test_set, train_labels, test_labels =
    train_test_split(topo_features_all_matrix, labels, test_size
    =1/3, random_state=42)
475
476 # Create a logistic regression model
477
478 model = LogisticRegression()
479
480 # train it
481 model.fit(train_set, train_labels)
482
483 # predict the test_set
484 predictions = model.predict(test_set)
```

```
485
486
487
488 Evaluating the Model
489
490
491 accuracy = accuracy_score(test_labels, predictions)
492 conf_matrix = confusion_matrix(test_labels, predictions)
493 class_report = classification_report(test_labels, predictions)
494
495 print(f"Accuracy: {accuracy}")
496 print("Confusion Matrix:")
497 print(conf_matrix)
498 print("Classification Report:")
499 print(class_report)
500
501
502 PS: The results are
503 Accuracy: 0.9871794871794872
504 Confusion Matrix:
505 [[35  1]
506  [ 0 42]]
507 Classification Report:
508               precision    recall  f1-score   support
509
510          0.0       1.00      0.97      0.99        36
511          1.0       0.98      1.00      0.99        42
512
513     accuracy                           0.99        78
514    macro avg       0.99      0.99      0.99        78
515 weighted avg       0.99      0.99      0.99        78
```