Summer 2017

A Method for Detection of Local Dimension in Point Cloud Data

Catherine Boersma
San Jose State University

A METHOD FOR DETECTION OF LOCAL DIMENSION IN POINT CLOUD DATA

A Thesis

Presented to

The Faculty of the Department of Mathematics

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Catherine T. Boersma

August 2017

The Designated Thesis Committee Approves the Thesis Titled

A METHOD FOR DETECTION OF LOCAL DIMENSION IN POINT CLOUD DATA

by

Catherine T. Boersma

APPROVED FOR THE DEPARTMENT OF MATHEMATICS

SAN JOSÉ STATE UNIVERSITY

August 2017

Dr. Timothy Hsu	Department of Mathematics
Dr. Wasin So	Department of Mathematics
Dr. Marion Campisi	Department of Mathematics

ABSTRACT

A METHOD FOR DETECTION OF LOCAL DIMENSION IN POINT CLOUD DATA

by Catherine T. Boersma

We have invented a method that uses the mathematical idea of local homology to calculate the local dimension (one-dimensional, two-dimensional, mixed dimension of varying types) of an underlying object in two-dimensional space from a point cloud approximation. Motivated by the need to find an efficient method for computing local homology, we define a new mathematical object – the Local Complex, and some variations of this idea – that we show to be exactly related to the Vietoris-Rips complex under some settings. This concept captures the essence of the local homology of point cloud data at any scale. We provide a computationally tractable heuristic – the simplex arc projection on a particular variation of the Local Complex – the Acute Local Complex – to produce yet another object, simply called the simplex arc projections. Homology computation for this latter class of objects is then described. This homology is then related to the true local homology, by example. This relationship is not tight (as we show by counter-example), and conditions under which it holds are the subject of future extensions to this research. Our method also includes mechanisms for detecting whether a particular dimensionality analysis is mathematically more meaningful (in technical terms, “persistent”) and not just a function of a choice of parameters. We also provide implementation and experimental results on synthetic data as well as a subset of the Sloan Digital Sky Survey data.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Timothy Hsu, for his patience and prodding. His thoughtful instruction and guidance have piloted this project. I am forever indebted for the many hours he spent walking me through the background material, carefully reading through drafts, and directing every step of our research efforts.

I am also grateful to professors Dr. Wasin So and Dr. Marion Campisi for making time in their busy schedules to be readers on my committee as well as for serving as examiners for my qualifiers.

I would also like to thank Dr. Bem Cayco, who encouraged my in a very personal and meaningful way by sharing her experiences of being a full-time graduate student and a first-time mom.

Thanks to Dr. Slobadan Simić, Dr. Jeff Scargle, David Goulette, Jian-Long Liu, Joey Fitch, Matthew Litrus, Hai Nguyen Au and Brandon Morrison for collaborating in research on the spring 2014 CAMCOS project studying the structure of the cosmic web. The CAMCOS project was an inspiration for my continued study of topological data analysis methods.

I would also like to thank Jennifer Kloke at Ayasdi for tutouring me through homology when I found myself hitting my head against a wall.

I am deeply grateful to my parents, brother, sister, and sisters-in-law for their love and support.

Finally and most importantly, I would like to thank my husband, Paul Varkey, whose love, encouragement, patience and guidance made this thesis work a joy.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

Figure

CHAPTER 1

INTRODUCTION

In this thesis, we present a novel topological data analysis algorithm, motivated by the study of the shape and distribution of galaxies in the universe. We are especially motivated by the anticipated massive point cloud data sets that will be produced by the Large Synoptic Survey Telescope (LSST) galaxy survey, the size of which necessitates the availability of efficient algorithms and tools to explore and make sense of it ([ITA+08]).

1.1 Context and Problem Description

A **point cloud** is a finite set of coordinate points in a metric space. Point cloud data represents, for example, a solid object or a picture by a sampling of its points in two or three dimensions. Such data might be generated by sensors or scanners, especially distributed sensor networks, or by, for example, the detection of individual galaxies in space as representing a galaxy cluster.

Topological Data Analysis (TDA) uses methods from topology to analyze the shape of data. It has applications in sensor networks ([DSG06]), astronomy ([Sou11]), protein complexes ([XW14]), and image processing ([CCDS06]).

Many techniques have been developed for reconstructing surfaces (two-dimensional objects in space) from point cloud approximation (for example, see Dey et. al. [DGGZ02] and for a survey, see Berger et. al. [BTS+14]). In this thesis, we tackle the problem of reconstructing an object of mixed dimensionality called a simplicial complex. That is, instead of assuming that the underlying object approximated by a point cloud is three-dimensional, we assume that an underlying object is made of one-dimensional, two-dimensional, and three-dimensional pieces glued together in some fashion. Such an object has a property of "local dimension" at any particular location: for example, some locations might be two-dimensional

(surface-like); other locations might be more one-dimensional (string-like); and still others might be locations where one-dimensional and two-dimensional pieces are glued together.

More formally, let $\mathscr{X}$ be a subset of $\mathbb{R}^{\mathbb{N}}$; $\mathscr{X}$ is a space whose structure we want to understand. Let $X \subset \mathscr{X}$ be a point cloud sample of $\mathscr{X}$ that we observe. Our goal is (eventually) to estimate the topological structure of $\mathscr{X}$ by using X to build a simplicial complex $\mathscr{S}$ that is a close approximation to $\mathscr{X}$ itself.

1.1.1 Sloan Digital Sky Survey (SDSS)

The Sloan Digital Sky Survey (SDSS) data set and our previous CAMCOS work served as motivation and a point of departure for this thesis. The SDSS is optical data collected from the 2.5 meter telescope at the Apache Point Observatory in New Mexico. Each point in the SDSS catalogue is an entire galaxy. Our work seeks to determine the local dimension of each galaxy in the data set, in order to identify structures of varying dimension. These structures have been coined “The Cosmic Web" [BKP95] and are comprised of over-dense and under-dense regions. Galaxies tend to group together and form compact three-dimensional structures called clusters, as well as elongated one-dimensional filaments and sheet-like two-dimensional walls. The cosmic web fills only a fraction of all space, leaving large, mostly empty regions called voids. In our analysis, we work with a thin rectangular subset of the SDSS flattened to two dimensions. The flattened subset considered in this work consists of 3357 galaxies.

Galaxy clusters were first identified in the 1970s as large collections of galaxies spanning hundreds of millions of light years. Many cluster-finding algorithms have been developed to identify three-dimensional regions ([KKM$^+$11]). Most studies of the cosmic web focus on one type of structure at a time (like cataloging filaments). There are a few approaches which use topological data analysis to analyze all the various

structures of the cosmic web (see Sousbie [Sou11]).

1.2 Main contribution : local homology heuristics

Our contributions involve the construction of a new mathematical object closely related to the Vietoris-Rips complex (VR complex), the Local Complex, along with a few variants, as well as the development of a novel heuristic algorithm on this complex that seeks to recover the local homology at any point, at any scale.

Our approach is parametric with respect to both the scale and the locality at which the data is algorithmically studied. Our results then provide a parameter-free way of scanning and automatically selecting interesting parameters at which to observe the data.

We also provide implementation and experimental results on synthetic data as well as a subset of the SDSS data.

These main contributions of this thesis are summarized and illustrated in Figure 1.2.1. The path highlighted in gray depicts the main algorithmic phases of our results. In the algorithm, we are trying to approximate the computation of the local homology of the VR complex. The Local Complex (Definition 5.2.1) approximates the VR complex by Theorem 5.2.2. The Local Complex is presented with three variations: the Local Shell Complex (Definition 5.2.5), the Local Central Hole Complex (Definition 5.2.6), and the Acute Local Complex (Definition 5.2.7). We apply the simplex arc projection map (Definition 6.1.1) to the Acute Local Complex to form the simplex arc projections. Then we count the components in the simplex arc projections, as a heuristic means of identifying its homology, yielding a result which is related to the local homology of a vertex in the VR complex (as explained in Definition 6.1.1 and Example 6.1.3).

The Local Complex depends on parameters β (scale) and α (locality). We iterate through the highlighted path for a wide range of α and β and then determine the values

of α and β where the homology coloring remains stable as the parameters change. We then use the heuristic local homologies at these (α, β) values to label and characterize the data. We show through examples that these results correspond to human intuition about the local dimension of both synthetic and real data.

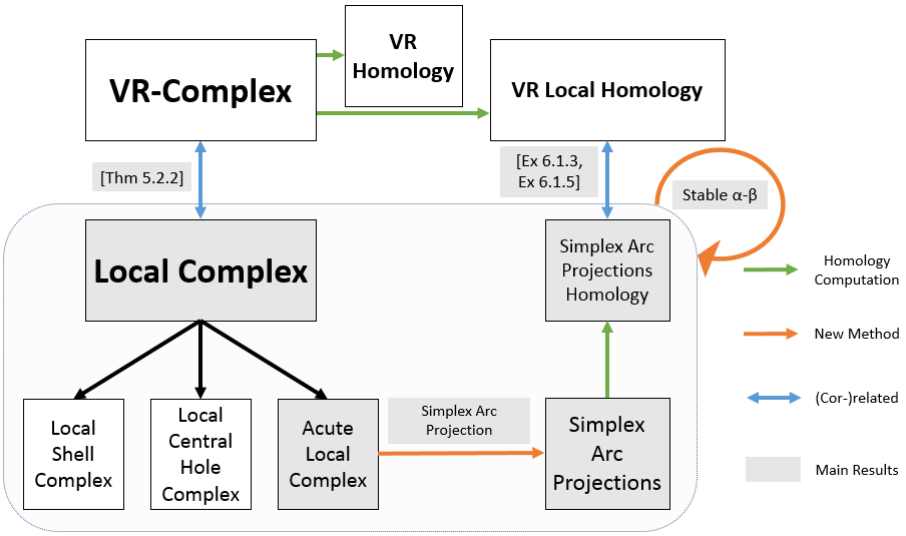


Figure 1.2.1: The main contributions of the thesis.

1.3 Organization of the thesis

The thesis is organized as follows: chapters 2 and 3 contain background information about algebraic topology and homology. We continue the foundational material with chapter 4, describing homotopy, exact sequences and local homology.

Chapter 5 contains the main conceptual contribution of this thesis – the local complex, presented against the backdrop of simplicial complexes of point cloud data.

In chapter 6, we define the simplex arc projection and through example, show how it is related to local homology. This serves as motivation for using the simplex arc projection as a heuristic for local homology.

Chapter 7 describes the details of the main algorithm – the computation of the Acute Local Complex and the simplex arc projection.

Finally, in chapter 8, we present the findings of the algorithm applied to both synthetic data sets and the SDSS galaxy data.

CHAPTER 2

ALGEBRAIC TOPOLOGY

We begin with a review of algebraic topology to motivate and justify the development of our algorithm. The discussion of the background material in chapters 2–4 borrows heavily from Hatcher [Hat02] and Munkres [Mun84] for algebraic topology. The introductory material is also inspired by Zomorodian [ZC05], who provides a clear introduction to computational topology.

2.1 Convex sets in $\mathbb{R}^N$

Definition 2.1.1. A set C in $\mathbb{R}^N$ is **convex** if for any two points x and y in C, the point $(1-t)\,x + t\,y$ is also in C, for all $t \in [0,1]$.

Theorem 2.1.2. *The intersection of convex sets is convex.*

Proof. Let $\mathscr{C}$ be a family of convex subsets of vector space V. Let $x, y \in \bigcap \mathscr{C}$. Then for every $C \in \mathscr{C}$, $x, y \in C$ by definition of set intersection. So for every $t \in [0,1]$, $(1-t)\,x + t\,y \in C$. Therefore $(1-t)\,x + t\,y \in \bigcap \mathscr{C}$ for every $t \in [0,1]$. So $\bigcap \mathscr{C}$ is convex. □

Definition 2.1.3. The **convex hull** of a set X is the smallest convex set which contains X; i.e. the convex hull is the intersection of all convex sets containing X (by Theorem 2.1.2).

2.2 Simplices in $\mathbb{R}^N$

Definition 2.2.1. Given a set $S = \{a_0, a_1, a_2, ..., a_n\}$ of points in $\mathbb{R}^N$, S is **geometrically independent** if for any $t_i \in \mathbb{R}$, whenever $\sum_{i=0}^{n} t_i = 0$ and $\sum_{i=0}^{n} t_i a_i = 0$ then $t_0 = t_1 = ... = t_n = 0$.

Theorem 2.2.2. *The set $S_0 = \{a_0, a_1, a_2, ..., a_n\}$ is geometrically independent if and only if the set $S_1 = \{a_1 - a_0, ..., a_n - a_0\}$ is linearly independent.*

Proof. Suppose $a_1 - a_0, ..., a_n - a_0$ are linearly independent. Suppose also that $t_0 a_0 + t_1 a_1 + ... + t_n a_n = 0$ and $t_0 + t_1 + ... + t_n = 0$. Then $t_0 = -t_1 - t_2 - ... - t_n$, and

$$(-t_1 - t_2 - ... - t_n) a_0 + t_1 a_1 + ... + t_n a_n = 0.$$

Therefore $t_1 (a_1 - a_0) + ... + t_n (a_n - a_0) = 0$, and $t_0 = t_1 = ... = t_n = 0$ because $a_1 - a_0, ..., a_n - a_0$ are linearly independent. Hence $a_0, a_1, ..., a_n$ is geometrically independent.

Now suppose $\{a_0, a_1, ..., a_n\}$ is geometrically independent. Suppose also that

$$t_1 (a_1 - a_0) + t_2 (a_2 - a_0) + ... + t_n (a_n - a_0) = 0.$$

Then $\sum_{i=1}^{n} t_i a_i - \sum_{i=1}^{n} t_i a_0 = 0$. Let $s_0 = -\sum_{i=1}^{n} t_i$, $s_i = t_i$ for $1 \le i \le n$. Then we have $\sum_{i=0}^{n} s_i a_i = 0$ and $\sum_{i=0}^{n} s_i = 0$. Then $s_0 = t_1 = ... = t_n = 0$ since $\{a_0, a_1, ..., a_n\}$ is geometrically independent. Therefore $a_1 - a_0, ..., a_n - a_0$ are linearly independent. □

Definition 2.2.3. Let $a_0, a_1, ..., a_n \in \mathbb{R}^N$ be geometrically independent. The n-**simplex** σ spanned by $\{a_0, a_1, ..., a_n\}$ is the set

$$\sigma = \left\{ t_0 a_0 + ... + t_n a_n \,\middle|\, t_i \ge 0, \sum_{i=0}^{n} t_i = 1 \right\}. \tag{2.1}$$

Example 2.2.4. Figure 2.2.1 illustrates examples of simplices of various dimensions.

Definition 2.2.5. Given a geometrically independent set $S = \{a_0, a_1, a_2, ..., a_n\}$, the n-**plane** P spanned by S consists of all $x \in \mathbb{R}^N$ such that $x = \sum_{i=0}^{n} t_i a_i$ for some $t_i \in \mathbb{R}$ with $\sum_{i=0}^{n} t_i = 1$.

Example 2.2.6. Figures 2.2.2 and 2.2.3 demonstrate the relationship between the n-simplex and n-plane spanned by $\{a_0, a_1, a_2\}$.

Theorem 2.2.7. *Let $a_0, a_1, ..., a_n \in \mathbb{R}^N$ be geometrically independent. Let σ be the n-simplex*

$$\sigma = \left\{ t_0 a_0 + ... + t_n a_n \,\middle|\, t_i \ge 0, \sum_{i=0}^{n} t_i = 1 \right\}.$$

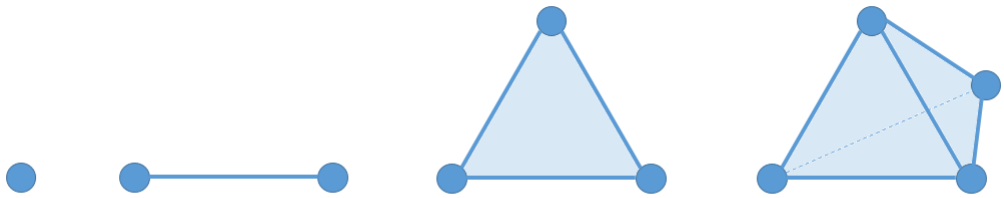


Figure 2.2.1: A 0-simplex, a 1-simplex, a 2-simplex and a 3-simplex.

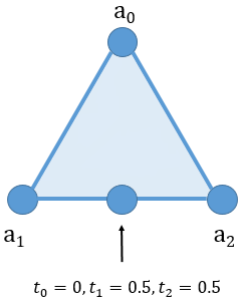


Figure 2.2.2: A 2-simplex.

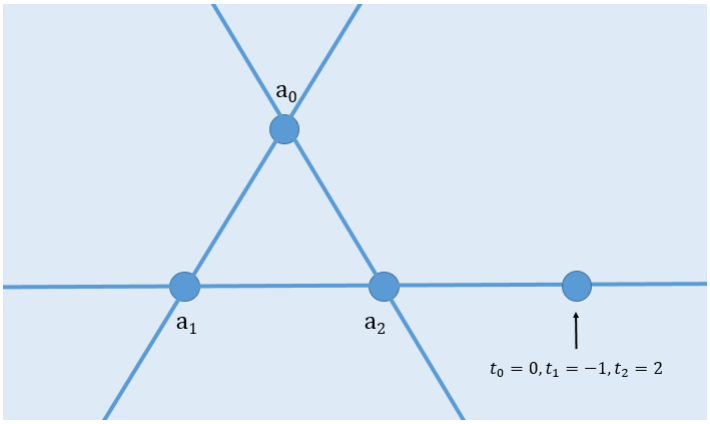


Figure 2.2.3: A 2-plane.

Then the t_i are uniquely determined by x.

Proof. Let $a_0, a_1, ..., a_n \in \mathbb{R}^N$ be geometrically independent. Suppose we have two sets of coefficients, t_i and s_i which determine the same point x in simplex σ. In other words, $t_i \geq 0$, $\sum_{i=0}^{n} t_i = 1$, $s_i \geq 0$, $\sum_{i=0}^{n} s_i = 1$, and

$$x = t_0 a_0 + ... + t_n a_n = s_0 a_0 + ... + s_n a_n. \tag{2.2}$$

Therefore, $(t_0 - s_0) a_0 + ... + (t_n - s_n) a_n = 0$ and $\sum_{i=0}^{n} (t_i - s_i) = 0$, and since $a_0, a_1, ..., a_n$ are geometrically independent, $(t_0 - s_0) = ... = (t_n - s_n) = 0$. So $t_i = s_i$ for all i. □

Definition 2.2.8. Let $a_0, a_1, ..., a_n \in \mathbb{R}^N$ be geometrically independent. Let σ be the n-simplex in (2.2). Then the t_i are the **barycentric coordinates** of x of σ with respect to $\{a_0, a_1, a_2, ..., a_n\}$.

Definition 2.2.9. Let σ be an n-simplex. The points $\{a_0, a_1, a_2, ..., a_n\}$ that span σ are the **vertices** of σ and n is the **dimension** of σ. Any simplex spanned by a nonempty subset of $\{a_0, a_1, a_2, ..., a_n\}$ is a **face** of σ. The face spanned by $\{a_1, a_2, ..., a_n\}$ is the **face opposite** a_0. Faces different from σ are called **proper faces**. The union of proper faces of σ is the **boundary** of σ, denoted by $\mathrm{Bd}(\sigma)$. The **interior** of σ is denoted $\mathrm{Int}(\sigma)$ and is equal to $\sigma - \mathrm{Bd}(\sigma)$.

Definition 2.2.10. An **affine transformation** T of $\mathbb{R}^N$ is a map that is a composition of translations and non-singular linear transformations on $\mathbb{R}^N$.

Theorem 2.2.11. *Let T be an affine transformation on $\mathbb{R}^N$. Then*

(1) T preserves barycentric coordinates (if $x = \sum t_i a_i$ then $T(x) = \sum t_i T(a_i)$) and therefore

(2) T preserves planes spanned by $\{a_0, a_1, a_2, ..., a_n\}$,

(3) T preserves lines through a_0 and a_1,

(4) T preserves convex sets,

(5) T preserves simplexes spanned by $\{a_0, a_1, a_2, ..., a_n\}$, and

(6) T preserves geometrically independent sets.

Proof. We begin by proving (1):

Let $x = \sum t_i a_i$. First suppose T is a translation. Then

$$T(x) = x + c = \sum t_i a_i + c.$$

Since the t_i are barycentric coordinates, $\sum t_i = 1$, so

$$T(x) = \sum t_i a_i + c \sum t_i = T(x) = \sum t_i (a_i + c) = \sum t_i T(a_i).$$

Now suppose T is a non-singular linear transformation. Then

$$T(x) = T \sum t_i a_i = T(t_0 a_0) + ... + T(t_n a_n) = t_0 T(a_0) + ... + t_n T(a_n) = \sum t_i T(a_i).$$

Since any affine T is a composition of translations and non-singular linear transformations, (1) follows. To prove (2), note that the plane through $\{a_0, a_1, ..., a_n\}$ is precisely the set of all points x in $\mathbb{R}^N$ such that $x = \sum_{i=0}^{n} t_i a_i$ for some $t_i \in \mathbb{R}$ with $\sum_{i=0}^{n} t_i = 1$. The plane through $\{T(a_0), T(a_1), ..., T(a_n)\}$ is the set of all points x in $\mathbb{R}^N$ such that $x = \sum_{i=0}^{n} t_i T(a_i)$ for some $t_i \in \mathbb{R}$ with $\sum_{i=0}^{n} t_i = 1$. By (1), T sends the plane through $\{a_0, a_1, ..., a_n\}$ to the plane through $\{T(a_0), T(a_1), ..., T(a_n)\}$.

To prove (3), note that the line through $\{a_0, a_1\}$ is simply a special case of the plane through $\{a_0, a_1, ..., a_n\}$ when $n = 1$. So by (2), T preserves lines through $\{a_0, a_1\}$.

To prove (4), let A be a convex set and let $x_0, x_1 \in A$. Then by Definition 2.1.1, the 1-simplex $\{t x_0 + (1-t) x_1 | 0 \leq t \leq 1\} \in A$. By part (1),

$$T(\{t x_0 + (1-t) x_1\}) = \{t T(x_0) + (1-t) T(x_1)\}.$$

So $T([x_0x_1]) \in T(A)$ and therefore T preserves convex sets.

To prove (6), suppose that T is an invertible linear transformation. Let $\{a_0, a_1, a_2, ..., a_n\}$ be geometrically independent. Then by Theorem 2.2.2, $\{a_1 - a_0, ..., a_n - a_0\}$ is linearly independent. Now suppose we have $\{b_0, b_1, ..., b_n\}$ such that

$$T(b_1(a_1 - a_0) + ... + b_n(a_n - a_0)) = b_1T(a_1 - a_0) + ... + b_nT(a_n - a_0) = 0.$$

Since T is injective and $T(0) = 0$, we have

$$b_1(a_1 - a_0) + ... + b_n(a_n - a_0) = 0.$$

Since $\{a_1 - a_0, ..., a_n - a_0\}$ is linearly independent, $b_1 = b_2 = ... = b_n = 0$. So $\{T(a_1 - a_0), ..., T(a_n - a_0)\}$ is linearly independent.

Now suppose T is a translation and $\{a_0, a_1, a_2, ..., a_n\}$ is geometrically independent. Then $T(\{a_0, a_1, a_2, ..., a_n\}) = \{a_0 + c, a_1 + c, a_2 + c, ..., a_n + c\}$. So by Theorem 2.2.2,

$$\{a_1 + c - (a_0 + c), a_2 + c - (a_0 + c), ..., a_n + c - (a_0 + c)\} = \{a_1 - a_0, a_2 - a_0, ..., a_n - a_0\}$$

are linearly independent, and therefore $\{a_0, a_1, a_2, ..., a_n\}$ are geometrically independent.

To prove (5), let σ be a simplex spanned by $\{a_0, a_1, ..., a_n\}$. Then

$$\sigma = \left\{ t_0a_0 + ... + t_na_n \,\middle|\, t_i \geq 0, \sum_{i=0}^{n} t_i = 1 \right\}.$$

By part (1), $T(\sum t_ia_i) = \sum t_iT(a_i)$, and by part (6), $\{T(a_0), T(a_1), ..., T(a_n)\}$ is a geometrically independent set, so

$$T(\sigma) = \left\{ t_0T(a_0) + ... + t_nT(a_n) \,\middle|\, t_i \geq 0, \sum_{i=0}^{n} t_i = 1 \right\}$$

is a simplex and T preserves simplices. □

Theorem 2.2.12. *Let the set $S = \{a_0, a_1, a_2, ..., a_n\} \in \mathbb{R}^N$ be a geometrically independent set and let σ be the simplex spanned by S.*

(1) The barycentric coordinates $t_i(x)$ of x with respect to $\{a_0, a_1, a_2, ..., a_n\}$ are continuous functions of x.

(2) The n-simplex σ is the union of all line segments joining a_0 to points of the simplex s spanned by $\{a_1, a_2, ..., a_n\}$. Two such line segments intersect only at a_0.

(3) The n-simplex σ is a compact, convex set in $\mathbb{R}^N$ which equals the convex hull of $\{a_0, a_1, a_2, ..., a_n\}$.

(4) The interior of σ is convex and is open in the plane P; its closure is σ. $\mathrm{Int}(\sigma)$ is the union of all open line segments joining a_0 to the points of $\mathrm{Int}(s)$ where s is the face of σ opposite a_0.

Proof.

(1) Let T be the affine transformation carrying a_0 to 0 and a_i to ϵ_i. Then T carries $x = \sum_{i=0}^{n} t_i a_i$ to the point $(t_1, ..., t_n, 0, ..., 0)$. Then $t_i = x_i \circ T$. Since the barycentric coordinates t_i are the components of the continuous map T, they are continuous.

(2) First we show that the union of the line segments is in σ. Let $x = \left\{ \sum_{i=1}^{n} s_i a_i | s_i \geq 0, \sum_{i=1}^{n} s_i = 1 \right\}$ be a point on the simplex spanned by $\{a_1, ..., a_n\}$. Then the line segment $\left\{ t_0 a_0 + (1 - t_0) \sum_{i=1}^{n} s_i a_i | 0 \leq t_0 \leq 1 \right\}$ is in σ because $t_0 s_i \geq 0$ and $t_0 + (1 - t_0)(s_1 + ... + s_n) = 1$.

Now we show that σ is in the union of the line segments. Given a point $\sum_{i=0}^{n} t_i a_i$ in σ with $t_0 \neq 1$, set $s_i = \frac{t_i}{1 - t_0}$ for $i = 1, ..., n$. This shows that every point in the simplex spanned by $a_0, ..., a_n$, except a_0, is in the union of the line segments.

Clearly a_0 is also in the union, since it is in each line segment. Therefore σ is the union of all line segments from a_0 to the simplex spanned by $a_1, ..., a_n$.

Furthermore, we show that two such line segments intersect only at a_0. Clearly two line segments from a_0 to the simplex spanned by $a_1, ..., a_n$ do intersect at a_0. Now suppose they intersect at some other point y. Then the two segments must lie on the same line L. The other endpoints of the line segments lie on the simplex spanned by $a_1, ..., a_n$. But then L must be contained in the simplex spanned by $a_1, ..., a_n$. This contradicts the fact that a_0 does not lie on the simplex spanned by $a_1, ..., a_n$, since $\{a_0, a_1, ..., a_n\}$ are geometrically independent.

(3) We demonstrate compactness by showing that σ is closed and bounded. The simplex σ is bounded since

$$\left\| \sum_{i=0}^{n} t_i a_i \right\| \leq \max\{t_0, ..., t_n\} \max\{\|a_o\|, ..., \|a_n\|\} \leq \max\{\|a_0\|, ..., \|a_n\|\}.$$

The standard n-plane is closed since it is the inverse image of $\{1\}$ under the continuous map $(x_i) \to \sum_i t_i$ and the n-simplex is closed because it is the intersection of the n-plane and the closed sets $t_i \geq 0$.

For convexity, suppose that $(t_0, .., t_n)$ and $(s_0, ..., s_n)$ are two distinct n-tuples satisfying $t_i \geq 0$, $s_i \geq 0$, $\sum_i t_i = 1$ and $\sum_i s_i = 1$. Then $\lambda(t_0, ..., t_n) + (1-\lambda)(s_0, ..., s_n)$ parametrizes a line segment between these two points for $\lambda \in [0, 1]$. The corresponding curve

$$\lambda \sum_{i=0}^{n} t_i a_i + (1-\lambda) \sum_{i=0}^{n} s_i a_i = \sum_{i=0}^{n} (\lambda t_i + (1-\lambda) s_i) a_i$$

is a line segment connecting $\sum_{i=0}^{n} t_i a_i$ to $\sum_{i=0}^{n} s_i a_i$. Since t_i and s_i are non-negative and sum to 1, and since $\lambda \in [0, 1]$, we have that $\lambda t_i + (1-\lambda) s_i$ is also non-negative and sums to 1. So σ is convex.

To show that σ is the convex hull of $\{a_0, ..., a_n\}$, we apply Theorem 2.1.2 and show that σ is the intersection of all convex sets containing $\{a_0, ..., a_n\}$. Suppose S is a convex set containing $a_0, ..., a_n$. Then for every point $x = \sum_{i=0}^{n} t_i a_i \in \sigma$, we can show that $x \in S$ as follows. By convexity, since a_0 and a_1 are in S, we have that $b_1 = \frac{t_0}{t_0+t_1} a_0 + \frac{t_1}{t_0+t_1} a_1$ must also be in S. Therefore,

$$\begin{aligned} b_2 &= \frac{t_0 + t_1}{t_0 + t_1 + t_2} b_1 + \frac{t_2}{t_0 + t_1 + t_2} a_2 \\ &= \frac{t_0}{t_0 + t_1 + t_2} a_0 + \frac{t_1}{t_0 + t_1 + t_2} a_1 + \frac{t_2}{t_0 + t_1 + t_2} a_2 \end{aligned}$$

is also in S. Continuing in this manner, we find

$$b_n = \frac{t_0}{t_0 + ... + t_n} a_0 + ... + \frac{t_n}{t_0 + ... + t_n} a_n \in S,$$

but $\sum_{i=0}^{n} t_i = 1$, so $x \in S$. Then σ is contained in every convex set containing $\{a_0, ..., a_n\}$.

(4) The barycentric coordinates t_i are continuous functions from the plane P spanned by vertices of σ to $\mathbb{R}$. The preimage of the open interval $(0, \infty)$ under each t_i is therefore open in P. The interior of σ is the intersection of these finitely many open sets and therefore it is open.

The interior of σ is convex since the convex combination $(1-t)\,x + t\,y$ of any $x, y \in \mathrm{Int}\,(\sigma)$ is in $\mathrm{Int}\,(\sigma)$.

$$\begin{aligned} (1-t)\,x + t\,y &= (1-t) \sum_{i=0}^{n} t_i a_i + t \sum_{i=0}^{n} s_i a_i \\ &= \sum_{i=0}^{n} \left((1-t)\,t_i + t s_i\right) a_i \end{aligned}$$

and if t_i and s_i are positive, so is $(1-t)\,t_i + t s_i$.

Since σ is closed, the closure of $\mathrm{Int}\,(\sigma)$ is a subset of σ. On the other hand if a point $x \in \mathrm{Bd}\,(\sigma)$ is not in the closure of $\mathrm{Int}\,(\sigma)$, one can construct a sequence of points in $\mathrm{Int}\,(\sigma)$ which converges to x as follows.

Without loss of generality, assume that $x = \sum_{i=0}^{n} t_i a_i$, and $t_i(x) > 0$ for $0 \le i \le k$ and $t_{k+1}(x) = \ldots = t_n(x) = 0$.

Let $r = \min\{t_i(x) \mid 0 \le i \le k\}/2$ and let $x_m = \sum_{i=0}^{k} \left(t_i(x) - \frac{r}{m(k+1)}\right) a_i + \sum_{i=k+1}^{n} \frac{r}{m(n-k)}$. Then $x_m \in \mathrm{Int}(\sigma)$ and the sequence converges to x.

Finally, we show that $\mathrm{Int}(\sigma)$ equals the union of open line segments which join a vertex a_0 to points in the interior of the opposite face. Let x be an interior point of σ so that $x = \sum_{i=0}^{n} t_i a_i$ with $t_i > 0$ and $\sum t_i = 1$.

Rewrite this as $x = t_0 a_o + (1 - t_0) \sum_{i=1}^{n} \frac{t_i}{1-t_0} a_i$. Therefore x is on an open line segment joining a_0 to a point on the face opposite a_0.

□

Definition 2.2.13. If $w \in \mathbb{R}^N$, then a **ray** $\mathscr{R}$ emanating from w is the set of all points of the form $w + tp$ where p is a fixed point of $\mathbb{R}^N - \vec{0}$ and t ranges over the nonnegative real numbers.

Theorem 2.2.14. *Let U be a bounded, convex open set in $\mathbb{R}^N$. Let $w \in U$.*

(1) Each ray emanating from w intersects $\mathrm{Bd}(U) = \overline{U} - U$ in precisely one point.

(2) There is a homeomorphism from $\overline{U}$ to B^n carrying $\mathrm{Bd}(U)$ onto S^{n-1}.

Proof. To prove (1), given a ray $\mathscr{R}$ emanating from w, $\mathscr{R} \cap U$ is convex, bounded and open in $\mathscr{R}$. Hence $\mathscr{R} \cap U$ consists of all points of the form $w + tp$ where $t \in [0, a)$. Then $\mathscr{R}$ intersects $\overline{U} - U$ at the point $x = w + ap$. Suppose $\mathscr{R}$ intersects $\overline{U} - U$ at another point, say y. Then x lies between w and y on $\mathscr{R}$ (see Figure 2.2.4). So $x = (1 - t) w + ty$ where $0 < t < 1$, and $w = \frac{x - ty}{1-t}$. Choose a sequence y_n of points of U converging to y and define $w_n = \frac{x - ty_n}{1-t}$. Since $w_n \to w$ and U is open, $w_n \in U$ for some n. But $x = tw_n + (1 - t) y_n$ and the point $x \in U$ because U is convex; contradiction.

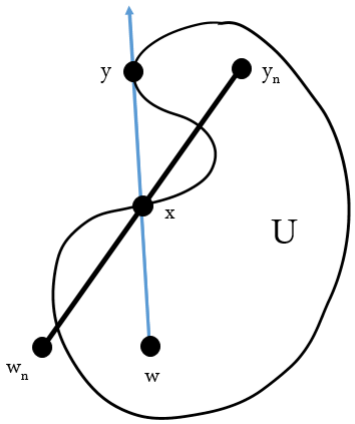


Figure 2.2.4: Each ray emanating from a point in a convex set intersects the boundary of the convex set in precisely one point.

To prove (2), assume $w = 0$. We may define a continuous map f of $\mathbb{R}^n - \vec{0}$ onto S^{n-1} by $f(x) = \frac{x}{\|x\|}$. By part 1, f restricts to a bijection of $\mathrm{Bd}(U)$ with S^{n-1}. Since $\mathrm{Bd}(U)$ is compact, this restriction is a homeomorphism; so let $g : S^{n-1} \to \mathrm{Bd}(U)$ be its inverse. Extend g to a bijection $G : B^n \to \overline{U}$ by letting G map the line segment joining $\vec{0}$ to the point u of S^{n-1} linearly onto the line segment joining $\vec{0}$ to $g(u)$, i.e. define

$$G(x) = \begin{cases} \left\| g\left(\frac{x}{\|x\|}\right) \right\| x & \text{if } x \neq 0 \\ \vec{0} & \text{if } x = 0 \end{cases}$$

G is continuous at $x = 0$ because if M is a bound for $\|g(x)\|$ then $\|x - 0\| < \delta$ implies $\|G(x) - G(0)\| < M\delta$. It is continuous elsewhere because G is differentiable at any $x \neq 0$. □

As a special case of Theorem 2.2.14, we have

Corollary 2.2.15. *Let σ be an n-simplex in $\mathbb{R}^N$. There is a homeomorphism of σ with the unit ball B^n that carries* $\mathrm{Bd}(\sigma)$ *onto the unit sphere S^{n-1}.* □

2.3 Simplicial complexes in $\mathbb{R}^N$

Definition 2.3.1. A **simplicial complex** K in $\mathbb{R}^N$ is a collection of simplices in $\mathbb{R}^N$ such that

(1) Every face of a simplex of K is in K; and

(2) The intersection of any two simplices of K is a face of each of the two simplices.

Example 2.3.2. Figures 2.3.1 and 2.3.2 provide examples of simplicial complexes. Figure 2.3.3 is an example of a collection of simplices which does not form a simplicial complex.

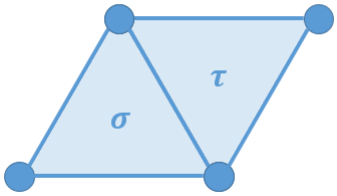


Figure 2.3.1: A simplicial complex.

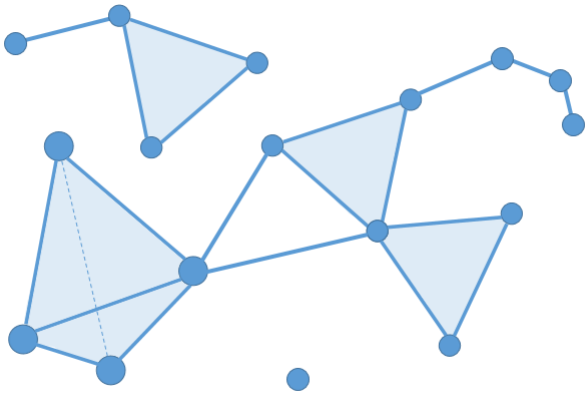


Figure 2.3.2: A simplicial complex.

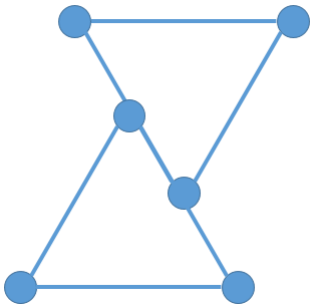


Figure 2.3.3: A collection of simplices which do not form a simplicial complex. The figure above is not a simplicial complex because the intersection of the two 1-simplices is not a face of either simplex.

Theorem 2.3.3. *Let K be a collection of simplices in $\mathbb{R}^N$ such that (1) every face of a simplex K is in K. Then the following are equivalent:*

(2) The intersection of two simplices of K is a face of each of the two simplices.

(2') Every pair of distinct simplices of K has disjoint interiors.

Proof. Assume K is a simplicial complex. Given two simplices σ and τ of K, suppose their interiors have a point x in common. Let $s = \sigma \cap \tau$. If s were a proper face of σ, then x would belong to $\mathrm{Bd}(\sigma)$, which it does not. Therefore, $s = \sigma$. Similarly $s = \tau$.

Now assume (1) and (2′) are true. We show that if σ and τ are simplices in K and $\sigma \cap \tau$ is nonempty, then $\sigma \cap \tau$ is the face σ' of σ spanned by those vertices $b_0, b_1, ..., b_m$ of σ that lie in τ. First, σ' is contained in $\sigma \cap \tau$ because $\sigma \cap \tau$ is convex and contains $b_0, b_1, ..., b_m$. Suppose $x \in \sigma \cap \tau$. Then $x \in \mathrm{Int}(s) \cap \mathrm{Int}(t)$ for some face s of σ and t of τ. Since we assume that distinct simplices of K have disjoint interiors, then $s = t$, and so the vertices of s lie in τ, so by definition, the vertices of s are elements of $b_0, b_1, ..., b_m$. Then s is a face of σ', so $x \in \sigma'$. □

Definition 2.3.4. If L is a subcollection of K that contains all faces of its elements, then L is a simplicial complex. It is called a **subcomplex** of K. One subcomplex of K is the collection of all simplices of K of dimension at most p, called the **p-skeleton** of K and is denoted $K^{(p)}$. The points of the collection $K^{(0)}$ are called **vertices** of K.

Definition 2.3.5. Let K be a simplicial complex. Let $|K|$ be the subset of $\mathbb{R}^{\mathbb{N}}$ that is the union of the simplices of K. We give each simplex σ its natural topology as a subspace of $\mathbb{R}^N$. We then topologize $|K|$ by declaring that a subset A of $|K|$ is closed in $|K|$ if and only if $A \cap \sigma$ is closed in σ for all $\sigma \in K$. This defines a topology on $|K|$ called the **CW-topology**. This is indeed a topology because the collection of all such subsets A is closed under finite unions and arbitrary intersections. The space $|K|$ with the subspace topology inherited from $\mathbb{R}^N$ is called the **underlying space of** K or the **polytope** of K. A

space that is the polytope of a simplicial complex is a **polyhedron**.

Theorem 2.3.6. *Let K be a simplicial complex and let $|K|$ be the underlying space of K. Every set which is closed in the topology on $|K|$ as a subspace of $\mathbb{R}^N$ is also closed in the CW-topology on $|K|$.*

Proof. Let K be a simplicial complex and let $|K|$ be the underlying space of K. Suppose A is a subset of $|K|$ which is closed in the subspace topology. Then $A = B \cap |K|$ for some B closed in $\mathbb{R}^N$. Then for any simplex $\sigma \in K$, we have to show that $A \cap \sigma$ is closed in σ, i.e. that $B \cap |K| \cap \sigma$ is closed in σ, i.e. that $B \cap \sigma$ is closed in σ. □

Example 2.3.7 (Counterexample to the Converse of Theorem 2.3.6)**.** Let D be a disc in $\mathbb{R}^2$. It can be thought of as the set union of all its radii, which are nothing but 1-simplices. Denote one of the radii as $\hat{\sigma}$. Consider the subset $\hat{D} = \{D \setminus \hat{\sigma}\} \cup \{0\}$. For any radius $\sigma \neq \hat{\sigma}$, $\hat{D} \cap \sigma = \sigma$. Also, $\hat{D} \cap \hat{\sigma}$ gives the centre of the disc, which is a 0-simplex that is closed in $\hat{\sigma}$. So $\hat{D}$ is closed in the CW complex. However, note that $\hat{D}$ is not closed in the subspace topology that D inherits from $\mathbb{R}^2$. This is illustrated in Figure 2.3.4.

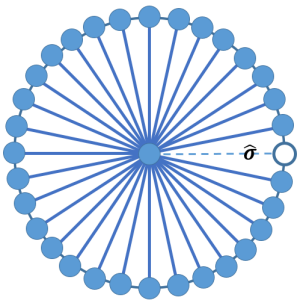
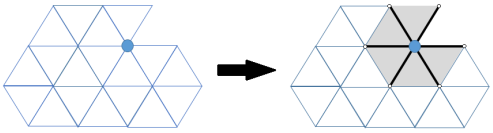


Figure 2.3.4: An example of a set which is closed in the CW-topology, but not in the subspace topology.

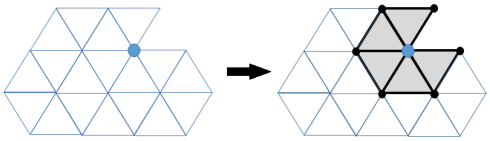
Definition 2.3.8. Let K be a simplicial complex. If v is a vertex of K, the **star** of v in K is the union of the interiors of those simplices of K that have v as a vertex. The star of v in

K is denoted $\mathrm{St}(v)$ or $\mathrm{St}(v,K)$. Its CW-closure, $\overline{\mathrm{St}}(v)$ is called the **closed star** of v in K. It is the union of all simplices of K having v as a vertex. $\overline{\mathrm{St}}(v) - \mathrm{St}(v)$ is the **link** of v in K, denoted $\mathrm{link}(v)$.

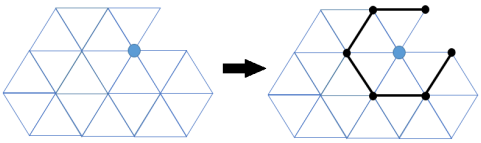
Example 2.3.9. Figure 2.3.5a illustrates the star of a vertex. The star of a vertex, v, is the interiors of all the simplices containing v. Note that the star of a vertex v is not a simplicial complex unless v is isolated (not in any simplices other than itself). The closed star, however, is a simplicial complex, as pictured in Figure 2.3.5b. The closed star includes all simplices touching the vertex. The link of a vertex is the boundary of the closure of the star as shown in Figure 2.3.5c. The link of a vertex is a simplicial complex.



(a) A vertex and its star.



(b) A vertex and its closed star.



(c) A vertex and its link.

Figure 2.3.5: Stars and links of vertices.

2.4 Simplicial maps

Theorem 2.4.1. *Let K be a simplicial complex. Every simplex $\sigma \in K$ is a subspace of K under the CW-topology.*

Proof. Let K be a simplicial complex and let σ be a simplex in K endowed with the CW-topology $\mathcal{T}$. For any closed set $B \in (K, \mathcal{T})$, $B \cap \sigma$ is closed in σ because K is endowed with the CW-topology.

Conversely, suppose we have a set $V \subset \sigma$ closed in the intrinsic topology of σ. We need to show $V = \sigma \cap B$ for some B closed in K. Consider $B = V$. Then, note that $V \cap \tau$ is closed in τ for each simplex τ, because $V \cap \tau \subset (V \cap \sigma) \cap \tau$, a subspace of σ.

Therefore every simplex $\sigma \in K$ is a subspace of K under the CW-topology. □

From this point on, for the simplicial complex K, assume $|K|$ has the CW-topology.

Theorem 2.4.2. *Let K be a simplicial complex and let X be a topological space. A map $f : |K| \to X$ is continuous on the polyhedron $|K|$ of K if and only if the restriction of f to each simplex σ of K is continuous on that simplex.*

Proof. Assume $f : |K| \to X$ is continuous on $|K|$. Then, from Theorem 2.4.1, we know that any simplex σ of K is closed under the CW-topology. We define the restriction of f to σ as $f|_\sigma : \sigma \to X$. For any closed set $B \subset X$, we have that $\left(f|_\sigma\right)^{-1}(B) = \sigma \cap f^{-1}(B)$ (as a property of inverse image). However, since f is continuous, $f^{-1}(B)$ is closed in $|K|$. Therefore, $\left(f|_\sigma\right)^{-1}(B)$ is closed in σ and $f|_\sigma$ is continuous.

Now assume that $f|_\sigma : \sigma \to X$ is continuous on σ for all σ. If B is a closed set of X, then $f^{-1}(B) \cap \sigma = \left(f|_\sigma\right)^{-1}(B)$, which is closed in σ by continuity of $f|_\sigma$. Thus $f^{-1}(B)$ is closed in $|K|$ by definition of the CW-topology, and f is continuous. □

Theorem 2.4.3. *Let K and L be complexes and let $f : K^{(0)} \to L^{(0)}$ be a map. Suppose whenever vertices $v_0, v_1, ..., v_n$ of K span a simplex of K, the points $f(v_0), f(v_1), ..., f(v_n)$*

are vertices of a simplex of L. Then f can be extended to a continuous map $g : |K| \to |L|$ such that $x = \sum_{i=0}^{n} t_i v_i$ implies that $g(x) = \sum_{i=0}^{n} t_i f(v_i)$. The map g is called the ***linear simplicial map*** *induced by the vertex map f.*

Proof. The map g maps the n-simplex σ spanned by $v_0, v_1, ..., v_n$ continuously to the simplex τ whose vertex set is $f(v_0), f(v_1), ..., f(v_n)$. We also see that g is continuous as a map of σ into τ and hence as a map of σ into $|L|$. So by Theorem 2.4.2, g is continuous as a map from $|K|$ into $|L|$. □

Note that the composition of two simplicial maps, $h \circ g$, is a simplicial map as well.

Definition 2.4.4. Suppose $f : K^{(0)} \to L^{(0)}$ is a bijective correspondence such that $v_0, v_1, ..., v_n$ of K span a simplex of K if and only if $f(v_0), f(v_1), ..., f(v_n)$ span a simplex of L. The map $g : |K| \to |L|$ is a **simplicial homeomorphism** or an **isomorphism** of K with L.

Theorem 2.4.5. *Suppose $f : K^{(0)} \to L^{(0)}$ is a bijection such that $v_0, v_1, ..., v_n$ of K span a simplex of K if and only if $f(v_0), f(v_1), ..., f(v_n)$ span a simplex of L. Then the induced simplicial map $g : |K| \to |L|$ is a homeomorphism.*

Proof. (1) Since f is bijective and $f : K^{(0)} \to L^{(0)}$ is a bijection such that $v_0, v_1, ..., v_n$ of K span a simplex of K if and only if $f(v_0), f(v_1), ..., f(v_n)$ span a simplex of L, f^{-1} induces a linear simplicial map h.

(2) Note that if $x = \sum t_i v_i$ then $g(x) = \sum t_i f(v_i)$ where

$$h\big(g(x)\big) = h\Big(\sum t_i f(v_i)\Big) = \sum t_i f^{-1}\big(f(v_i)\big) = \sum t_i v_i = x.$$

So h and g are inverses on any simplex of K and therefore on all of $|K|$.

□

Corollary 2.4.6. *Let Δ^N denote the complex of an N-simplex and its faces. If K is a finite complex then K is isomorphic to a subcomplex of Δ^N for some N.*

Proof. Let $v_0, v_1, ..., v_N$ be vertices of K. Then choose $a_0, a_1, ..., a_N$ to be geometrically independent points in $\mathbb{R}^{\mathbb{N}}$ and let Δ^N consist of the N-simplex they span along with its faces. The vertex map $f(v_i) = a_i$ induces an isomorphism of K with a subcomplex of Δ^N, as described in Theorem 2.4.5. □

Definition 2.4.7. The **dimension** of K, denoted $\dim(K)$, is the largest dimension of a simplex of K. If $K \in \mathbb{R}^{\mathbb{N}}$ then $\dim(K) \leq N$.

Definition 2.4.8. A **simplicial n-complex** is a simplicial complex where the largest dimension of any simplex in the complex is n.

2.5 Abstract simplicial complexes

Definition 2.5.1. A **partial order,** $\preceq$ is a binary relation which is reflexive ($x \preceq x$), anti-symmetric (if $x \preceq y$ then $y \not\preceq x$) and transitive (if $x \preceq y$ and $y \preceq z$ then $x \preceq z$). A **poset** is a set with a partial order.

Definition 2.5.2. The **upset** of a partially ordered set $(X, \preceq)$ is a subset U with the property that if $x \in U$ and $x \preceq y$ then $y \in U$. Similarly, a **downset** is a subset L with the property that if $x \in L$ and $y \preceq x$ then $y \in L$.

Example 2.5.3. Figure 2.5.1 illustrates the **Boolean lattice** (collection of all subsets, ordered by inclusion) of the set $\{x, y, z\}$ with the upset $\uparrow x$ highlighted, where $\uparrow x = \{y \in X \,|\, y \succeq x\}$.

Definition 2.5.4. An **abstract simplicial complex** is a collection $\mathscr{S}$ of finite non-empty sets such that if A is an element of $\mathscr{S}$, so is every non-empty subset of A. So $\mathscr{S}$ is a

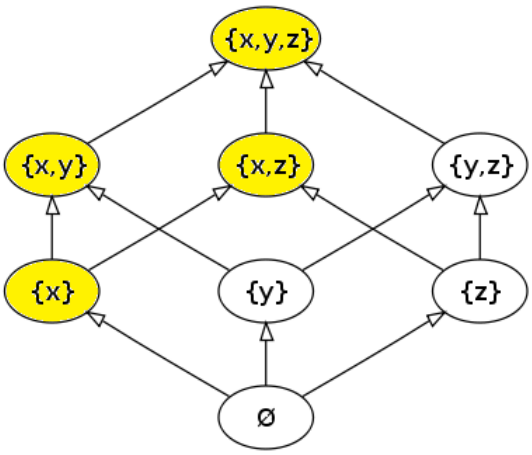


Figure 2.5.1: An upset of the powerset of $\{x, y, z\}$.

downset in some Boolean lattice. If $\mathscr{S}$ is a abstract simplicial complex, then $A \in \mathscr{S}$ is called a **simplex** of $\mathscr{S}$ and its **dimension** is one less than the number of its elements. Each non-empty subset of A is called a **face** of A. The **dimension** of $\mathscr{S}$ is the largest dimension of one of its simplices (or ∞). The **vertex set** V of $\mathscr{S}$ is the union of one-point elements of $\mathscr{S}$. A subcollection of $\mathscr{S}$ that is itself a complex is called a **subcomplex** of $\mathscr{S}$.

In other words, to define a simplicial complex S we:

(1) Choose the vertex set V of S (these are the 0-cells of S),

(2) For $1 \leq k \leq n$, specify which k-subsets of V are in S,

(3) Check that S is actually a downset.

Definition 2.5.5. The abstract simplicial complexes $\mathscr{S}$ and $\mathscr{T}$ are **isomorphic** if there is a bijection f mapping vertices of $\mathscr{S}$ to vertices of $\mathscr{T}$ such that

$$\{a_0, a_1, ..., a_n\} \in \mathscr{S}$$

if and only if

$$\{f(a_0), f(a_1), ..., f(a_n)\} \in \mathscr{T}.$$

Definition 2.5.6. Let K be a simplicial complex in $\mathbb{R}^N$. If $\mathscr{K}$ is the collection of all subsets $\{a_0, a_1, ..., a_n\}$ of the vertex set V, such that $a_0, a_1, ..., a_n$ span a simplex of K then $\mathscr{K}$ is the **vertex scheme** of K.

Theorem 2.5.7.

(1) Every finite abstract complex $\mathscr{S}$ is isomorphic to the vertex scheme of some simplicial complex $K \in \mathbb{R}^N$.

(2) *Two simplicial complexes in* $\mathbb{R}^N$ *are linearly isomorphic if and only if their vertex schemes are isomorphic.*

Proof. The proof of part (2) follows directly from Theorem 2.4.5.

We proceed with the proof of part (1). Given an index set J, let Δ^J be the collection of all simplices in $\mathbb{E}^J$ spanned by finite subsets of the standard basis ϵ_α for $\mathbb{E}^J$. It is clear that Δ^J is a simplicial complex; if σ and τ are two simplices of Δ^J then their combined vertex set is geometrically independent and spans a simplex of Δ^J.

Let S be an abstract complex with finite vertex set V. Choose an index set J large enough so there is an injective function $f : V \to \{\epsilon_\alpha \mid \alpha \in J\}$. We specify a subcomplex K of Δ^J by the condition that for each abstract simplex $a_0, ..., a_n \in S$, the geometric simplex spanned by $f(a_0), ..., f(a_n)$ is to be in K. Then it follows that K is a simplicial complex and S is isomorphic to the vertex scheme of K. The function f is the required correspondence between vertex sets. □

Definition 2.5.8. If $\mathscr{S}$ is isomorphic with the vertex scheme of the simplicial complex K, we call K a **geometric realization** of $\mathscr{S}$. By Theorem 2.4.6, K is well-defined up to homeomorphism.

Example 2.5.9. Figure 2.5.2 demonstrates the geometric realization of a simplicial complex, resulting in a cylinder.

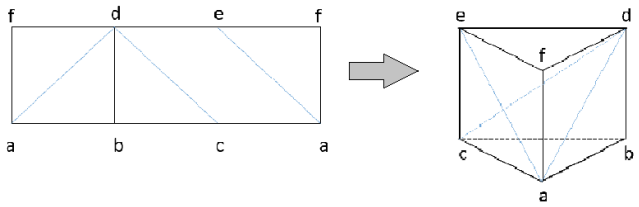


Figure 2.5.2: The geometric realization of a simplicial complex. The simplicial complex is on the left with its geometric realization as a cylinder on the right.

Definition 2.5.10. Given a finite abstract simplicial complex, L, a **labelling** of the vertices of L is a bijective function f mapping the set of L to a set of labels $\mathscr{L}$. Let $\mathscr{S}$ be the simplicial complex with vertex set $\mathscr{L}$ and $\{b_0, b_1, ..., b_k\}$ a simplex of $\mathscr{S}$ if and only if

$$\{b_0, b_1, ..., b_k\} = \{f(a_0), f(a_1), ..., f(a_n)\}$$

for some simplex $\{a_0, a_1, ..., a_n\}$ in L. If K is the geometric realization of $\mathscr{S}$, then K is the **complex derived from the labelled complex** L and $g : |L| \to |K|$ is the **associated pasting map**. The map g is a closed quotient map.

Definition 2.5.11. If L is a complex, a subcomplex L_0 of L is said to be a **full subcomplex** of L if each simplex of L whose vertices belong to L_0 belongs to L_0 itself.

Example 2.5.12. Figure 2.5.3 depicts a full subcomplex of the exterior of a tetrahedron. A triangle with interior is a full subcomplex, but a triangle without interior is not a full subcomplex of a tetrahedron.

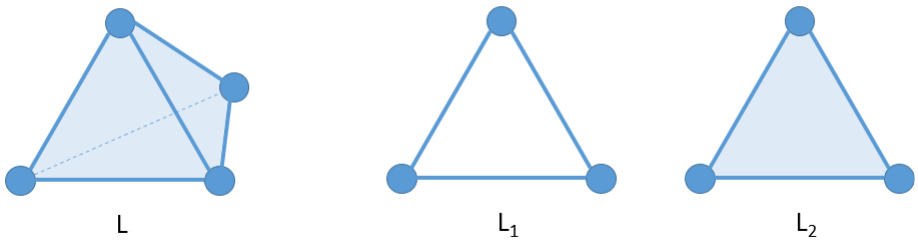


Figure 2.5.3: A full subcomplex. Suppose L is the exterior of a tetrahedron. Then L_1, the exterior of a triangle is not a full subcomplex of L, but L_2, the triangle including its interior, is a full subcomplex of L.

Theorem 2.5.13. *Let L be a complex. Let f be a labelling of L. Let $g : |L| \to |K|$ be the pasting map induced by f. Let L_0 be a full subcomplex of L. Suppose that whenever v and w are vertices of L with the same label (i.e. $f(v) = f(w)$), we have that*

- *v and w belong to L_0, and*
- *$\overline{\mathrm{St}}(v)$ and $\overline{\mathrm{St}}(w)$ are disjoint.*

Then $\dim\big(g(\sigma)\big) = \dim(\sigma)$ *for all simplices $\sigma \in L$. If $g(\sigma_1) = g(\sigma_2)$, then σ_1 and σ_2 must be disjoint simplices belonging to L_0.*

Proof. Suppose that $\dim\big(g(\sigma)\big) \neq \dim(\sigma)$ for some simplex $\sigma \in L$. Then, because g is a function sending vertices to vertices, $\dim\big(g(\sigma)\big) \not> \dim(\sigma)$.

So we suppose $\dim(\sigma) > \dim\big(g(\sigma)\big)$. So there exists a vertex $w \in g(\sigma)$ such that there is more than one vertex $v \in L$ such that $g(v) = w$. Say v_1 and v_2 are two such vertices. But $v_1 v_2 \in \overline{\mathrm{St}}(v_1)$ and $v_1 v_2 \in \overline{\mathrm{St}}(v_2)$. Contradiction.

Suppose $g(\sigma_1) = g(\sigma_2)$ but σ_1 and σ_2 are not disjoint, i.e. they share v as a common vertex. Let u_1 be another vertex in σ_1 and let u_2 be another vertex in σ_2 such that $g(u_1) = g(u_2)$. But $v \in \overline{\mathrm{St}}(u_1)$ and $v \in \overline{\mathrm{St}}(u_2)$. Contradiction. □

CHAPTER 3

HOMOLOGY OF SIMPLICIAL COMPLEXES

3.1 Oriented simplices

Definition 3.1.1. Let σ be a simplex. Define two orderings of its vertex set to be equivalent if they differ by even permutation. If $\dim(\sigma) > 0$ the orderings fall into two equivalence classes, and each class is called an **orientation** of σ. If $\dim(\sigma) = 0$, then there is only one equivalence class. An **oriented simplex** is a simplex σ with an orientation.

Definition 3.1.2. If $v_0, v_1, ..., v_p$ are geometrically independent, then let $v_0 v_1 v_2 ... v_p$ be the **simplex** spanned by $v_0, v_1, ..., v_p$ and let $[v_0, v_1, ..., v_p]$ be called the **ordered simplex** and equivalence class of the ordering $(v_0, v_1, ..., v_p)$.

Example 3.1.3. Figure 3.1.1 illustrates examples of oriented simplices of various dimension.

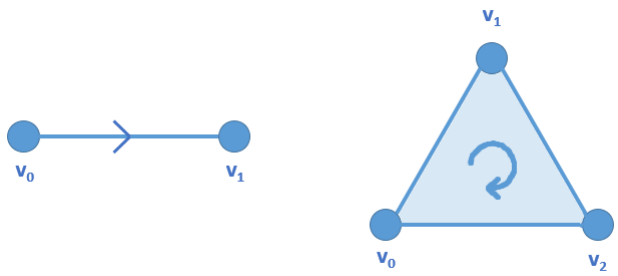


Figure 3.1.1: An oriented 1-simplex and 2-simplex.

Definition 3.1.4. Let K be a simplicial complex. A p**-chain** on K is a function c from the set of oriented p-simplices to $\mathbb{Z}$ such that

- $c(\sigma) = -c(\sigma')$ if σ and σ' are the same simplex, with opposite orientation.

- $c(\sigma) = 0$ for all but finitely many oriented p-simplices σ.

Definition 3.1.5. The group resulting from adding the values of p-chains is called the **group of (oriented) p-chains** of K, denoted $C_p(K)$.

Definition 3.1.6. If σ is an oriented simplex, the **elementary chain**, c, corresponding to σ is defined by

$$c(\tau) = \begin{cases} 1 & \text{if } \tau = \sigma, \\ -1 & \text{if } \tau = \sigma', \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 3.1.7. *The group of oriented p-chains of K, $C_p(K)$, is a free abelian group. A basis for $C_p(K)$ can be found by orienting each p-simplex and using their corresponding elementary chains as a basis.*

Proof. Each p-chain can be written as a linear combination of elementary chains σ_i via $c = \sum n_i \sigma_i$. The chain c assigns the value

$$c = \begin{cases} n_i & \text{to the oriented } p\text{-simplex } \sigma_i, \\ -n_i & \text{to the opposite orientation of } \sigma_i, \\ 0 & \text{to all oriented } p\text{-simplices not included in the summation.} \end{cases}$$

$C_0(K)$ has a natural basis (since a 0-simplex has only one orientation) but in general $C_p(K)$ has no natural basis if $p > 0$. One must start by orienting the p-simplices of K to obtain a basis. □

Corollary 3.1.8. *Any function f from the oriented p-simplices of K to an abelian group G extends uniquely to a homomorphism $C_p(K) \to G$ provided*

$$f(-\sigma) = -f(\sigma) \tag{3.1}$$

for all p-simplices σ.

Proof. Given (3.1), pick an orientation of each of the p-simplices to obtain a basis for $C_p(K)$. Then we apply the universal property of free abelian groups: for every arbitrary function f from B to some abelian group A, there exists a unique group homomorphism from F to A which extends f. (See Hungerford Theorem 1.1 and the remarks following it [Hun80, pages 72–73]). □

Definition 3.1.9. The **boundary operator** $\partial_p : C_p(K) \to C_{p-1}(K)$ is defined

$$\partial_p \sigma = \partial_p [v_0, v_1, ..., v_p] = \sum_{i=0}^{p} (-1)^i [v_0, ..., \hat{v}_i, ..., v_p], \tag{3.2}$$

where $\hat{v}_i$ means the vertex v_i is to be deleted from the list. Since $C_p(K)$ is the trivial group for $p < 0$, ∂_p is the trivial homomorphism for $p \leq 0$.

Example 3.1.10. The boundary of a 1-simplex $[v_0, v_1]$ is

$$\partial_1 [v_0, v_1] = v_1 - v_0.$$

Example 3.1.11. The boundary of the 2-simplex $[v_0, v_1, v_2]$ in Figure 3.1.2 is

$$\partial_1 [v_0, v_1, v_2] = [v_1, v_2] - [v_0, v_2] + [v_0, v_1].$$

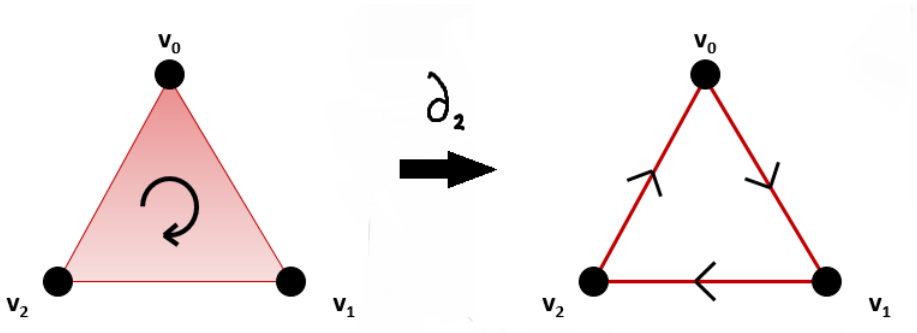


Figure 3.1.2: An oriented 2-simplex and its boundary.

Example 3.1.12. The boundary of the boundary of a 2-simplex is 0, since

$$\begin{aligned}\partial_1\partial_2[v_0,v_1,v_2] &= \partial_1([v_1,v_2]-[v_0,v_2]+[v_0,v_1])\\ &= v_2-v_1-(v_2-v_0)+v_1-v_0\\ &= 0.\end{aligned}$$

Theorem 3.1.13. *The composition of two boundary maps is* 0*, i.e.,* $\partial_{p-1}\partial_p = 0$.

Proof. Let ∂_{p-1} and $\partial_p = 0$ be boundary maps. Then

$$\partial_{p-1}\partial_p[v_0,v_1,...,v_p] = \sum(-1)^i\partial_{p-1}[v_0,...,v_i,...,v_p]$$

$$= \sum(-1)^i(-1)^j[...,v_j,...,v_i,...] + \sum(-1)^i(-1)^{j-1}[...,v_i,...,v_j,...], \tag{3.3}$$

and the terms in the two sums in (3.3) cancel in pairs. □

Definition 3.1.14. The kernel of $\partial_p : C_p(K) \to C_{p-1}(K)$ is called the group of p-**cycles** and is denoted $Z_p(K)$. The image of $\partial_{p+1} : C_{p+1}(K) \to C_p(K)$ is called the group of p-**boundaries** and is denoted $\partial_{p+1}(K)$. The p^{th} **homology group** of K is defined

$$H_p(K) = Z_p(K)/\partial_{p+1}(K), \tag{3.4}$$

or, equivalently,

$$H_p(K) = \ker(\partial_p)/\operatorname{im}(\partial_{p+1}). \tag{3.5}$$

Definition 3.1.15. We say that a chain c is **carried by** a subcomplex L of K if c has value 0 on every simplex that is not in L. Two p-chains c and c' are called **homologous** if $c - c' = \partial_{p+1}d$ for some $p+1$ chain d. If $c = \partial_{p+1}d$, we say c is **zero-homologous**.

Example 3.1.16. The two chains, c_1 and c_2, in Figure 3.1.3 and 3.1.4 are homologous since $c_2 - c_1 = \partial(S)$. (One must imagine that the shaded areas are filled with triangles).

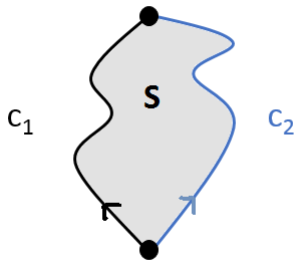


Figure 3.1.3: Homologous p-chains.

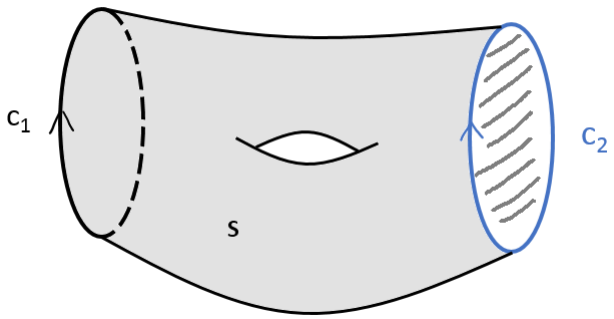


Figure 3.1.4: Homologous p-chains.

3.2 Examples of homology computations

Example 3.2.1. Let L be the complex in Figure 3.2.1, whose underlying space is a rectangle. Let $\mathrm{Bd}\,(L)$ denote the complex whose space is the boundary of the rectangle. Orient each 2-simplex σ_i of L by a counterclockwise arrow. Orient the 1-simplices arbitrarily. Then

(1) Every 1-cycle of L is homologous to a 1-cycle carried by $\mathrm{Bd}\,(L)$,

(2) If d is a 2-chain of L and if ∂d is carried by $\mathrm{Bd}\,(L)$, then d is a multiple of the chain $\sum \sigma_i$, where the sum is over all the simplices.

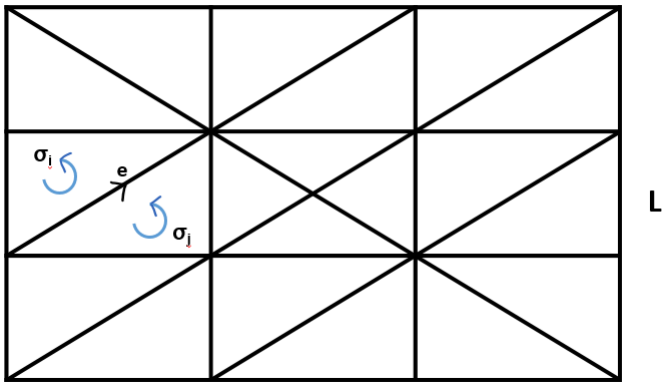


Figure 3.2.1: A complex whose underlying space is a rectangle, illustrating Example 3.2.1.

Proof. If σ_i and σ_j have an edge in common, then ∂d must have a value of 0 on e, as illustrated in Figure 3.2.1. It follows that d must have the same value on σ_i as it does on σ_j. Continuing, we see that d has the same value on every oriented 2-simplex σ_i, proving (2). For (1), start with a 1-chain c of L and push it off the 1-simplices, one at a time. First, one shows that c is homologous to a 1-chain c_1 carried by the subcomplex pictured in Figure 3.2.2. Then one shows that c_1 is homologous to a 1-chain c_2 carried

by the subcomplex pictured in Figure 3.2.1. Finally, one notes that in the case where the original chain c is a cycle, then the chain c_2 is also a cycle. So c_2 must be carried by $\mathrm{Bd}(L)$, otherwise c_2 would have a non-zero coefficient on one or more of $v_1, v_2, ..., v_5$. □

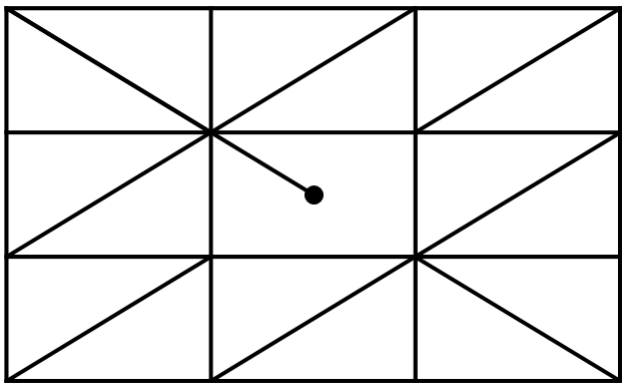


Figure 3.2.2: Illustration of the proof of Example 3.2.1.

Example 3.2.2. Let T be the complex represented by the labeled rectangle in Figure 3.2.3, whose underlying space is a torus. Then $H_1(T) \cong \mathbb{Z} \bigoplus \mathbb{Z}$ and $H_2(T) \cong \mathbb{Z}$. Orient each 2-simplex of L counterclockwise; use the induced orientation of the 2-simplices of T; let γ denote their sum. Let $w_1 = [a,b] + [b,c] + [c,a]$ and $z_1 = [a,d] + [d,e] + [e,a]$. Then γ generates $H_2(T)$ and w_1 and z_1 represent a basis for $H_1(T)$.

Proof. Let $g : |L| \to |T|$ be the pasting map. Let $A = g(|\mathrm{Bd}(L)|)$. Then A is homeomorphic to a space that is the wedge of two circles. Orient the 1-simplices of T arbitrarily. Since g makes identifications only among simplices of $\mathrm{Bd}(L)$,

(1) Every 1-cycle of T is homologous to a 1-cycle carried by A,

(2) If d is a 2-chain of T and if ∂d is carried by A, then d is a multiple of γ (from 3.2.1),

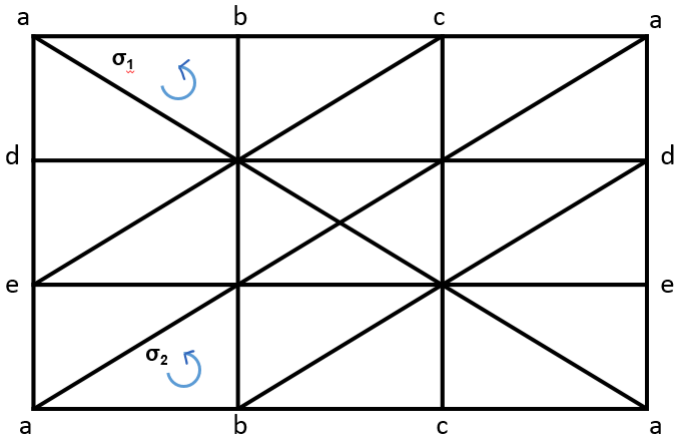


Figure 3.2.3: A complex whose underlying space is a torus.

(3) If c is a 1-cycle of T carried by A, then c is of the form $nw_1 + mz_1$ (since A is just the 1-dimensional complex pictured in Figure 3.2.4),

(4) $\partial\gamma = 0$.

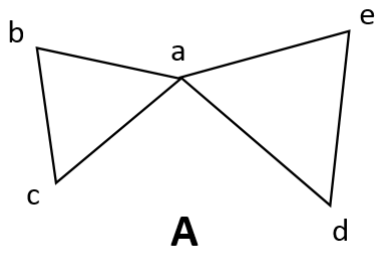


Figure 3.2.4: A is a 1-dimensional complex.

It is clear that $\partial\gamma = 0$ on every 1-simplex of T not in A. One can also directly check that it has value 0 on each 1-simplex of A. For example, $[a,b]$ appears in the expression for $\partial\sigma_1$ with value -1 and in $\partial\sigma_2$ with value $+1$ so $\partial\gamma$ has value 0 on $[a,b]$. Now we can compute the homology of T. Every 1-cycle of T is homologous to a 1-cycle of the form $c = nw_1 + mz_1$ by (1) and (3). Such a cycle bounds only if it is trivial. If $c = \partial d$ for some d then (2) applies to show $d = \partial\gamma$ for some p since $\partial\gamma = 0$ by (4), we have $c = \partial d = 0$. We conclude $H_1(T) \cong \mathbb{Z} \bigoplus \mathbb{Z}$ and the 1-cycles w_1 and z_1 form a basis for the 1-dimensional homology. To compute $H_2(T)$, note that any 2-cycle d of T must be of the form $p\gamma$ for some p by (2). Each such 2-chain is a cycle, by (4), and there are no 3-chains for it to bound. So $H_2(T) \cong \mathbb{Z}$ and this group has the 2-cycle γ as a generator. □

3.3 Singular homology

Definition 3.3.1. A **singular n-simplex** in a space X is a continuous map $\sigma : \Delta^n \to X$.

The fact that the n-simplex is singular expresses that idea that σ may not be a nice embedding, but may have singularities where its image does not look at all like a

simplex. The map σ need not be injective, so there can be non-equivalent singular n-simplices with the same image in X. Many of the definitions for singular homology are analogous to those for simplicial homology, as we will see below.

Definition 3.3.2. A **singular n-chain** is a finite sum $\sum n_i\sigma_i$ for $n_i \in \mathbb{Z}$ and $\sigma_i : \Delta^n \to X$. The n-chains are elements of the free abelian group $C_n(X)$ whose basis is the set of singular n-simplices in X.

Definition 3.3.3. A **boundary homomorphism** is a map $\partial_n : C_n(X) \to C_{n-1}(X)$ whose basis elements are

$$\partial_n(\sigma) = \sum_i (-1)^i \sigma|[v_0, ..., \hat{v}_i, ..., v_n]. \tag{3.6}$$

Theorem 3.3.4. *The composition $\partial_{n-1}\partial_n$ is zero.*

Proof. Let $\partial_n : C_n(X) \to C_{n-1}(X)$ be a boundary homomorphism. Then

$$\begin{aligned}
\partial_{n-1}\partial_n(\sigma) &= \sum_{j<i} (-1)^i (-1)^j \sigma\left[v_0, ..., \hat{v}_j, ..., \hat{v}_i, ...v_n\right] \\
&\qquad + \sum_{i<j} (-1)^i (-1)^{j-1} \sigma\left[v_0, ..., \hat{v}_i, ..., \hat{v}_j, ..., v_n\right] \\
&= \sum_{j<i} (-1)^i (-1)^j \sigma\left[v_0, ..., \hat{v}_j, ..., \hat{v}_i, ...v_n\right] \\
&\qquad + \sum_{j<i} (-1)^j (-1)^{i-1} \sigma\left[v_0, ..., \hat{v}_j, ..., \hat{v}_i, ..., v_n\right] \\
&= \sum_{j<i} (-1)^i (-1)^j \sigma\left[v_0, ..., \hat{v}_j, ..., \hat{v}_i, ...v_n\right] \\
&\qquad - \sum_{j<i} (-1)^j (-1)^i \sigma\left[v_0, ..., \hat{v}_j, ..., \hat{v}_i, ..., v_n\right] = 0.
\end{aligned}$$

□

Since the composition of two singular boundary homomorphism is zero, just as in the case for simplicial homology, we can define the singular homology groups in the same way as we did for simplicial homology.

Definition 3.3.5. The n^{th} **singular homology group** is the quotient group

$$H_n(X) = \ker \partial_n / \operatorname{im} \partial_{n+1}, \tag{3.7}$$

or, equivalently,

$$H_n(X) = Z_n(X) / \partial_{n+1}(X), \tag{3.8}$$

where $Z_n(X)$ is the group of singular n-cycles and $\partial_{n+1}(X)$ is the group of singular n-boundaries.

Theorem 3.3.6. *The simplicial and singular homology groups of simplicial Δ-complexes are always isomorphic.*

The proof of Theorem 3.3.6 is discussed in Hatcher [Hat02, pages 127–129].

3.4 Example: simplicial chain complexes for a Möbius strip

Let X be the complex represented by the labeled rectangle in Figure 3.4.1, whose underlying space is a Möbius strip.

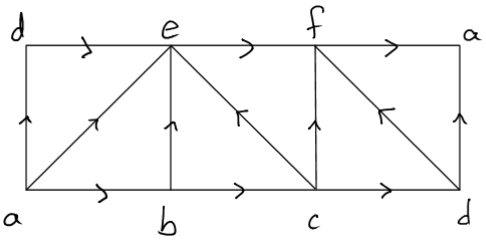


Figure 3.4.1: A complex whose underlying space is a Möbius strip.

The chain groups are

$$C_0(X) = \operatorname{span}\{x_a, x_b, x_c, x_d, x_e, x_f\},$$

$$C_1(X) = \operatorname{span}\{x_{ab}, x_{ad}, x_{ae}, x_{af}, x_{bc}, x_{be}, x_{cd}, x_{ce}, x_{cf}, x_{de}, x_{df}, x_{ef}\},$$

and

$$C_2(X) = \operatorname{span}\{x_{abe}, x_{ade}, x_{adf}, x_{bce}, x_{cdf}, x_{cef}\}.$$

Then the boundary map $C_1 \xrightarrow{\partial_1} C_0$ can be expressed as the 6 x12 matrix A_∂ where

$$A_\partial = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Here the rows represent the 0-simplices x_a, x_b, x_c, x_d, x_e and x_f respectively, and the columns represent the 1-simplices. So the first column represents the boundary of the 1-simplex x_{ab}, which is $\partial_1(x_{ab}) = x_b - x_a$.

Then it can be verified that the rank of A_∂ is 5, so the dimension of A_∂ is 5 and the codimension of A_∂ is 1.

The boundary map A_∂ acts on the set of all oriented 1-simplices, which can be

represented by the matrix B, where

$$B = \begin{bmatrix} x_{ab} \\ x_{ad} \\ x_{ae} \\ x_{af} \\ x_{bc} \\ x_{be} \\ x_{cd} \\ x_{ce} \\ x_{cf} \\ x_{de} \\ x_{df} \\ x_{ef} \end{bmatrix}.$$

Then the product $A_\partial B$ is

$$A_\partial B = \begin{bmatrix} -x_{ab} - x_{ad} - x_{ae} - x_{af} \\ x_{ab} - x_{bc} - x_{be} \\ x_{bc} - x_{cd} - x_{ce} - x_{cf} \\ x_{ad} + x_{cd} - x_{de} - x_{df} \\ x_{ae} + x_{be} + x_{ce} + x_{de} - x_{ef} \\ x_{af} + x_{cf} + x_{df} + x_{ef} \end{bmatrix}.$$

Similarly, the boundary map $C_2 \xrightarrow{\partial_2} C_1$ can be expressed as the 12 x 6 matrix D_∂

where

$$D_\partial = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here the rows represent the 1-simplices and the columns represent the 2-simplices, so the first column represents the boundary of the 2-simplex x_{abe}, which is $\partial_2(x_{abe}) = x_{ab} - x_{ae} + x_{be}$. Note that the rank of D_∂ is 6 and the codimension of D_∂ is also 6.

The property that $\partial_{n-1} \circ \partial_n = 0$ in Theorem 3.1.13 can be verified by taking the matrix product $A_\partial D_\partial$.

Then D_∂ acts on the set of all oriented 2-simplices, which can be represented by the

matrix E, where

$$E = \begin{bmatrix} x_{abe} \\ x_{ade} \\ x_{adf} \\ x_{bce} \\ x_{cdf} \\ x_{cef} \end{bmatrix}.$$

Then the product $D_\partial E$ is

$$D_\partial E = \begin{bmatrix} x_{abe} \\ x_{ade} + x_{adf} \\ -x_{abe} - x_{ade} \\ -x_{adf} \\ x_{bce} \\ x_{abe} - x_{bce} \\ x_{cdf} \\ x_{bce} + x_{cef} \\ -x_{cdf} - x_{cef} \\ x_{ade} \\ x_{adf} + x_{cdf} \\ x_{cef} \end{bmatrix}.$$

Then the chain complex for the Möbius strip is detailed below in Figure 3.4.2.

Note that the dimension of C_2 is 6 since C_2 contains six 2-simplices, namely $x_{abe}, x_{ade}, x_{adf}, x_{bce}, x_{cdf}$, and x_{cef}. The dimension of C_1 is 12 since C_1 contains twelve 1-simplices, namely $x_{ab}, x_{ad}, x_{ae}, x_{af}, x_{bc}, x_{be}, x_{cd}, x_{ce}, x_{cf}, x_{de}, x_{df}$, and x_{ef}. Furthermore, the dimension of C_0 is 6 since C_0 is composed of the six 0-simplices x_a, x_b, x_c, x_d, x_e, and x_f. Then we can compute the homology groups of the Möbius

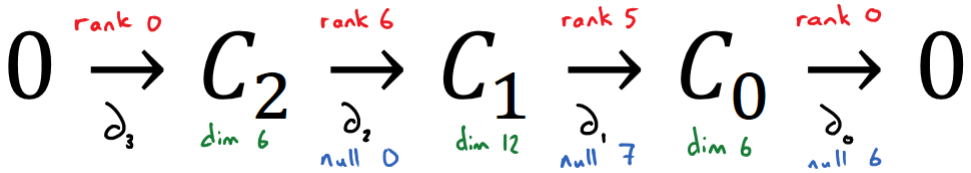


Figure 3.4.2: The chain complex for the Möbius strip.

strip. Recall that the definition of a homology group from (3.7) is

$$H_n = \ker \partial_n / \operatorname{im} \partial_{n+1},$$

and so

$$\dim(H_n) = \operatorname{null}(\partial_n) - \operatorname{rank}(\partial_{n+1}).$$

Then we find that $\dim H_2 = 0$, $\dim H_1 = 1$ and $\dim H_0 = 1$, so the homology groups of the Möbius strip are $H_2 \approx 0$, $H_1 \approx \mathbb{Z}$ and $H_0 \approx \mathbb{Z}$.

CHAPTER 4

LONG EXACT SEQUENCES AND LOCAL HOMOLOGY

4.1 Homotopy

Definition 4.1.1. A **homotopy** between two continuous functions f and g from a topological space X to a topological space Y is defined to be a continuous function $H : X \times [0,1] \to Y$ such that if $x \in X$ then $H(x,0) = f(x)$ and $H(x,1) = g(x)$. The function H describes a continuous deformation of f into g.

Definition 4.1.2. Two functions are said to be **homotopic** if there exists a homotopy between them.

Example 4.1.3. The two bold paths in Figure 4.1.1 are homotopic relative to their endpoints, and the thin blue lines illustrate a homotopy parametrized at a few arbitrary values in $[0,1]$.

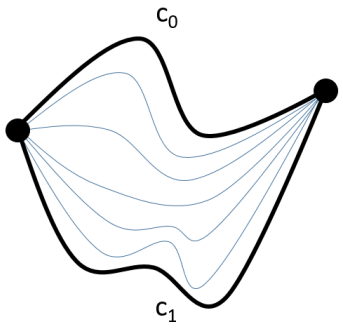


Figure 4.1.1: Homotopic paths. The paths c_0 and c_1 are homotopic relative to their endpoints.

Definition 4.1.4. Given two spaces X and Y, we say that they are **homotopy equivalent** if there exist continuous maps $f : X \to Y$ and $g : Y \to X$ such that $g \circ f$ is homotopic to the identity map id_X and $f \circ g$ is homotopic to id_Y.

Theorem 4.1.5. *Homotopy equivalence is an equivalence relation.*

Proof. We need to check that homotopy equivalence is reflexive, symmetric, and transitive. Homotopy equivalence is reflexive because the map $H : X \times [0,1] \to X$, $H(x,t) = f(x)$ is a homotopy from f to f.

Next we show that homotopy equivalence is symmetric. Suppose $F : X \times [0,1] \to X$ is a homotopy from f to g. Then the map $G : X \times [0,1] \to X$, where

$$G(x,t) = F(x,1-t),$$

is a homotopy from g to f.

Finally, we show that homotopy equivalence is transitive. Suppose $F : X \times [0,1] \to X$ is a homotopy from f to g, and $G : X \times [0,1] \to X$ is a homotopy from g to h. Then the map $H : X \times [0,1] \to X$, where

$$H(x,t) = \begin{cases} F(x,2t) & \text{if } 0 \le t \le 1/2 \\ G(x,2t-1) & \text{if } 1/2 \le t \le 1 \end{cases}$$

is a homotopy from f to h. □

Example 4.1.6. The circle, the annulus, and the open cylinder (not including the lids) are homotopy equivalent (see Figure 4.1.2).

Example 4.1.7. The simplicial complex pictured in Figure 4.1.3 is homotopic to a point, since any simplex with a free face can be collapsed.

Example 4.1.8. Spaces that are homotopy equivalent might not be homeomorphic. For example, an open and closed interval are homotopy equivalent, but not homeomorphic, since a closed interval is compact, but an open interval is not.

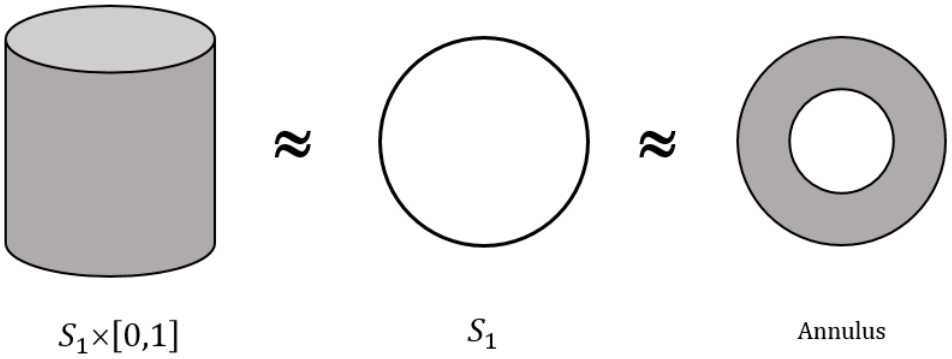


Figure 4.1.2: Homotopy equivalent spaces. The open cylinder, the circle and the annulus are homotopy equivalent.

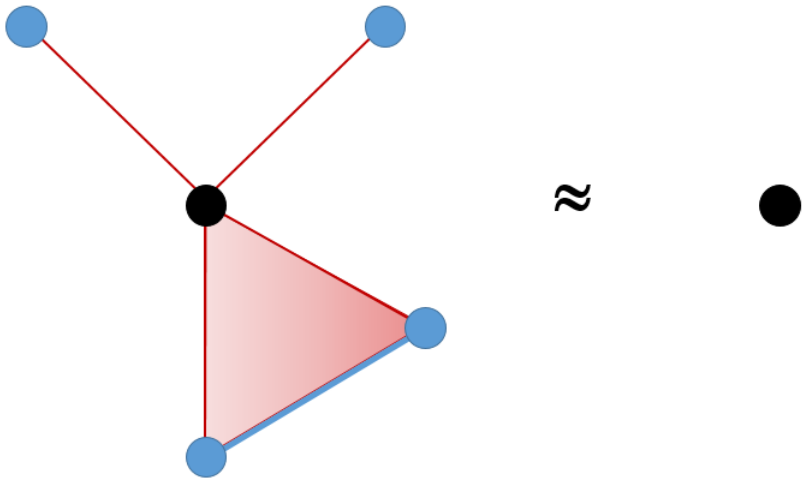


Figure 4.1.3: A simplicial complex which is homotopic to a point.

4.2 Homotopy invariance

For the following discussion, recall the definitions of singular chain complexes and their homology groups from Section 3.3.

Definition 4.2.1. Let $C_n(X)$ and $C_n(Y)$ be singular chain complexes on topological spaces X and Y. For a map $f : X \to Y$, the **induced homomorphism**

$$f_\# : C_n(X) \to C_n(Y)$$

is defined by composing each singular n-simplex $\sigma : \Delta^n \to X$ with f to get a singular n-simplex $f_\#(\sigma) = f\sigma : \Delta^n \to Y$, then extending $f_\#$ linearly via

$$f_\#\left(\sum_i n_i\sigma_i\right) = \sum_i n_i f_\#(\sigma_i) = \sum_i n_i f\sigma_i.$$

Definition 4.2.2. A diagram of maps with the property that any two compositions of maps starting at one point in the diagram and ending at another are equal is called a **commutative diagram**.

Definition 4.2.3. A **chain map** $f_\#$ between singular chain complexes $C_n(X)$ and $C_n(Y)$ is a sequence of homomorphisms that, for each n, commutes with the boundary operators on the two chain complexes (so that $f_\#\partial = \partial f_\#$). Such a map sends cycles to cycles and boundaries to boundaries.

Theorem 4.2.4. *The induced homomorphism $f_\#$ is a chain map.*

$$\begin{array}{ccccccccc}
\cdots & \longrightarrow & C_{n+1}(X) & \xrightarrow{\partial} & C_n(X) & \xrightarrow{\partial} & C_{n-1}(X) & \longrightarrow & \cdots \\
 & & \downarrow{\scriptstyle f_\#} & & \downarrow{\scriptstyle f_\#} & & \downarrow{\scriptstyle f_\#} & & \\
\cdots & \longrightarrow & C_{n+1}(Y) & \xrightarrow{\partial} & C_n(Y) & \xrightarrow{\partial} & C_{n-1}(Y) & \longrightarrow & \cdots
\end{array}$$

Proof. The maps $f_\# : C_n(X) \to C_n(Y)$ satisfy $f_\#\partial = \partial f_\#$ since

$$f_\#\partial(\sigma) = f_\#\left(\sum_i (-1)^i \sigma|[v_0,...,\hat{v}_i,...,v_n]\right) = \sum_i (-1)^i f\sigma|[v_0,...,\hat{v}_i,...,v_n] = \partial f_\#(\sigma).$$

Thus we have a commutative diagram, pictured above. □

Theorem 4.2.5. *A chain map between chain complexes induces homomorphisms between the homology groups of the two complexes.*

Proof. Recall from Definition 3.3.5 that the n-th homology group of a chain complex is $H_n(X) = Z_n(X)/\partial_{n+1}(X)$, where $Z_n(X)$ are the n-cycles and $\partial_{n+1}(X)$ are the n-boundaries.

Let α be a cycle in $C_n(X)$. Then $\partial\alpha = 0$, so $f_\#(\partial\alpha) = 0$, since $f_\#$ is a homomorphism. Now, since $f_\#$ is a chain map, $f_\#\partial = \partial f_\#$, therefore $\partial\left(f_\#\alpha\right) = 0$. Therefore $f_\#\alpha$ is a cycle in $C_n(Y)$, proving that $f_\#$ takes cycles to cycles.

Now let β be a boundary in $C_n(X)$. Then $\beta = \partial\gamma$, for some $\gamma \in C_{n+1}(X)$, so $f_\#\left(\beta\right) = f_\#\left(\partial\gamma\right) = \partial\left(f_\#\gamma\right)$. Therefore images of ∂ (boundaries) map to images of ∂ under $f_\#$, and $f_\#$ is well-defined mod boundaries, i.e. $f_\#$ is well-defined on $H_n(X)$.

Therefore, $f_\#$ induces a homomorphism $f_* : H_n(X) \to H_n(Y)$. □

Theorem 4.2.6. *If two maps $f, g : X \to Y$ are homotopic, then they induce the same homomorphism $f_* = g_* : H_n(X) \to H_n(Y)$ on singular homology.*

See Hatcher for a proof of Theorem 4.2.6 [Hat02, pages 112–113].

Corollary 4.2.7. *The maps $f_* : H_n(X) \to H_n(Y)$ induced by a homotopy equivalence $f : X \to Y$ are isomorphisms for all n.*

Proof. Choose $g : Y \to X$ such that $g \circ f$ is homotopic to the identity map id_X and $f \circ g$ is homotopic to id_Y. Then g_* is the corresponding induced map from $H_n(Y)$ to $H_n(X)$. By properties (i) and (ii) in Hatcher [Hat02, page 111], we know that $\left(fg\right)_* = f_* g_*$ and

$\mathrm{id}_Y = \mathrm{id}_{H_n(Y)}$. Then $g_* \circ f_*$ is homotopic to the identity map $\mathrm{id}_{H_n(X)}$ and $f_* \circ g_*$ is homotopic to $\mathrm{id}_{H_n(Y)}$, and g_* is the inverse of f_*. So the induced maps are isomorphisms. □

4.3 Exact sequences

Definition 4.3.1. An **exact sequence** is a sequence of groups G_i and group homomorphisms f_i

$$G_0 \xrightarrow{f_1} G_1 \xrightarrow{f_2} \ldots \xrightarrow{f_n} G_n$$

where the image of each homomorphism is equal to the kernel of the next: $\mathrm{im}\left(f_k\right) = \ker\left(f_{k+1}\right)$. A **long exact sequence** is an exact sequence with infinitely many groups and homomorphisms.

A number of algebraic concepts can be expressed in terms of exact sequences:

(1) The sequence $0 \to A \xrightarrow{\alpha} B$ is exact if and only if $\ker(\alpha) = 0$, in other words, if α is injective.

(2) The sequence $A \xrightarrow{\alpha} B \to 0$ is exact if and only if $\mathrm{im}(\alpha) = B$, in other words, if α is surjective.

(3) The sequence $0 \to A \xrightarrow{\alpha} B \to 0$ is exact if and only if α is an isomorphism, by (1) and (2).

(4) The sequence

$$0 \to A \xrightarrow{\alpha} B \xrightarrow{\beta} C \to 0 \tag{4.1}$$

is exact if and only if α is injective, β is surjective, and $\ker\left(\beta\right) = \mathrm{im}(\alpha)$, so β induces an isomorphism $C \approx B/\mathrm{im}(\alpha)$. This can be written as $C \approx B/A$, if we think of α as an inclusion of A as a subgroup of B.

Definition 4.3.2. The exact sequence in (4.1) is called a **short exact sequence**.

Definition 4.3.3. Let A_n, B_n and C_n be chain complexes. Let i be the inclusion map of A_n as a subgroup of B_n, let j be the quotient map, and let ∂ be the boundary map. The sequence

$$0 \to A_n \to B_n \to C_n \to 0 \tag{4.2}$$

is a **short exact sequence of chain complexes** if the sequence is a short exact sequence for every n. An expanded version of the sequence in (4.2) is illustrated in Figure 4.3.1, where the columns are chain complexes, the rows are short exact sequences, and the diagram commutes.

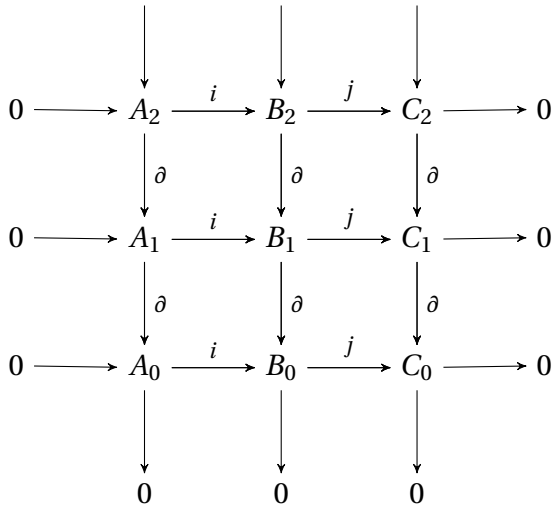


Figure 4.3.1: An expanded commutative diagram of a short exact sequence of chain complexes, where the rows are exact and the columns are chain complexes, which we denote A, B and C.

Definition 4.3.4. Given a subspace $A \subset X$, for each k, we can form the short exact sequence

$$0 \to C_k(A) \to C_k(X) \to C_k(X)/C_k(A) \to 0, \tag{4.3}$$

where $C_k(X)$ denotes the singular chains on the space X. The expanded version of the

short exact sequence in (4.3) is Figure 4.3.2, where i is the inclusion and j is the quotient map.

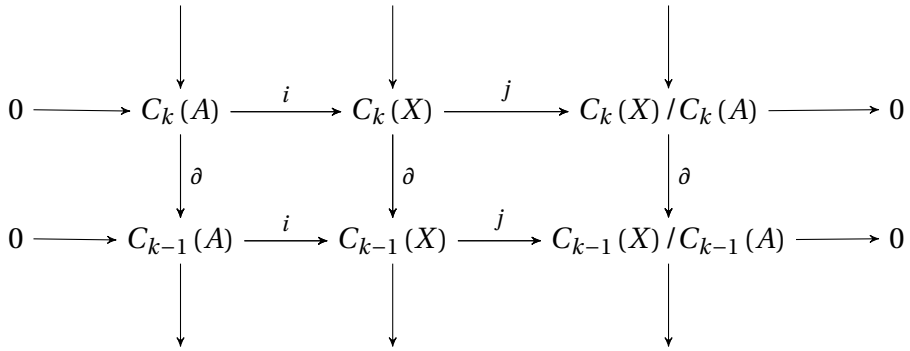


Figure 4.3.2: A short exact sequence of chain complexes.

The corresponding homology is called the **relative homology**, $H_n(X,A)$, where $H_n(X,A)$ is the homology of the right-hand column of the commutative diagram in Figure 4.3.2, and

$$H_n(X,A) = H_n(C_k(X)/C_k(A)). \tag{4.4}$$

Theorem 4.3.5. *The diagram in Figure 4.3.2 is a short exact sequence of chain complexes.*

Proof. Inclusion maps are injective and quotient maps are surjective.

We now show that $\ker(j) = \operatorname{im}(i)$. The image of i is simply $C_k(A)$. The kernel of the quotient map $j : C_k(X) \to C_k(X)/C_k(A)$ is $C_k(A)$. So $\ker(j) = C_k(A) = \operatorname{im}(i)$, and therefore the diagram in Figure 4.3.2 is a short exact sequence of chain complexes. □

Theorem 4.3.6. *A short exact sequence of chain complexes* $0 \to A \xrightarrow{i} B \xrightarrow{j} C \to 0$ *induces a long exact sequence of homology groups (see Figure 4.3.3).*

See Hatcher for a proof of Theorem 4.3.6 [Hat02, pages 116–117].

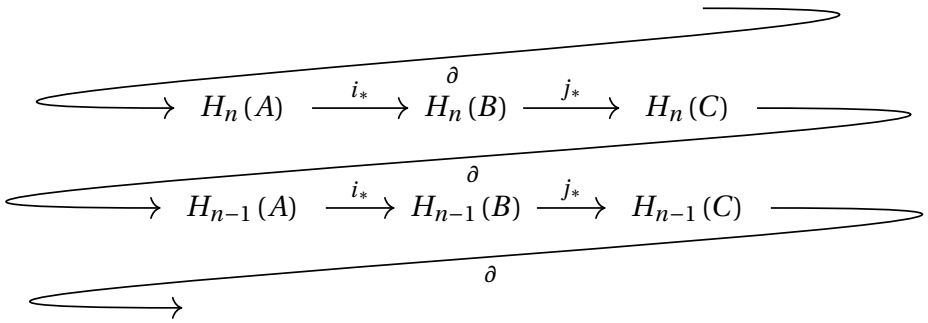


Figure 4.3.3: A long exact sequence of homology groups.

Definition 4.3.7. Given a subspace $A \subset X$, the induced inclusion map i_*, the induced quotient map j_* and the boundary map ∂, Theorem 4.3 yields a **long exact sequence of homology groups**, shown in Figure 4.3.4.

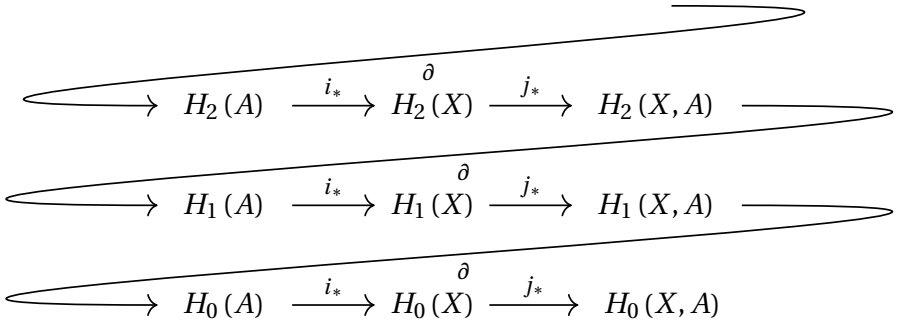


Figure 4.3.4: A long exact sequence of homology groups.

4.4 Excision

Theorem 4.4.1 (Excision Theorem)**.** *Given subspaces $Z \subset A \subset X$ such that the closure of Z is contained in the interior of A, then the inclusion $(X-Z, A-Z) \hookrightarrow (X, A)$ induces isomorphisms $H_n(X-Z, A-Z) \to H_n(X, A)$ for all n. Equivalently, for subspaces $A, B \subset X$ whose interiors cover X, the inclusion $(B, A \cap B) \hookrightarrow (X, A)$ induces isomorphisms $H_n(B, A \cap B) \to H_n(X, A)$ for all n.*

The Excision Theorem is illustrated in Figure 4.4.1. For a proof of the Excision Theorem, see Hatcher [Hat02, Theorem 2.20, pages 119–124].

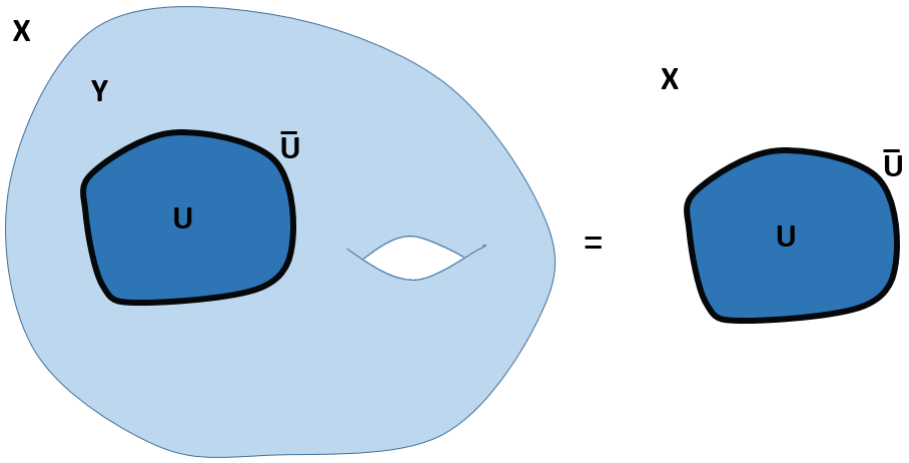


Figure 4.4.1: Through excision, $H_*(X, Y) = H_*\left(\overline{U}, \mathrm{Bd}(U)\right)$.

Sometimes, we desire the homology of a point to be 0. In order to force this case, we can make a small modification to homology theory, called reduced homology, described in detail below.

Definition 4.4.2. Let X be a simplicial complex with chain complex

$$\cdots \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0$$

and homology groups defined by

$$H_n(X) = \ker(\partial_n) / \operatorname{im}(\partial_{n+1}).$$

To define reduced homology, we begin with the augmented chain complex

$$\cdots \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\epsilon} \mathbb{Z} \to 0,$$

where $\epsilon\left(\sum_i n_i \sigma_i\right) = \sum_i n_i$. Now we define the **reduced homology groups** by

$$\tilde{H}_n(X) = \ker(\partial_n) / \operatorname{im}(\partial_{n+1}),$$

for positive n, and

$$\tilde{H}_0(X) = \ker(\epsilon) / \operatorname{im}(\partial_1).$$

Definition 4.4.3. The n-th **local homology group** of a space X at a point x is defined to be $H_n(X, X - \{x\})$. This represents the local homology of X in neighborhoods close to x.

Definition 4.4.4. A **manifold of dimension** n (or n-manifold) is a Hausdorff space M in which each point has an open neighborhood homeomorphic to $\mathbb{R}^n$.

The dimension of M is characterized by the fact that for $x \in M$, the local homology group $H_i(M, M - \{x\}; \mathbb{Z})$ is nonzero only for $i = n$.

By excision, we have

$$H_i(M, M - \{x\}; \mathbb{Z}) \approx H_i\left(\mathbb{R}^n, \mathbb{R}^n - \{0\}; \mathbb{Z}\right).$$

Writing out the long exact sequence, we find

$$\cdots \to H_i\left(\mathbb{R}^n, \mathbb{R}^n - \{0\}\right) \to H_{i-1}\left(\mathbb{R}^n - \{0\}\right) \to H_{i-1}\left(\mathbb{R}^n\right) \to \cdots.$$

Since $\mathbb{R}^n$ is contractible,

$$H_i(M, M - \{x\}; \mathbb{Z}) \approx \tilde{H}_{i-1}\left(\mathbb{R}^n - \{0\}; \mathbb{Z}\right).$$

And since $\mathbb{R}^n - \{0\}$ is homotopy equivalent to S^{n-1},

$$H_i(M, M-\{x\}; \mathbb{Z}) \approx \tilde{H}_{i-1}\left(S^{n-1}; \mathbb{Z}\right).$$

Therefore, if x has a neighborhood homeomorphic to $\mathbb{R}^n$, then $H_i(X, X-\{x\}; \mathbb{Z})$ is $\tilde{H}_{i-1}\left(S^{n-1}; \mathbb{Z}\right)$. In our work, the point x often has a neighborhood homeomorphic to $\mathbb{R}^2$, so the homology groups are those of the circle $\tilde{H}_{i-1}\left(S^1; \mathbb{Z}\right)$. This is the justification for why, if we are attempting to find the local homology of a point in a disk like that of Figure 4.5.10, we can examine the relative homology of the star of the vertex relative to the link of the vertex.

4.5 A survey of the homology groups for possible links in two dimensions

Here we compute the relative homology groups for the star X of a vertex relative to the link A of a vertex for the some possible cases of subcomplexes in two-dimensional space. Note that in each case, the star of the vertex is contractible, so the homology groups are always the same, namely $H_n(X) \approx 0$ for $n \geq 1$ and $H_0(X) \approx \mathbb{Z}$. The relative homology groups can be found directly from the homology groups of the star and the link, so in practice, we concern ourselves with the homology groups of the link of a vertex.

In Figures 4.5.2–4.5.10, we display the long exact sequence $H_*(X, A)$.

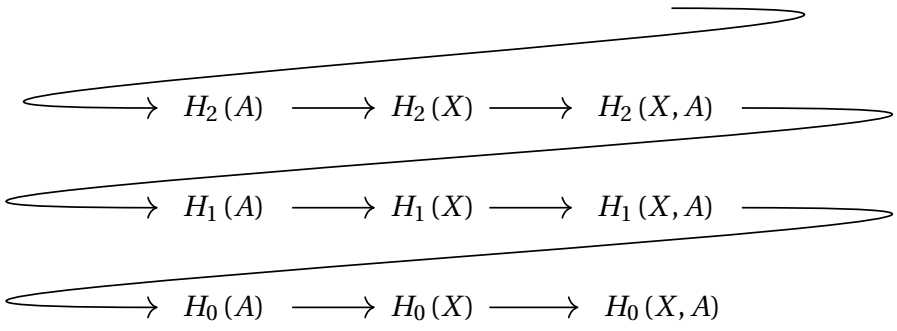


Figure 4.5.1: A long exact sequence of relative homology groups.

Figure 4.5.2: A line segment with X in red and A in blue.

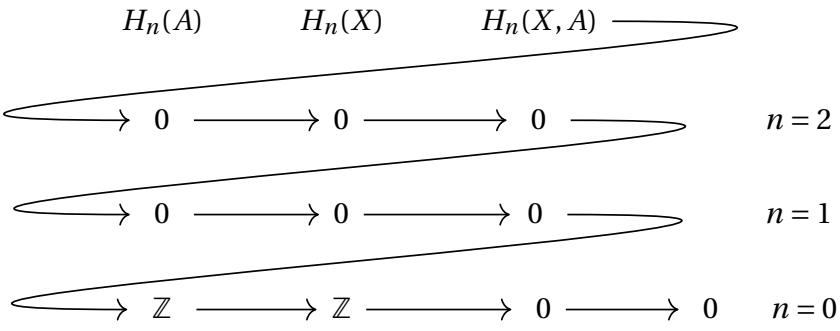


Figure 4.5.3: $H_*(X,A)$ for a line segment.

Figure 4.5.4: A line with X in red and A in blue.

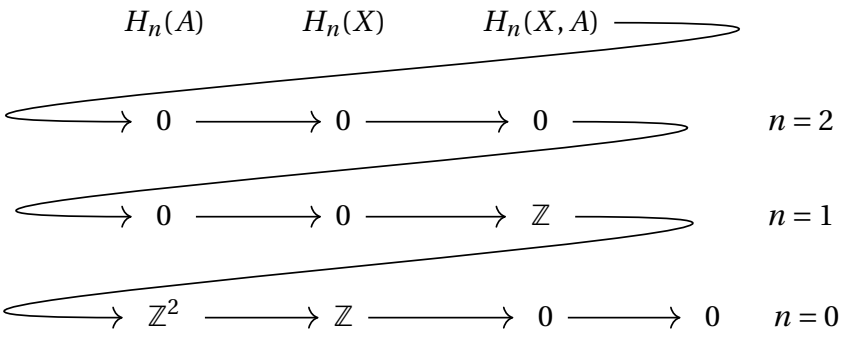


Figure 4.5.5: $H_*(X,A)$ for a line.

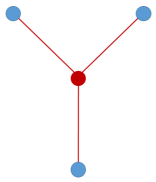


Figure 4.5.6: A y-shaped graph with X in red and A in blue.

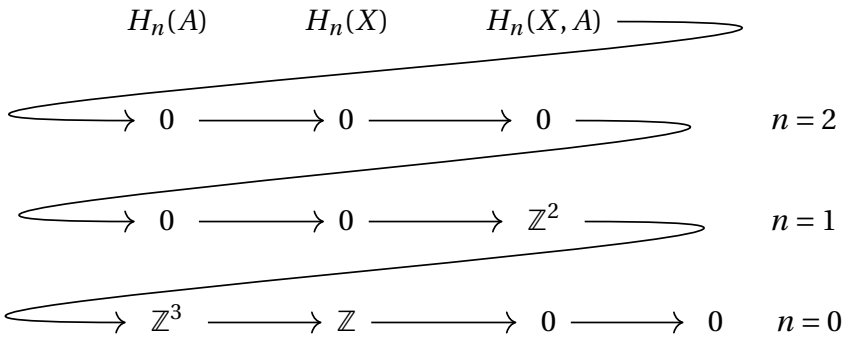


Figure 4.5.7: $H_*(X,A)$ for a y-shaped graph.

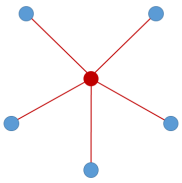


Figure 4.5.8: An n-star with X in red and A in blue.

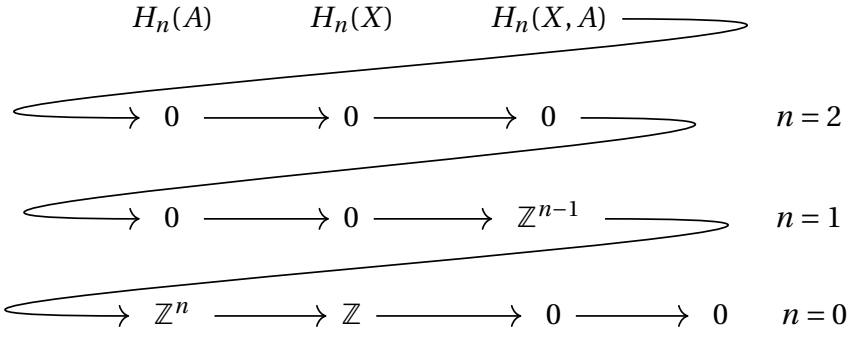


Figure 4.5.9: $H_*(X,A)$ for an n-star.

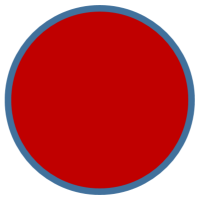


Figure 4.5.10: A disk with X in red and A in blue.

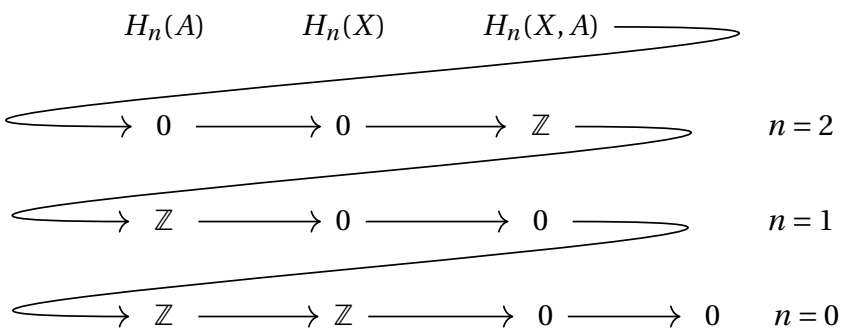


Figure 4.5.11: $H_*(X,A)$ for a disk.

CHAPTER 5

SIMPLICIAL COMPLEXES OF POINT CLOUD DATA

The goal of this work is to take a point cloud X (see Section 1.1), coming from a simplicial complex $\mathscr{X}$, and assign a label to each point $x \in X$ indicating the point's local dimension in X. To do so, we begin with a triangulation of the point cloud with a simplicial complex.

We will now examine some examples of simplicial complexes with the goal of analyzing the complexes used in our main algorithm.

5.1 The Čech and Vietoris-Rips complexes

In this section, we review two well-known simplicial complexes, the Čech and Vietoris-Rips complexes, which were developed as a means of extending homology theory from simplicial complexes to metric spaces.

Definition 5.1.1. The ϵ-cover $X(\epsilon)$ of a point cloud X is a collection of balls of diameter ϵ whose union contains X as a subset, i.e.

$$X(\epsilon) = \bigcup_{x \in X} B_{\epsilon/2}(x).$$

Definition 5.1.2. Given a point cloud X in a metric space M, and a real number $\epsilon > 0$, the **Čech complex** $C_\epsilon(X)$ is the abstract simplicial complex composed of simplices $\sigma \subset X$, where

$$C_\epsilon(X) = \left\{ \sigma \subset X \,\middle|\, \bigcap_{x \in \sigma} B_{\epsilon/2}(x) \neq \emptyset \right\}. \tag{5.1}$$

In other words, the 0-cells of $C_\epsilon(X)$ are the points in X and the 1-cells are edges with length less than ϵ. The n-cells are sets of $n+1$ points that can be enclosed in a ball of radius $\frac{\epsilon}{2}$, or in other words,

$$C_\epsilon(X) = \left\{ \sigma \,\middle|\, \sigma \subset X, \mathrm{rad}(\sigma) < \frac{\epsilon}{2} \right\}. \tag{5.2}$$

Example 5.1.3. Figure 5.1.1 illustrates the formation of a Čech complex. We start with a point cloud (on the left) and form a ball of radius $\frac{\epsilon}{2}$ around each point (in the center). Then for every subset $S \subset X$, add S to the simplicial complex if there is a common point of intersection among all of the $\frac{\epsilon}{2}$-balls in S (on the right).

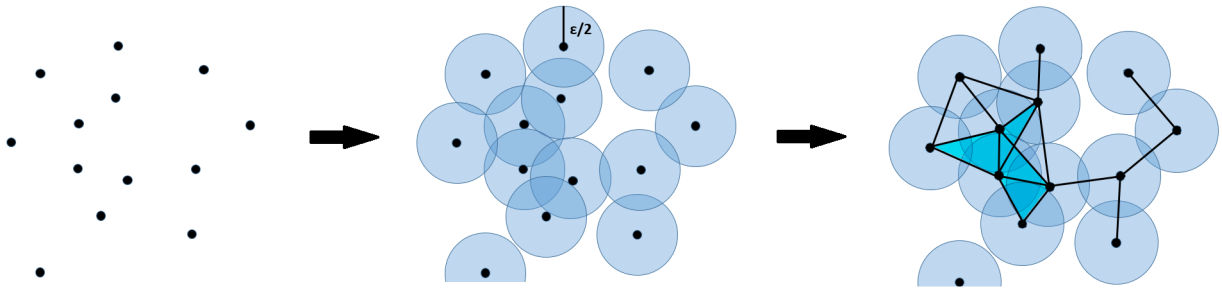


Figure 5.1.1: A Čech complex.

Theorem 5.1.4. *The Čech complex is a simplicial complex.*

Proof. Let X be a point cloud in a metric space M and let ϵ be a positive real number. Suppose that σ is a simplex in the Čech complex $C_\epsilon(X)$. Then for all $x \in \sigma$,

$$\bigcap_{x \in \sigma} B_{\epsilon/2}(x) \neq \emptyset.$$

Therefore, for every simplex $\tau \subset \sigma$,

$$\bigcap_{x \in \tau} B_{\epsilon/2}(x) \supseteq \bigcap_{x \in \sigma} B_{\epsilon/2}(x).$$

So τ is also a simplex in $C_\epsilon(X)$ and therefore the Čech complex is a downset. □

Now, we would like to know whether the Čech complex $C_\epsilon(X)$ resembles the inherent structure underlying the point cloud data X. In other words, the point cloud X was sitting in some metric space M, and the union of the $\frac{\epsilon}{2}$-balls forms a topological space that is close in structure to X. But does the Čech complex have the same

topological structure as the union of the $\frac{\epsilon}{2}$ balls? In the following nerve theorem, we find that indeed they do have the same homotopy types, if certain conditions are met.

Definition 5.1.5. A cover $\{U_i\}$ of a topological space X is a **good cover** if it is an open cover, and if all non-empty finite intersections of cover sets are contractible.

For example, covers by convex sets are good because convex sets are contractible, and the intersection of convex sets is convex.

Theorem 5.1.6 (The Nerve Theorem)**.** *Let X be a point cloud in a metric space M and let ϵ be a real number. Let $C_\epsilon(X)$ be the Čech complex and let $X(\epsilon)$ be the ϵ-cover of X. If the ϵ-cover of X is good, then the homotopy types of $X(\epsilon)$ and $C_\epsilon(X)$ are the same.* □

For the proof of Theorem 5.1.6, see Leray [Ler45] and for further discussion, read De Silva and Ghrist [DSG07, page 13].

The Čech complex $C_\epsilon(X)$ is beneficial because it approximates the topological structure underlying the point cloud data X. However, computing the Čech complex is difficult; in fact, one needs to solve a non-trivial geometric problem exponentially many times to compute the Čech complex. Therefore, we turn to another simplicial complex, the Vietoris-Rips complex.

Definition 5.1.7. Given a set of points X in a metric space M and a positive real number ϵ, the **Vietoris-Rips (VR) complex** $\mathscr{R}_\epsilon(X)$ is the abstract simplicial complex whose k-simplices are determined by subsets of $k+1$ points in X with diameter at most ϵ:

$$\mathscr{R}_\epsilon(X) = \{\sigma \,|\, \sigma \subset X, \operatorname{diam}(\sigma) < \epsilon\}. \tag{5.3}$$

In other words, the 0-cells of $\mathscr{R}_\epsilon(X)$ are the points in X and the 1-cells are the edges of length at most ϵ. The n-cells are sets of $n+1$ points with diameter at most ϵ.

Example 5.1.8. Figure 5.1.2 illustrates the formation of a Vietoris-Rips complex. We start with a point cloud (on the left) and form a ball of radius $\frac{\epsilon}{2}$ around each point (in the center). Then for every subset $S \subset X$ of $k+1$ points, we add S to the simplicial complex if its diameter is at most ϵ (on the right). Note that the Čech complex $C_\epsilon(X)$ is a subset of the Vietoris-Rips complex $\mathscr{R}_\epsilon(X)$, as we will later prove in Theorem 5.1.13. The VR complex and the Čech complex in Figures 5.1.1 and 5.1.2 are constructed from the same underlying point cloud X. In this particular example, the VR complex in Figure 5.1.2 has two additional 1-simplices labeled a and b, and one additional 2-simplex labeled c, when compared with the Čech complex in Figure 5.1.1.

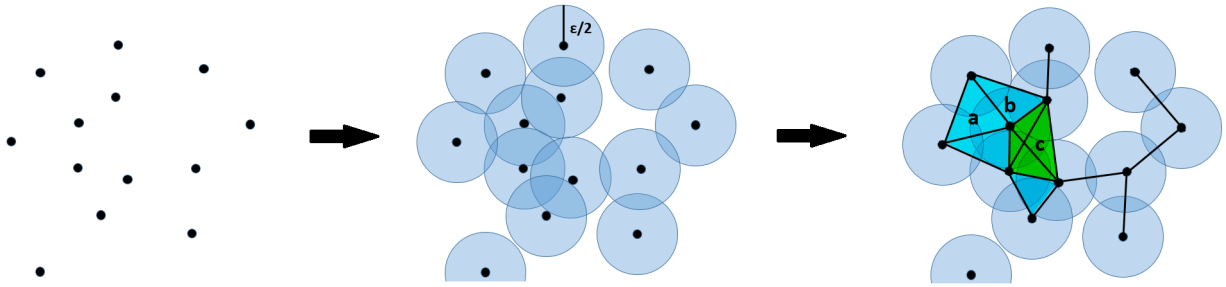


Figure 5.1.2: A Vietoris-Rips complex. The VR complex is a superset of the Čech complex. Comparing with the Čech complex in Figure 5.1.1, the additional simplices are labeled a, b and c.

Example 5.1.9. Figure 5.1.3 shows how the Vietoris-Rips and Čech complexes can differ. Suppose we choose three points whose $\frac{\epsilon}{2}$-balls intersect pairwise, but share no common point of intersection. Then the associated Čech complex has a triangle with no interior there, while the associated Vietoris-Rips complex also includes the interior of that 2-simplex. Figure 5.1.3 also illustrates how the VR complex does not always capture the underlying homology of the ϵ-cover of the point set X.

Definition 5.1.10. The **flag complex** $\mathscr{F}(G)$ of an undirected graph G is an abstract

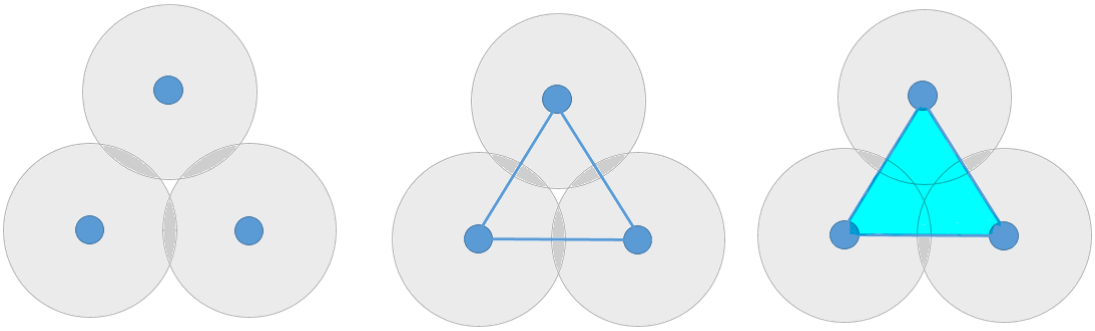


Figure 5.1.3: The difference between a Čech and VR complex. For a set of three points on the left, with a careful choice of parameter, the Čech (center) and VR (right) complexes can differ. The Čech complex has the same topology as the cover, but the homology of the Vietoris Rips complex is different, since it does not contain a hole.

simplicial complex, formed by the sets of vertices in the cliques of G. Specifically, the 0-cells of $\mathscr{F}(G)$ are the vertices of G, the 1-cells of $\mathscr{F}(G)$ are the edges of G, and the n-cells are all cliques of $n+1$ vertices in G. Observe that $\mathscr{F}(G)$ is determined by the 1-skeleton of G.

In order to construct a Vietoris-Rips complex, we can begin by forming the 1-skeleton by adding an edge for every two points whose distance is at most ϵ. We then form the flag complex of the 1-skeleton by adding a simplex of dimension $k-1$ for every clique of k vertices in the 1-skeleton. This algorithm still requires exponential time to compute, but is faster than the construction of the Čech complex, and has the advantage of being defined simply by its 1-skeleton, as shown by Theorem 5.1.12.

Definition 5.1.11. The 1-**skeleton of** $\mathscr{R}_\epsilon(X)$ is the graph whose vertices are points in X and whose edges are all edges of length at most ϵ, denoted as $G_\epsilon(X)$.

Theorem 5.1.12. *Let X be a point cloud and let ϵ be a real number. Suppose we form a graph $G_\epsilon(X)$ by first including an edge for every two points within ϵ, and then generate the flag complex $\mathscr{F}(G_\epsilon(X))$. Then $\mathscr{F}(G_\epsilon(X))$ is $\mathscr{R}_\epsilon(X)$ (the Vietoris-Rips complex with parameter ϵ).*

Proof. Let X be a point cloud and let $G_\epsilon(X)$ be the 1-skeleton of $\mathscr{R}_\epsilon(X)$. Let σ be an n-simplex in $\mathscr{F}(G_\epsilon(X))$. Then σ is composed of $n+1$ vertices and $\binom{n+1}{2}$ edges of length at most ϵ. So the diameter of σ is at most ϵ, and therefore σ is a simplex in the Vietoris-Rips complex $\mathscr{R}_\epsilon(X)$. Now suppose σ is an n-simplex in the VR complex $\mathscr{R}_\epsilon(X)$. Then σ is composed of $n+1$ points with diameter at most ϵ. Then for all $x \in \sigma$, $d(x_1, x_2) < \epsilon$, so every edge is represented in $G_\epsilon(X)$, so the collection of edges is a clique. Therefore σ is in the flag complex $\mathscr{F}(G_\epsilon(X))$. □

Unfortunately, we do not get the benefits of the nerve theorem (Theorem 5.1.6)

from the VR complex, as shown by Figure 5.1.3, but we can compare the Vietoris-Rips complex with the Čech complex.

Theorem 5.1.13. *Let X be a point cloud in a metric space M and let $\epsilon > 0$ be a real number. Let C_ϵ be the Čech complex and let $\mathscr{R}_\epsilon(X)$ be the Vietoris-Rips complex. Then*

$$C_\epsilon(X) \subset \mathscr{R}_\epsilon(X) \subset C_{2\epsilon}(X). \tag{5.4}$$

Proof. Let X be a point cloud in a metric space M and let $\epsilon > 0$ be a real number. Suppose σ is a simplex in the Čech complex $C_\epsilon(X)$. Then there is a common point of intersection between all the $\frac{\epsilon}{2}$-balls centered at the 0-cells in σ. So the $\frac{\epsilon}{2}$-balls intersect pairwise and σ is a simplex in the Vietoris-Rips complex $\mathscr{R}_\epsilon(X)$. Now suppose τ is a simplex in $\mathscr{R}_\epsilon(X)$. Then the $\frac{\epsilon}{2}$-balls centered at the 0-cells of $\mathscr{R}_\epsilon(X)$ intersect pairwise, so by the triangle inequality, if x and z are 0-cells in $\mathscr{R}_\epsilon(X)$, and $y \in B_{\epsilon/2}(x) \cap B_{\epsilon/2}(z)$, then

$$d(x,z) < d(x,y) + d(y,z) < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

Therefore, the maximum distance between any two 0-cells is ϵ. Now form a ball of radius ϵ around each 0-cell in τ. Since the maximum distance between any two 0-cells in τ is ϵ, $\tau \subset B_\epsilon(x)$ for every 0-cell x in τ. So $\bigcap B_\epsilon(\tau) \neq 0$. Therefore $\tau \subset C_{2\epsilon}(X)$. □

So as long as the Čech complexes for ϵ and 2ϵ are good approximations for the underlying space of the point cloud, then so is the Vietoris-Rips complex for ϵ. See also de Silva and Ghrist [DSG07, Theorem 6, page 16], who have strengthened these inclusions when $M = \mathbb{R}^N$.

5.2 The Local Complex

This section contains the first main conceptual ideas contributed in this thesis – viz. the definition of a new local neighborhood complex, the Local Complex, as well as a

family of related variations.

The notion of Local Complex that we will define shortly will be shown to be closely related to the link of a point in the VR complex. Further, we will show that this new idea tries to capture the role of any particular point in a point cloud's topological shape, at some scale.

Definition 5.2.1. Suppose p is an arbitrary vertex in a point cloud X in $\mathbb{R}^N$. For $\alpha, \beta > 0$, let G be the graph with vertex set V and edge set E defined by

$$\begin{aligned} V &= \left\{ x \in X, x \neq p \,\middle|\, N_{\alpha/2}(x) \cap N_{\beta/2}(p) \neq 0 \right\} \\ &= \left\{ x \in X, x \neq p \,\middle|\, d(x,p) < \frac{\alpha+\beta}{2} \right\}, \\ E &= \{ x_1, x_2 \in X \,|\, d(x_1, x_2) < \alpha \}. \end{aligned} \tag{5.5}$$

We define the **Local Complex** $L(X, p, \alpha, \beta)$ to be the flag complex of the graph G. In other words, the 0-cells of the Local Complex are all the points in X whose ball of diameter α intersects a ball of diameter β centered at p. The 1-cells are all edges with length at most α. The n-cells are cliques of $n+1$ points in the graph G.

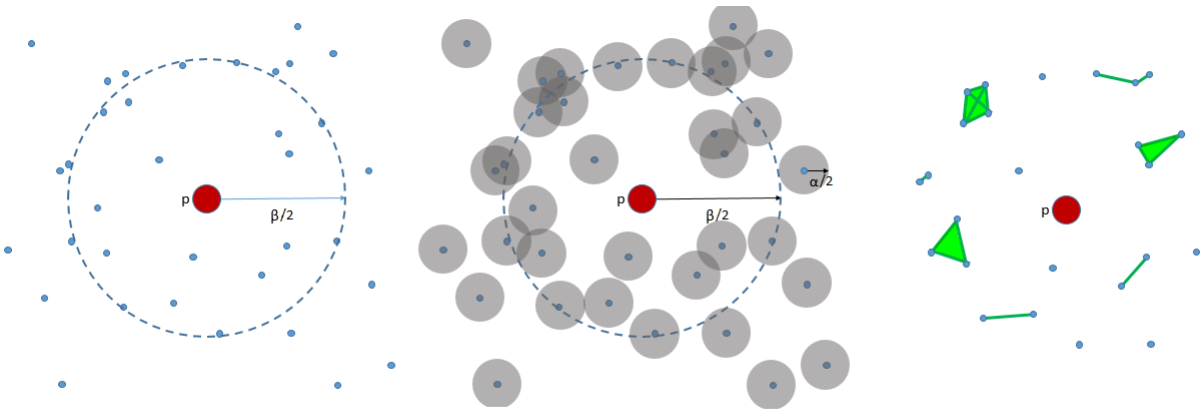


Figure 5.2.1: The formation of a Local Complex about point p with parameters α and β.

In the following theorem, we prove a special case equivalency between the Local Complex and the VR complex, that helps to further ground intuition regarding their close relationship.

Theorem 5.2.2. *The link of a vertex p in the Vietoris-Rips complex $\mathscr{R}_\epsilon(X)$ is precisely the Local Complex $L\big(X,p,\epsilon,\epsilon\big)$.*

Proof. Suppose X is a point cloud with Vietoris-Rips complex $\mathscr{R}_\epsilon(X)$. Let p be an arbitrary point in X. Recall that the link of p is the set of all faces opposite p for every simplex containing p. Suppose $L\big(X,p,\epsilon,\epsilon\big)$ is the Local Complex of p (so $\alpha=\beta=\epsilon$). Suppose x is a 0-cell in the link of p in the VR complex $\mathscr{R}_\epsilon(X)$. Then x and p are joined by a 1-cell, so $d\big(x,p\big)<\epsilon=\big(\alpha+\beta\big)/2$. Therefore x is also a 0-cell in the Local Complex. Now suppose x is a 0-cell in the Local Complex. Then $d\big(x,p\big)<\big(\alpha+\beta\big)/2$. So $d\big(x,p\big)<\epsilon$. So there is a 1-cell connecting x and p. So x is a face of a simplex containing p. Therefore the 0-cells in the link of p in the VR complex with parameter ϵ are the same as the 0-cells in the Local Complex $L\big(X,p,\epsilon,\epsilon\big)$.

Now suppose $\tau=[x_1x_2]$ is a 1-simplex in the link of p in $\mathscr{R}_\epsilon(X)$. Then $d(x_1,x_2)<\epsilon=\alpha$. So τ is also a 1-simplex in the Local Complex $L\big(X,p,\epsilon,\epsilon\big)$.

The converse is true by the same argument. Since both $\mathscr{R}_\epsilon(X)$ and $L\big(X,p,\epsilon,\epsilon\big)$ are flag complexes of the same graph G, we have $\text{link}\big(p,\mathscr{R}_\epsilon(X)\big)=L\big(X,p,\epsilon,\epsilon\big)$. □

Theorem 5.2.3. *Let X be a point cloud and let p be a point in X. The link of p in $\mathscr{F}(G)$, denoted $\text{link}\big(p,\mathscr{F}(G)\big)$, is the flag complex of all points adjacent to p in G.*

The proof of Theorem 5.2.3 is discussed in Bridson and Haefliger [BH11, Remarks 5.16, page 210].

5.2.1 Variations on the Local Complex

In this section, we describe a series of variations on the Local Complex idea, each of which corresponds to a different experimental approach tried in the course of this

research.

Definition 5.2.4. Let p be a point in a metric space M. Then

$$S_\epsilon(p) = \{x \in M | d(x,p) = \epsilon\}$$

is the **sphere** of radius ϵ around point p.

The Local Shell Complex allows us to examine the local neighborhood of a vertex p in our point cloud, while excluding all points very close to p. This is beneficial because points extremely close to p might introduce distortions into the characterization of the topology of the local neighborhood of p. Points nearby p could also be considered as measurement error in the determination of the position of p.

Definition 5.2.5. Suppose p is an arbitrary vertex in a point cloud X in $\mathbb{R}^N$. Let G be the graph with vertex set V and edge set E defined by

$$\begin{aligned} V &= \left\{x \in X, x \neq p \,\middle|\, N_{\alpha/2}(x) \cap S_{\beta/2}(p) \neq 0\right\} \\ &= \left\{x \in X \,\middle|\, \frac{\beta-\alpha}{2} < d(x,p) < \frac{\beta+\alpha}{2}\right\}, \\ E &= \{x_1, x_2 \in X \,|\, d(x_1,x_2) < \alpha\}. \end{aligned} \tag{5.6}$$

We define the **Local Shell Complex** $L_0(X,p,\alpha,\beta)$ to be the flag complex of the graph G. In other words, the 0-cells of the Local Shell Complex are all the points in X whose ball of diameter α intersects a sphere of diameter β centered at p. The 1-cells are all edges with length at most α. The n-cells are cliques of $n+1$ points in the graph G.

The Local Central Hole Complex, defined below, introduces a third parameter, γ, that serves as a lever to control the size of the central hole, in a manner independent of α and β.

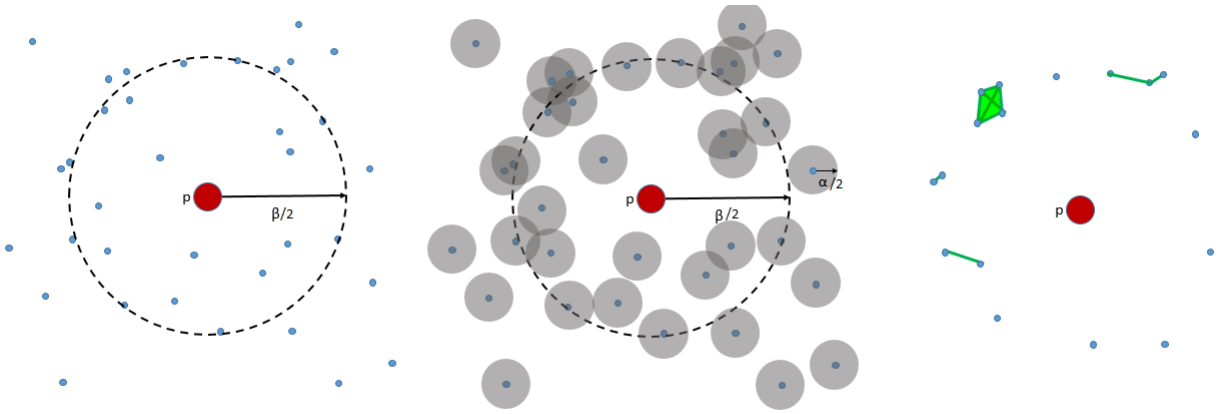


Figure 5.2.2: The formation of a Local Shell Complex about point p with parameters α and β.

Definition 5.2.6. Suppose p is an arbitrary vertex in a point cloud X in $\mathbb{R}^N$. Let G be the graph composed of vertex set V and edge set E defined by

$$V = \left\{ x \in X \,\middle|\, \max\left(\gamma, \frac{\beta-\alpha}{2}\right) < d(x,p) < \frac{\beta+\alpha}{2} \right\},$$
$$E = \{x_1, x_2 \in X \,|\, d(x_1,x_2) < \alpha\}. \tag{5.7}$$

We define the **Local Central Hole Complex** $L_1(X,p,\alpha,\beta,\gamma)$ to be the flag complex of the graph G. In other words, the 0-cells of the Local Shell Complex are all the points in X whose ball of diameter α intersects a sphere of diameter β centered at p, but excluding any points of distance less than or equal to γ from p. The 1-cells are all edges with length at most α. The n-cells are cliques of $n+1$ points in the graph G.

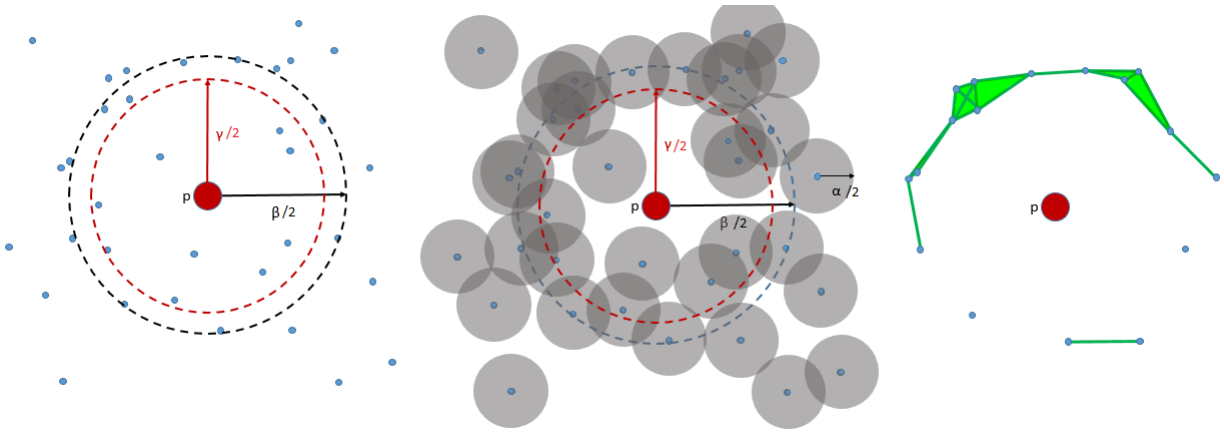


Figure 5.2.3: The formation of a Local Central Hole Complex about point p with parameters α, β and γ.

Another way to minimize the bias introduced by points very close to the vertex p, is to prevent edges from being formed across the central hole. We achieve this variation with the Acute Local Complex, described below.

Definition 5.2.7. Suppose p is an arbitrary vertex in a point cloud X in $\mathbb{R}^N$. Let $L(X,p,\alpha,\beta)$ be a Local Complex, in the sense of Definition 5.2.1, 5.2.5, or 5.2.6. Let x_1

and x_2 be points in the point cloud X. Let $(p - x_1)$ and $(p - x_2)$ be the vectors from p to x_1 and p to x_2, respectively. Then the **Acute Local Complex** $L^*(X, p, \alpha, \beta)$ selects only edges in the Local Complex that do not "cross" the p-shell, by stipulating that the dot product of the vectors formed from p to either end point of the edge $\overline{x_1 x_2}$ is positive, or in other words,

$$(p - x_1) \cdot (p - x_2) > 0. \tag{5.8}$$

Figure 5.2.4 gives an example of an edge which would be disallowed under the dot product constraint in the Acute Local Complex. Such an edge does not represent the homology of the points surrounding p.

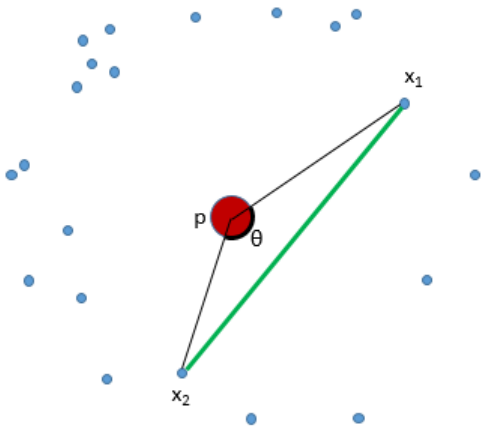


Figure 5.2.4: An edge that would be disallowed in the Acute Local Complex.

Effectively, the complexes and modifications described in Definitions 5.2.5, 5.2.6 and 5.2.7 achieve a similar result, with subtle variations. In practice, and for most of the experimental results reported in this research, we have used the Local Complex in Definition 5.2.1 with the acute constraint in Definition 5.2.7.

CHAPTER 6

APPROXIMATING LOCAL HOMOLOGY

In this chapter, we explore different types of local homologies, characterized using link homology, that can typically arise in two-dimensional and three-dimensional point cloud data.

We present a geometric operation on a Local Complex, the simplex arc projection, that appears to approximate the homology of L fairly well.

This method that will then serve to motivate an intuitive and novel heuristic and algorithm, presented in the next chapter, that we believe captures the essential topology of point cloud data.

Remark 6.0.1. It can be shown that the link of a vertex p in a complex embedded in $\mathbb{R}^2$ has only H_0 and $H_1 \neq 0$.

6.1 Simplex arc projection

Definition 6.1.1. Let X be a point cloud and let $\mathscr{S}$ be a simplicial complex embedded in $\mathbb{R}^2$. We define the **simplex arc projection** of the link of a vertex p in the simplicial complex as follows: take every simplex in the link of vertex p and project it onto a unit circle centered at p (note that overlapping arcs are automatically merged). More specifically, given a point x in a simplex in the link of vertex p, the coordinates of the projected point are given by

$$f(x) = \frac{x-p}{|x-p|}. \tag{6.1}$$

Remark 6.1.2. It can be shown that the n-simplices for $n > 1$ have the same image under the simplex arc projection as the 1-skeleton.

Example 6.1.3. In general, the simplex arc projection of the link of p in the VR complex is not homotopy equivalent to the link. A counterexample from actual data is provided

in Figure 6.1.1. The simplex arc projection is not homotopy equivalent to the link because the link of p has a hole, labeled a, while the projection, labeled s, has no holes.

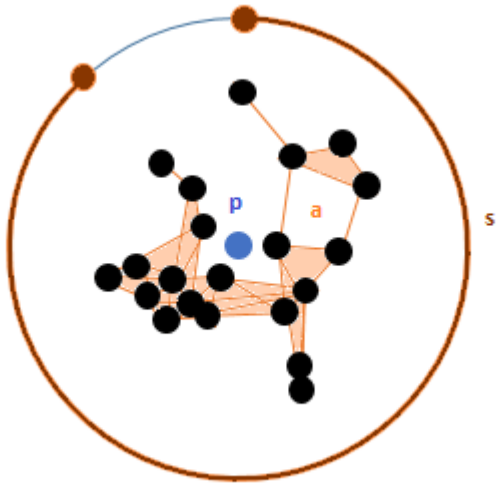


Figure 6.1.1: The link of p in the VR complex is not always homotopic to the simplex arc projection s. In this example, the link has a hole, labeled a, while the arc projection has no holes.

Remark 6.1.4. Recall that by Theorem 5.2.2, the link of a vertex in the VR complex is the same as the Local Complex (when $\epsilon = \alpha = \beta$). Therefore, the simplex arc projection operation is automatically applicable to the Local Complex. This forms the basis of the main algorithm of this thesis (Chapter 7).

We present two examples to illustrate this operation and to establish the relationship between the simplex arc projections and the local homology of the link of a vertex in the VR complex. In the first example (Figures 6.1.2 and 6.1.3), we have a link with five cells (four 0-simplices and one 1-simplex) and a projection with three connected components. The 0-homology of the link is $\mathbb{Z}^3$ and its rank is the number of components in the projection.

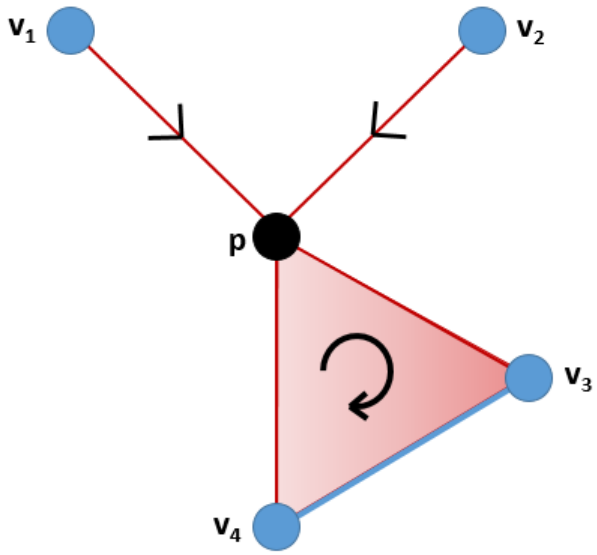


Figure 6.1.2: An oriented simplicial complex. The star of p is in red and the link of p is in blue.

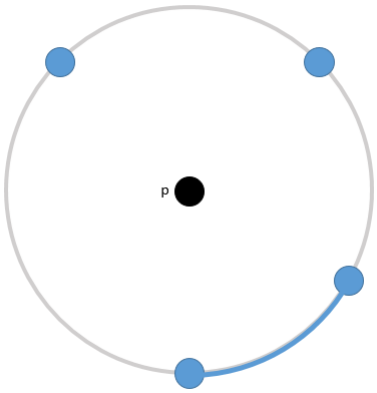


Figure 6.1.3: The projection of the simplicial complex in Figure 6.1.2 onto the unit circle.

Example 6.1.5. We compute the homology of the link (A) of the vertex p for the simplicial complex illustrated in Figure 6.1.2. The link A is the four vertices $[v_1], [v_2], [v_3]$, and $[v_4]$ and the oriented 1-simplex $[v_3 v_4]$. The chain complex is

$$\ldots \xrightarrow{\partial_3} C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0,$$

where C_n are the formal sums of singular n-simplices with integer coefficients and ∂_n are the boundary homomorphisms. We begin by describing the specific chain groups of the link of p, namely, C_0 is isomorphic to $\mathbb{Z}^4$ with basis $[v_1], [v_2], [v_3]$, and $[v_4]$, and C_1 is isomorphic to $\mathbb{Z}$ with basis given by the oriented 1-simplex $[v_3 v_4]$. The chain groups in other dimensions are zero. The boundary homomorphism $\partial_1 : C_1 \to C_0$ is

$$\partial_1 ([v_3 v_4]) = [v_4] - [v_3].$$

The homology groups are, by definition,

$$H_n (A) = \ker \partial_n / \operatorname{im} \partial_{n+1}.$$

Then the 0^{th} homology group of the link is

$$H_0 (A) = \ker \partial_0 / \operatorname{im} \partial_1.$$

The kernel of the 0^{th} boundary map $\ker \partial_0$ is $\langle [v_1], [v_2], [v_3], [v_4] \rangle$. The image of the first boundary map $\operatorname{im} \partial_1$ is $\langle [v_3], [v_4] \rangle$. Therefore the 0^{th} homology group $H_0 (A)$ is isomorphic to $\mathbb{Z}^3$ with basis given (for example) by the vertices $[v_1], [v_2]$, and $[v_3]$. (That $H_0 (A)$ is isomorphic to $\mathbb{Z}^3$ indicates that the link is composed of three disconnected components).

As one can see in Figure 6.1.3, the projection of the link onto the unit circle also has three components.

The first homology group of the link is

$$\begin{aligned} H_1 (A) &= \ker \partial_1 / \operatorname{im} \partial_2 \\ &= 0/0 = 0. \end{aligned}$$

Similarly, all higher homology groups of the link are also 0.

Now we will compute the homology groups of the closed star X of the vertex p. The closed star is composed of five 0-simplices, five 1-simplices and one 2-simplex. The boundary homomorphism $\partial_1 : C_1 \to C_0$ is

$$\begin{aligned}
\partial_1\left(-[p,v_1]\right) &= -\left([v_1]-[p]\right) = [p]-[v_1],\\
\partial_1\left(-[p,v_2]\right) &= [p]-[v_2],\\
\partial_1\left([p,v_3]\right) &= [v_3]-[p],\\
\partial_1\left([v_3,v_4]\right) &= [v_4]-[v_3],\\
\partial_1\left(-[p,v_4]\right) &= [p]-[v_4].
\end{aligned}$$

Then the 0^{th} homology group of the closed star is

$$\begin{aligned}
H_0(X) &= \ker\partial_0 / \operatorname{im}\partial_1\\
&= Z^5/Z^5 = Z.
\end{aligned}$$

The first homology group of the closed star is

$$\begin{aligned}
H_1(X) &= \ker\partial_1 / \operatorname{im}\partial_2\\
&= 0.
\end{aligned}$$

The kernel of the first boundary map ($\ker\partial_1$) is generated by the cycle $\left([p,v_3]+[v_3,v_4]-[p,v_4]\right)$. The image of the second boundary map ($\operatorname{im}\partial_2$) is also generated by the cycle $\left([p,v_3]+[v_3,v_4]-[p,v_4]\right)$. So the first homology group is isomorphic to 0. This implies that the closed star contains no one-dimensional holes. We summarize these computations in the long exact sequence in Figure 6.1.4.

We can generalize the definition of simplex arc projection to any dimension.

Definition 6.1.6. Let X be a point cloud and let $\mathscr{S}$ be a simplicial complex embedded in $\mathbb{R}^n$. We define the **simplex projection** of the link of a vertex p in the complex as

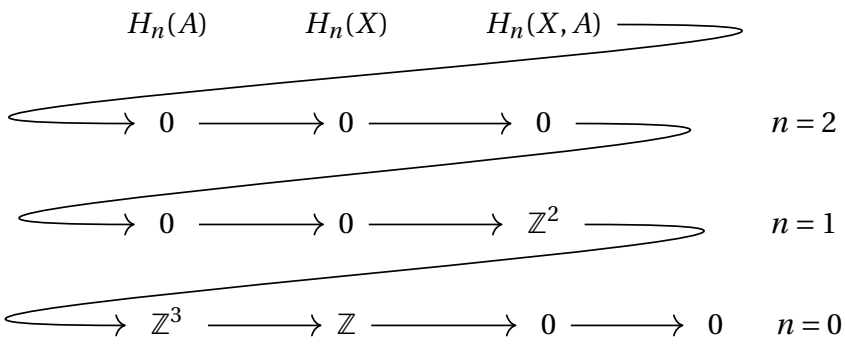


Figure 6.1.4: The long exact sequence of the simplicial complex in Figure 6.1.2.

follows: take every simplex in the link of vertex p and project it onto a unit $(n-1)$-sphere centered at p. More specifically, given a point x in a simplex in the link of vertex p, the coordinates of the projected point are given by

$$f(x) = \frac{x-p}{|x-p|}.$$

Computation of the image of the simplex projection is much more complicated than the simplex arc projection and is a goal of future work.

Example 6.1.7. We illustrate the application of the procedure in three-dimensional space. Let the complex in Figure 6.1.5 be a simplicial complex in $\mathbb{R}^3$. The link of the vertex p is the eight vertices and four edges colored blue. In Figure 6.1.7, we illustrate the intersection of the simplicial complex in Figure 6.1.5 with a unit sphere centered at p. The projection on the unit sphere is the equator and four vertices pictured in Figure 6.1.8. The projection is related to the long exact sequence in 6.1.6. For example, the long exact sequence has $H_0(A) = \mathbb{Z}^5$, which corresponds to the five components on the projection on the sphere.

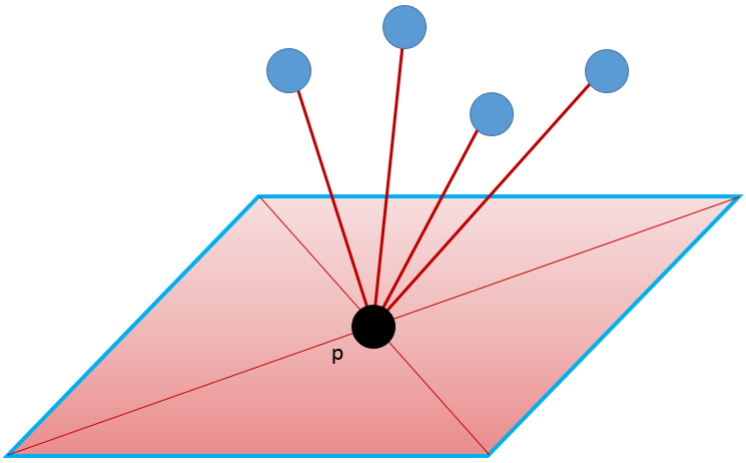


Figure 6.1.5: A fern.

$$
\begin{array}{ccccccc}
 & H_n(A) & & H_n(X) & & H_n(X,A) & \\
\longrightarrow & 0 & \longrightarrow & 0 & \longrightarrow & \mathbb{Z} & \qquad n=2 \\
\longrightarrow & \mathbb{Z} & \longrightarrow & 0 & \longrightarrow & \mathbb{Z}^4 & \qquad n=1 \\
\longrightarrow & \mathbb{Z}^5 & \longrightarrow & \mathbb{Z} & \longrightarrow & 0 \longrightarrow 0 & \qquad n=0
\end{array}
$$

Figure 6.1.6: The long exact sequence of the simplicial complex in Figure 6.1.5.

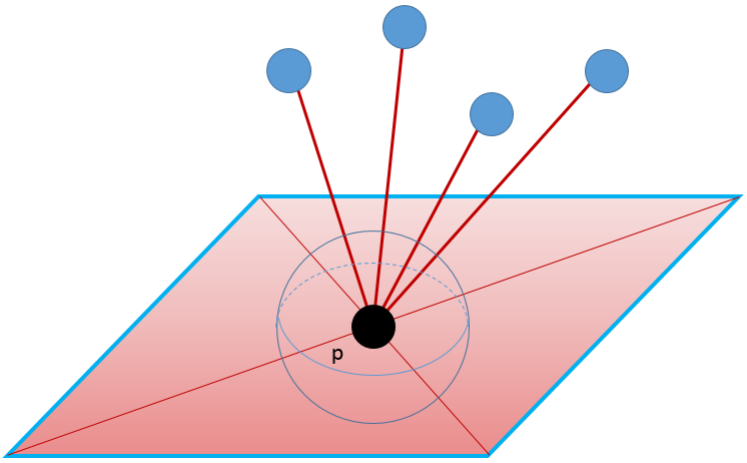


Figure 6.1.7: The intersection of the simplicial complex in Figure 6.1.5 with the unit sphere.

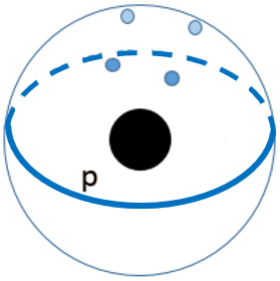


Figure 6.1.8: The projection of the simplicial complex in Figure 6.1.5 onto the unit sphere.

CHAPTER 7

THE ALGORITHM

Given a point cloud X, the algorithm assigns to each vertex $p \in X$ a color based on its apparent local dimension.

To do so, we begin by selecting two parameters: α and β. We form what we call an Acute Local Complex $L^*\left(X, p, \alpha, \beta\right)$ (Definitions 5.2.1 and 5.2.7). We then project the Acute Local Complex onto a unit circle centered at p and determine the homology of the union of the resulting arcs. We then color the whole data set based on the homologies of the simplex arc projections. Along the way, we speed up the computations by binning the points. We also consider only the 1-skeleton of the Acute Local Complex since we are projecting onto the circle and need not compute all the triangles and higher-dimensional tetrahedra in the complex.

Following this, we get the dataset to instruct the choice of α and β by examining a wide range of choices of α and β and studying the rate of change of the simplex projection-based coloring as we vary α and β. The values of α and β with minimal coloring change are flagged as stable.

7.1 Overview

The main algorithmic contributions of this thesis are two-fold:

(1) **Core algorithm:** For a fixed parametrized scale, we first construct the Acute Local Complex, followed by the simplex arc projections, and finally we determine a heuristic characterization of local homology.

(2) **Multiscale iteration:** Iteration of step (1) at different scales in order to understand interesting scales for the data.

The next two sections explain these aspects in depth.

7.2 Details of the algorithm

Here we describe the algorithm for computing the Local Shell Complex $L_0(X, p, \alpha, \beta)$ and the Local Central Hole Complex $L_1(X, p, \alpha, \beta, \gamma)$ as described in Definitions 5.2.5 and 5.2.6.

7.2.1 Binning

We begin by preprocessing the data for more efficient computation by binning the point cloud data (lines 146–190 of the appendix). Given a set of points, we find its bounding box (see Figure 7.2.1) and then add a buffer of $\beta + \alpha$ in all directions. This ensures that the $\beta + \alpha$ disc of each point (including, especially, the boundary points) is contained in the bounding box.

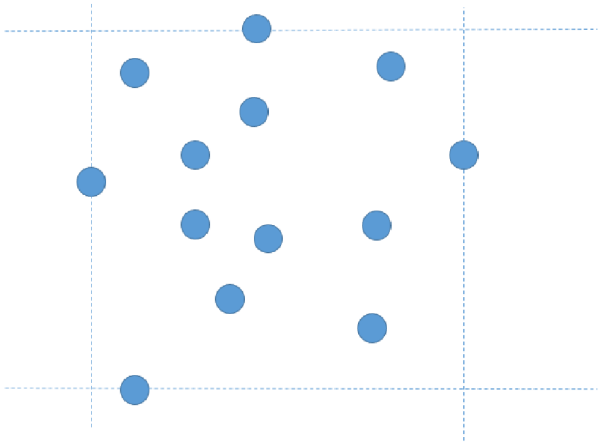


Figure 7.2.1: The bounding box of a vertex set.

We then align the bottom and left sides of the bounding box to the nearest grid cell (see Figure 7.2.2). The binning procedure divides the bounding box of the data set into a grid, whose scale is given by the user. It returns the points in association with their containing cells.

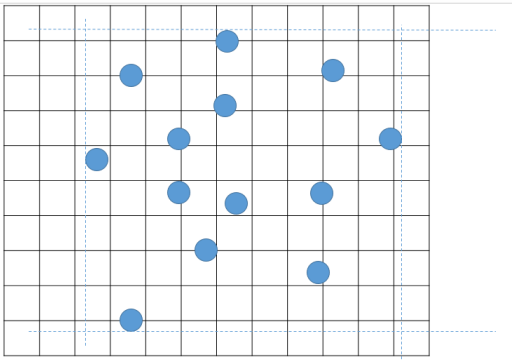


Figure 7.2.2: The extended bounding box with buffer for the points' disks, as well as the binning grid.

7.2.2 Candidate points in the bounding square

For each point, p, in the point cloud data, we return a small, though not exact, set of candidate points that lie within $(\beta+\alpha)/2$ of p (lines 192–209 in the appendix). Concretely, we return points in a large enough square region that properly contains the $(\beta+\alpha)/2$ disc around p (see Figure 7.2.3).

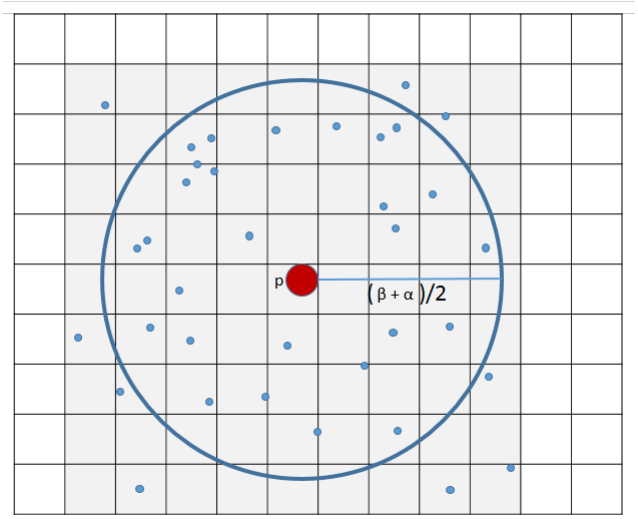


Figure 7.2.3: Local Complex candidate points. The candidate points within a circumscribed square around a neighborhood of radius $(\alpha+\beta)/2$ about p.

7.2.3 Points in the Local Complex

Given the set of candidate points obtained earlier (cf. 7.2.2), we return the points that are present in the sphere of radius $(\beta+\alpha)/2$ (see Figure 7.2.4 and lines 211–220 of the appendix).

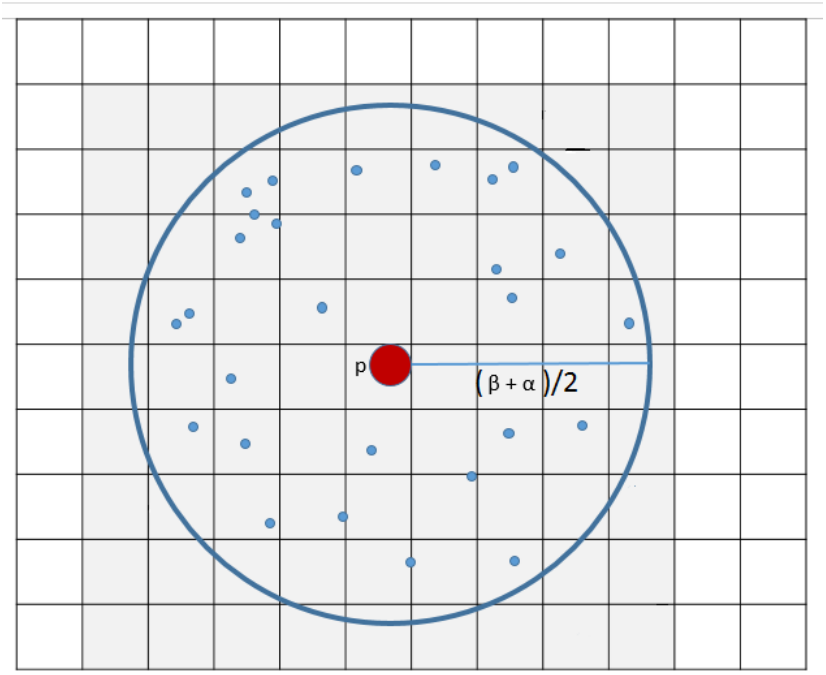


Figure 7.2.4: The Local Complex points. The points that lie in a sphere of radius $(\beta+\alpha)/2$.

.

7.2.4 Variations on the Local Complex

Here we recall variations of the Local Complex previously discussed in Chapter 5.

The Local Shell Complex If we wish to have our local complex avoid points very close to p, we use the Local Shell Complex described in Definition 5.2.5. The Local Shell variation is illustrated in Figure 7.2.5.

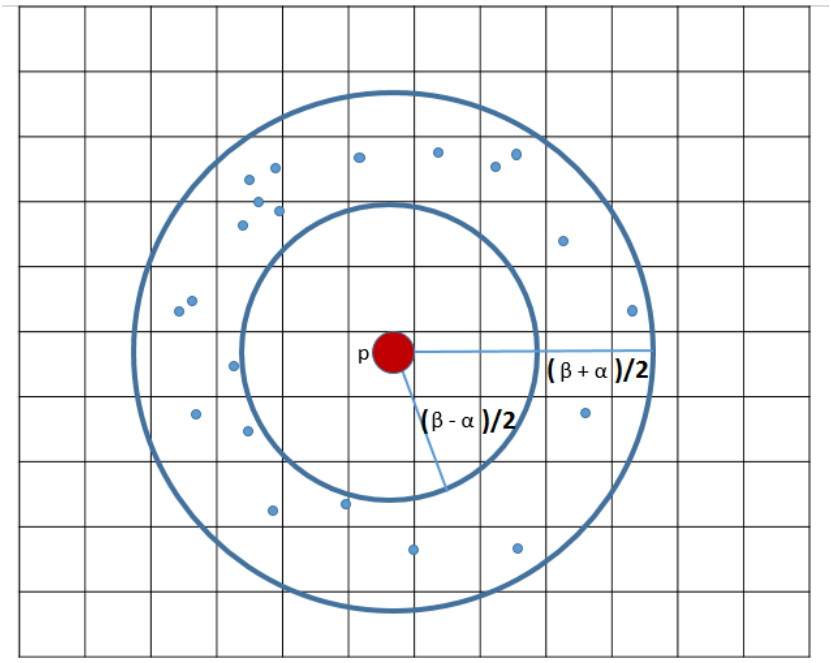


Figure 7.2.5: The Local Shell Complex consists of the points that lie in a shell of inner radius $(\beta - \alpha)/2$ and outer radius $(\beta + \alpha)/2$.

The Local Central Hole Complex Sometimes, we would like to set $\alpha > \beta$. In this case, we may use another parameter γ to ensure that points close p are still avoided. This is done by simply considering points in the modified shell with inner radius $\max(\gamma, \beta - \alpha)$ and outer radius as before $(\beta + \alpha)$. This is the Local Central Hole Complex described in Definition 5.2.6 and line 215 of the appendix.

The Acute Local Complex This complex, described in Definition 5.2.7 is the main variation used in most of the experimental work supporting this thesis. The implementation of this variation will be discussed in detail in the next subsection.

7.2.5 Computation of Acute Local Complex

For each point p and points in its $\alpha\beta$ shell, as constructed above (see 7.2.3), we obtain the 1-skeleton, i.e. edges and vertices, of the Acute Local Complex. In particular, we pick edges that are smaller than some parameter ϵ (see Figure 7.2.6). In practice, we choose ϵ to be equal to α. Further, we apply the acute variation described in Definition 5.2.7 by ensuring that the edges do not cross the shell, i.e. we only pick edges such that the dot product of the vectors formed from p to either end point is positive, implying that the angle between these vectors is acute (see Figure 5.2.4 and lines 90–102 of the appendix).

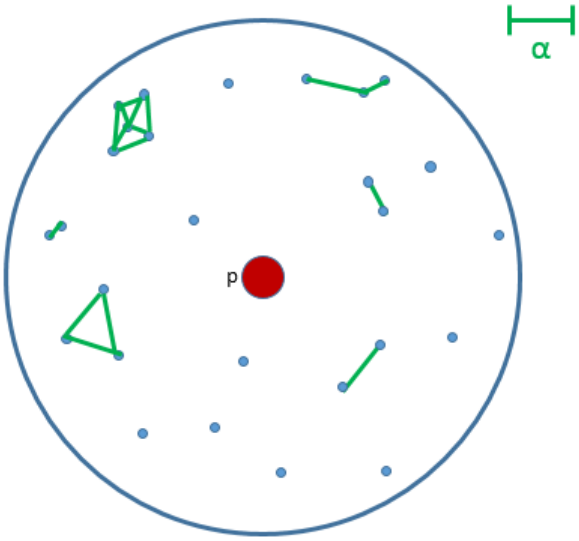


Figure 7.2.6: The Local Complex edges. Construct all edges between vertices at most α apart.

7.2.6 Merged simplex projections

We project the simplices obtained onto a unit circle[1] centered at p, applying Definition 6.1.1. The $\arctan\left(\frac{y_i}{x_i}\right)$ of the end points (x_i, y_i) of the edge are recorded as θ_1 and θ_2, with the former being the smaller of the two (see Figure 7.2.7). Note that the largest possible edge projection subtends π radians (in the case where the edge contains the point p), and therefore every edge projection is a minor arc on the unit circle. The projection and merging of edges is described in lines 311–354 of the appendix.

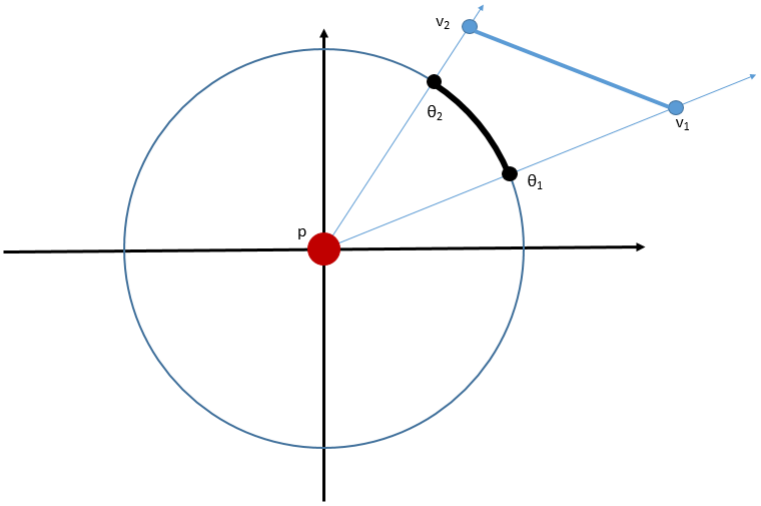


Figure 7.2.7: Projection of an edge, $\overline{v_1 v_2}$, onto an arc on the unit circle.

Recall that the range of the arc tangent is $-\pi$ to π. There are four cases for θ_1 and θ_2:

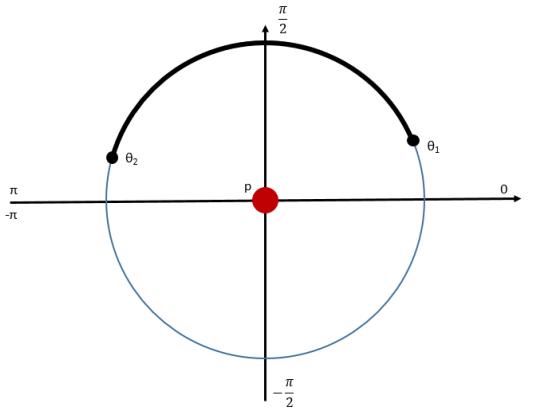
(a) both positive (Figure 7.2.8a),

(b) both negative (Figure 7.2.8b),

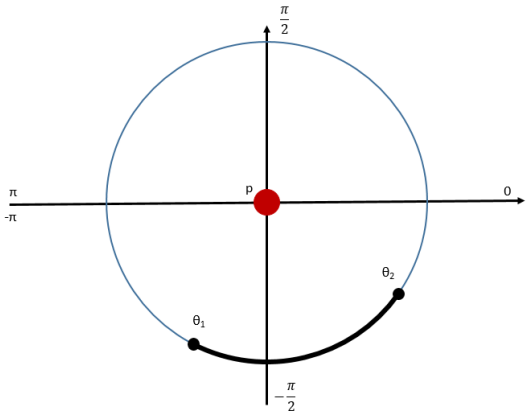
(c) $\theta_1 < 0$ and $\theta_2 > 0$ and $\theta_2 - \theta_1 \leq \pi$ (Figure 7.2.8c), and

[1] In principle, any circle will do and the unit circle is only required for concreteness, since, in this case, the lengths of the arcs yielded by the projections will be in radians.

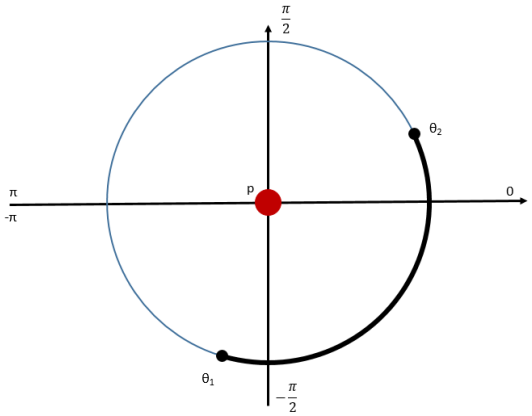
(d) $\theta_1 < 0$ and $\theta_2 > 0$ and $\theta_2 - \theta_1 > \pi$ (Figure 7.2.8d).



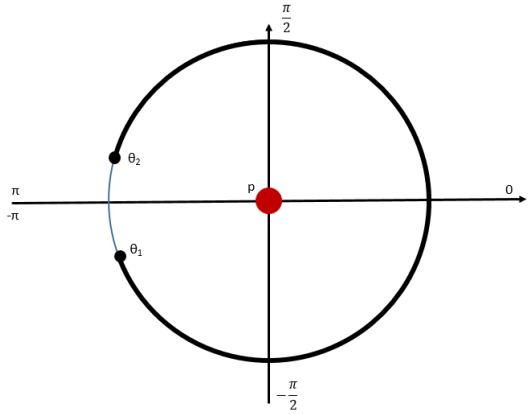
(a) Case one: both angles are positive.



(b) Case two: both angles are negative.



(c) Case three: one angle is positive, one angle is negative, and they form a minor arc.



(d) Case four: one angle is positive, one angle is negative, and they form a major arc.

Figure 7.2.8: The four possible cases for the configuration of two angles, θ_1 and θ_2, on the unit circle.

In all but the last case, we have that the arc $\widehat{\theta_1\theta_2}$ is a minor arc. In case (d), however, the arc $\widehat{\theta_1\theta_2}$ is major. We fix this by breaking this arc into two pieces: an arc $\widehat{\theta_2\pi}$ and an arc $\widehat{-\pi\theta_1}$ (Figure 7.2.9). These two arcs are both minor (cases (a) and (b), respectively). This process effectively transforms the arcs on the unit circle to line segments in the interval $[-\pi, \pi]$, which enables the merging procedure described next.

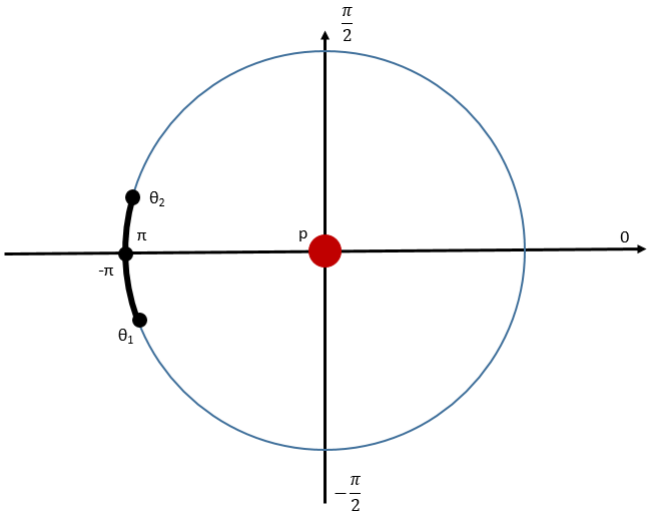


Figure 7.2.9: Case (d): replace the major arc with two minor arcs, one from θ_2 to π and the other from $-\pi$ to θ_1.

We sort our arcs by the smaller angle θ_1, to yield a list of arcs in order. Overlapping arcs are then merged to yield a list of disjoint arcs, as described in the method in lines 332–354 of the code in the appendix. Finally, we correct for any fix that we applied earlier for case (d), by combining any arcs of the form $\widehat{\phi\pi}$ and $\widehat{-\pi\psi}$. The arcs used for the combination are deleted and the combined arc is placed at the head (i.e., as the greatest element) of the merged disjoint list.

Example 7.2.1 (Merging the Arc Projections)**.** We discuss the process of merging the arc projections by computing the merged arcs for the collection of arcs in Figure 7.2.10. We begin with a list of arcs, where $\widehat{\theta_1\theta_2}$ is written (θ_1, θ_2), and order them by the smaller angle θ_1.

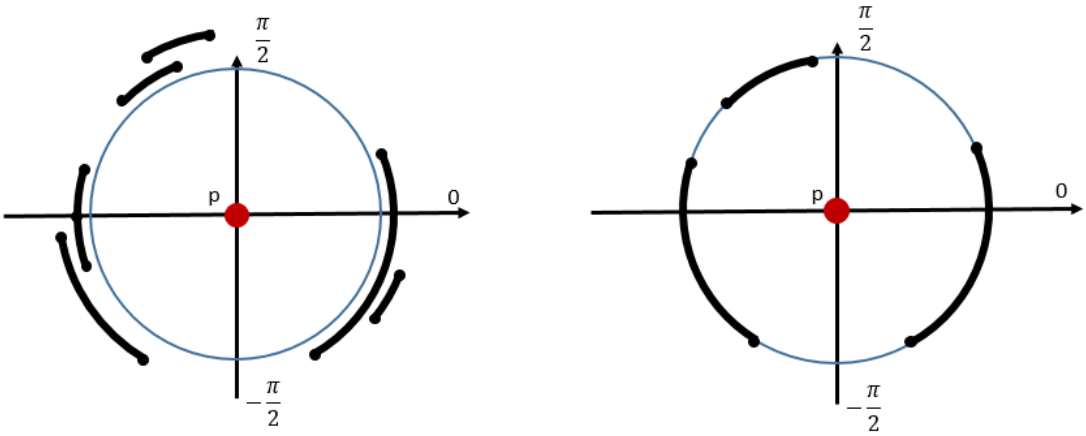


Figure 7.2.10: The collection of arc projections followed by the result of merging the arcs.

In the following, the angles are expressed in degrees instead of radians for readability:

$$L_0 = [(-180, -165), (-175, -105), (-75, 10), (-45, -35), (95, 120), (110, 135), (170, 180)].$$

We automatically include the first arc of L_0 in the merged list M_1 and create L_1 by

removing the first arc from L_0.

$$M_1 = \left[\, (-180, -165) \, \right],$$

$$L_1 = \left[\, (-175, -105) \, , (-75, 10) \, , (-45, -35) \, , (95, 120) \, , (110, 135) \, , (170, 180) \right].$$

In the following iteration, we take the first arc in the list L_1 and, if the θ_1-value of the first arc in L_1 is smaller than the θ_2-value of the last arc in M_1, we replace the θ_2-value of the last arc in M_1 with the maximum of the θ_2-values of the first arc of L_1 and the last arc of M_1 to get

$$M_2 = \left[\, (-180, -105) \, \right],$$

$$L_2 = \left[\, (-75, 10) \, , (-45, -35) \, , (95, 120) \, , (110, 135) \, , (170, 180) \right].$$

In the next iteration, since the θ_1-value of the first arc in L_2 is greater than the θ_2-value of the last arc in M_2, we include the first arc of L_2 in M_2 as a new entry:

$$M_3 = \left[(-180, -105) \, , \, (-75, 10) \, \right],$$

$$L_3 = \left[\, (-45, -35) \, , (95, 120) \, , (110, 135) \, , (170, 180) \right].$$

In the next iteration, since the θ_1-value of the first arc in L_3 is less than the θ_2-value of the last arc in M_3, we merge the first arc of L_3 with the last arc of M_3, replacing the θ_2-value of the last arc in M_3 with the maximum of the θ_2-values of the two merged arcs:

$$M_4 = \left[(-180, -105) \, , \, (-75, 10) \, \right],$$

$$L_4 = \left[\, (95, 120) \, , (110, 135) \, , (170, 180) \right].$$

Continuing this process, we find

$$M_5 = \left[(-180, -105) \, , (-75, 10) \, , \, (95, 120) \, \right],$$

$$L_5 = \left[\, (110, 135) \, , (170, 180) \right],$$

$$M_6 = \left[(-180,-105),(-75,10),(95,135)\right],$$

$$L_6 = \left[(170,180)\right],$$

$$M_7 = [(-180,-105),(-75,10),(95,135),(170,180)],$$

$$L_7 = [].$$

The last step involves merging any arcs of the form $(\theta_1, 180)$ and $(-180, \theta_2)$ (noting that, after the main procedure, we are left with at most one of each type), so our final list of merged arcs becomes

$$M_8 = [(170,-105),(-75,10),(95,135)].$$

7.2.7 Simplex projection-based coloring

Finally, we color each of the points in the original vertex set based on the number of components in the merged arcs, according to the following scheme (lines 356–372 in the appendix). The simplex projection-based coloring scheme is illustrated in Figure 7.2.11.

- Pink, if the merged components contains only the arc $\overbrace{-\pi\pi}$, indicating a point surrounded by points in its local neighborhood, i.e. a point in the interior of a two-dimensional region ($H_1(A) = \mathbb{Z}$ and $H_0(A) = \mathbb{Z}$, as in Figure 4.5.10).
- Blue, if the merged components contains a single arc, indicating a point on a boundary of a two-dimensional region ($H_0(A) = \mathbb{Z}$, as in Figure 4.5.2).
- Light green, if there are two components, indicating a point in a region homeomorphic to a line ($H_0(A) = \mathbb{Z}^2$, as in Figure 4.5.4).
- Red, if there are three components ($H_0(A) = \mathbb{Z}^3$, as in Figure 4.5.6).
- Light blue, if there are no components, indicating an isolated point ($H_0(A) = 0$).

- In every other case, the point is yellow (typically $H_0(A) > \mathbb{Z}^3$, as in Figure 4.5.8).

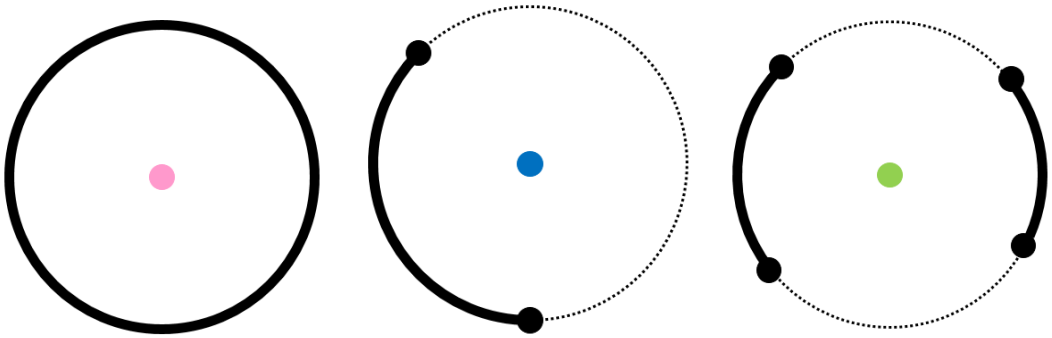
7.3 Stable points of α and β

In the previous section, we discussed an algorithm to construct the Acute Local Complex $L^*(X, p, \alpha, \beta)$. Now the question is: How can we choose α and β in such a way that the simplex projection-based colorings approximate the local homology of the underlying space meaningfully and not by some accident of parameters? We look to the data set to instruct our choice of α and β. In order to do so, we examine all choices of α and β and compare how much the simplex projection-based colorings change from one choice of α and β to the next, giving an algorithm for determining "interesting" α and β for any point cloud data set and, therefore, a scale-free characterization of the data set's shape.

We first simplify the discussion by examining the case where $\alpha = \beta$. Under this setting, the approach involves examining the simplex projection-based colorings of a range of α from zero, scanned up by a fixed increment, to some upper-bound. For each successive value of α, we determine the percentage of points whose color changed, compared to the previous value. The local and global minima of this sequence of percentage changes will indicate the values of α that are locally stable, i.e. those producing structures that do not change too much in the local neighborhood of a given α under consideration.

Definition 7.3.1. Let X be a point cloud data set of N points. Let α be the parameter in the Acute Local Complex $L^*(X, p, \alpha, \alpha)$ described in Definitions 5.2.1 and 5.2.7, and let us examine the sequence of simplex projection-based colorings of X as α varies from α_0 to α_n. Consider

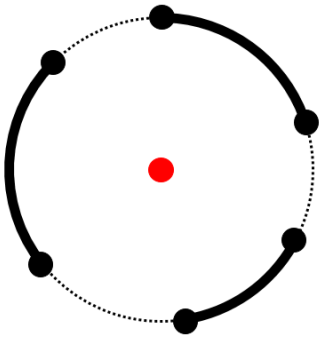
$$\min\left(\left\{N_{\alpha_1}/N, N_{\alpha_2}/N, ..., N_{\alpha_n}/N\right\}\right),$$



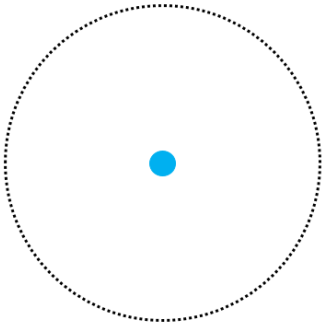
(a) Whole circle.

(b) One component.

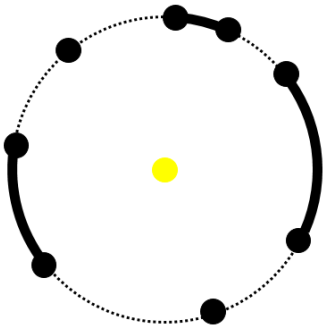
(c) Two components.



(d) Three components.



(e) No components.



(f) Many components.

Figure 7.2.11: Simplex projection-based coloring scheme.

where N_{α_k} is the number of points whose simplex projection-based color changes from the labeling at α_{k-1} to the labeling at α_k. Then the set of α_k such that N_{α_k}/N are local minima, represents the **stable α points**. For a dataset X, we denote this set as $\mathrm{STB}_\alpha(X)$, or, simply, STB_α, if the dataset is clear from the context.

Example 7.3.2 (Example: Multiscale Square)**.** This is a synthetic point cloud data set with rectangular "holes" at two different scales (4 at a small scale and 1 big one at a larger scale). The point cloud is uniformly randomly sampled from a square with vertices $(0,0), (0,10), (10,10)$ and $(10,0)$. Then points within the black regions in Figure 7.3.1 are removed using the methods in lines 27–42 of the code in the appendix. The resulting point cloud is pictured below in Figure 7.3.2.

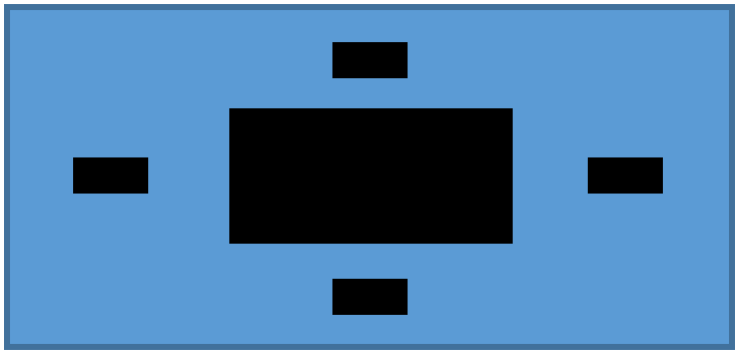


Figure 7.3.1: The Multiscale Square dataset is uniformly randomly sampled from the blue region.

Here, we vary α from 0 to 1.4 by an increment of 0.1 and compute the simplex projection-based colorings. To enable better visualization, we plot the logarithm of the percentage changes versus α in the graph pictured in Figure 7.3.3.

We note that, for this example, $\mathrm{STB}_\alpha = \{0.2, 0.9, 1.3\}$. Their simplex projection-based colorings are shown in Figures 7.3.5, 7.3.6 and 7.3.7.

In Figure 7.3.4 we show the colorings for all the values of α. The interesting ones are

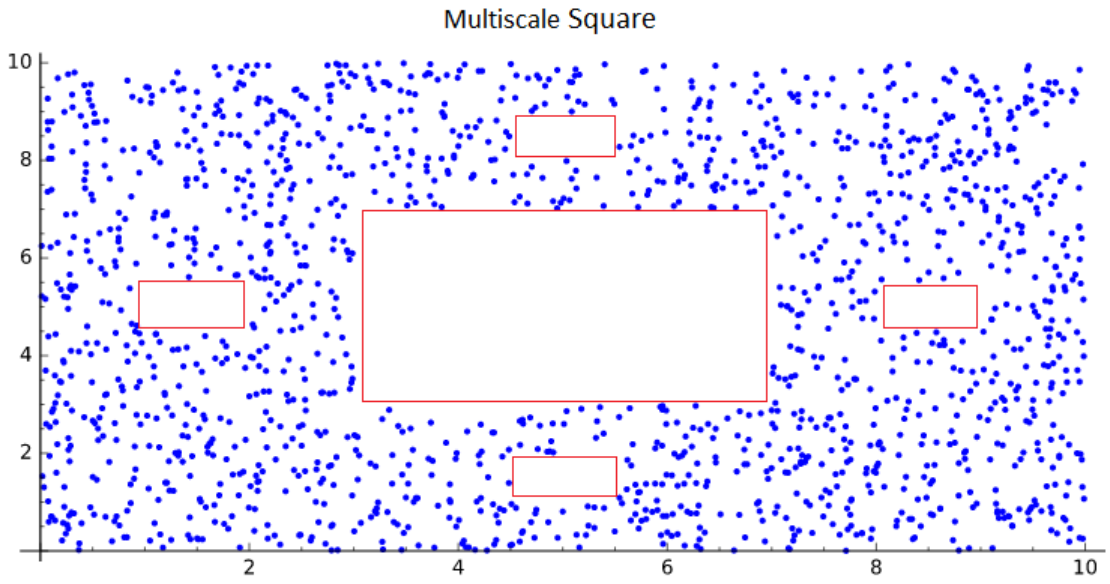


Figure 7.3.2: The point cloud data set Multiscale Square has structure at two interesting scales.

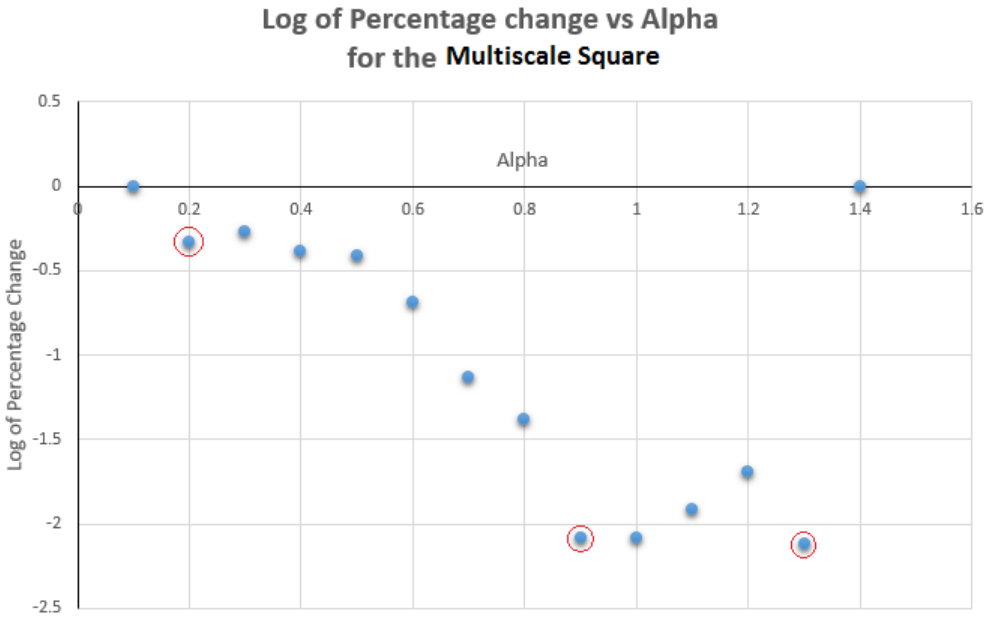


Figure 7.3.3: The logarithm of the percentage of points that changed their simplex projection-based coloring in successive α, versus α.

sub-figures 7.3.4b, 7.3.4f and 7.3.4h. In the context of the full sweep of α, we may better appreciate the usefulness of this algorithm, in that it picks out the scales at which the structures characterized by the Local Complexes are also intuitively interesting in a visual sense.

7.3.1 Two-dimensional analysis of stable α and β

We now describe the algorithm for identifying stable values of α and β when α and β are not necessarily equal. As described in the simplified case above, we examine the percentage of points whose simplex-projection based coloring changed from one choice of α to the next, while keeping β fixed. This results in a matrix (array of arrays), where one array corresponds, as in the simple case, to some fixed value of β, with α varying. Similarly, we examine the percentage of points whose coloring changes from one choice of β to the next, while keeping α fixed. We vary both α and β from 0 to some pre-selected fixed value.

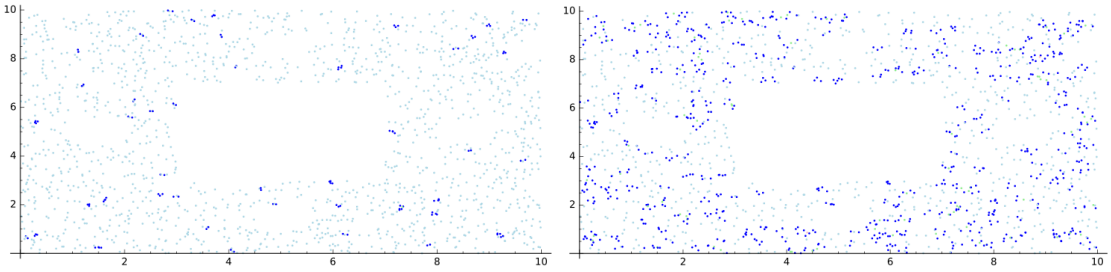
As in the simplified situation, here we are interested in values of α that serve as local minima, separately, for each array of the first matrix, corresponding to different fixed values of β, and, similarly, the set of β that serve as local minima, separately, for each array of the second matrix, corresponding to different fixed values of α.

Finally, any pair of α and β that characterize local minima in both variables are identified as special interesting values. In other words:

Definition 7.3.3. Let X be a point cloud data set of N points. Let β be the parameter in the Acute Local Complex $L^*\left(X, p, \alpha_i, \beta\right)$ described in Definitions 5.2.1 and 5.2.7, and let us examine the sequence of simplex projection-based colorings of X as β varies from β_0 to β_n. Consider

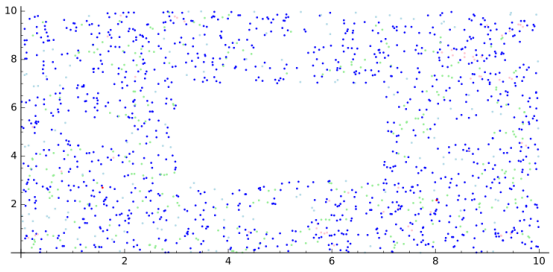
$$\min\left(\left\{N_{\beta_1}/N, N_{\beta_2}/N, ..., N_{\beta_n}/N\right\}_{\alpha_i}\right),$$

where N_{β_k} is the number of points whose simplex projection-based color changed

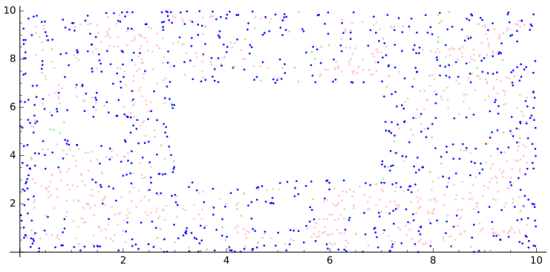


(a) $\alpha = 0.1$

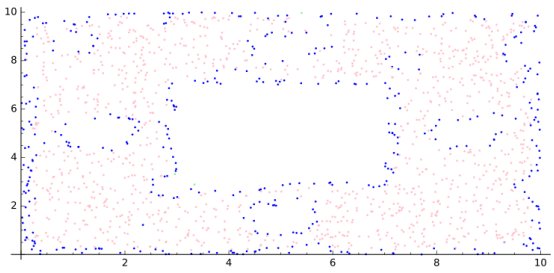
(b) $\alpha = 0.2$



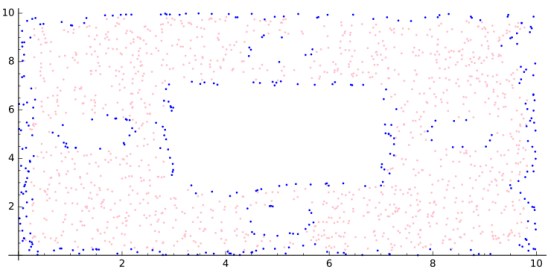
(c) $\alpha = 0.3$



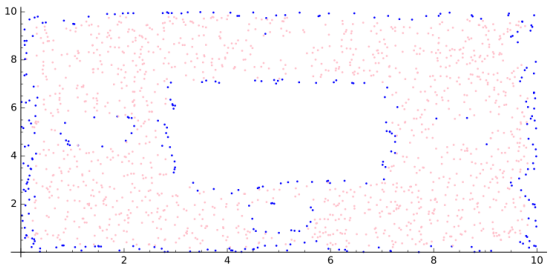
(d) $\alpha = 0.5$



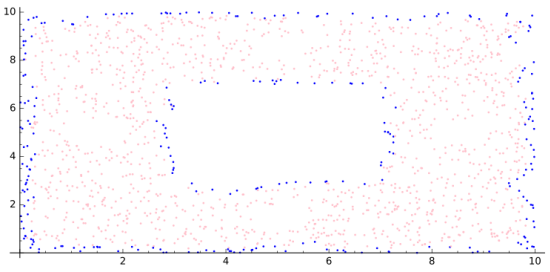
(e) $\alpha = 0.7$



(f) $\alpha = 0.9$



(g) $\alpha = 1.1$



(h) $\alpha = 1.3$

Figure 7.3.4: Simplex projection-based colorings of the Multiscale Square.

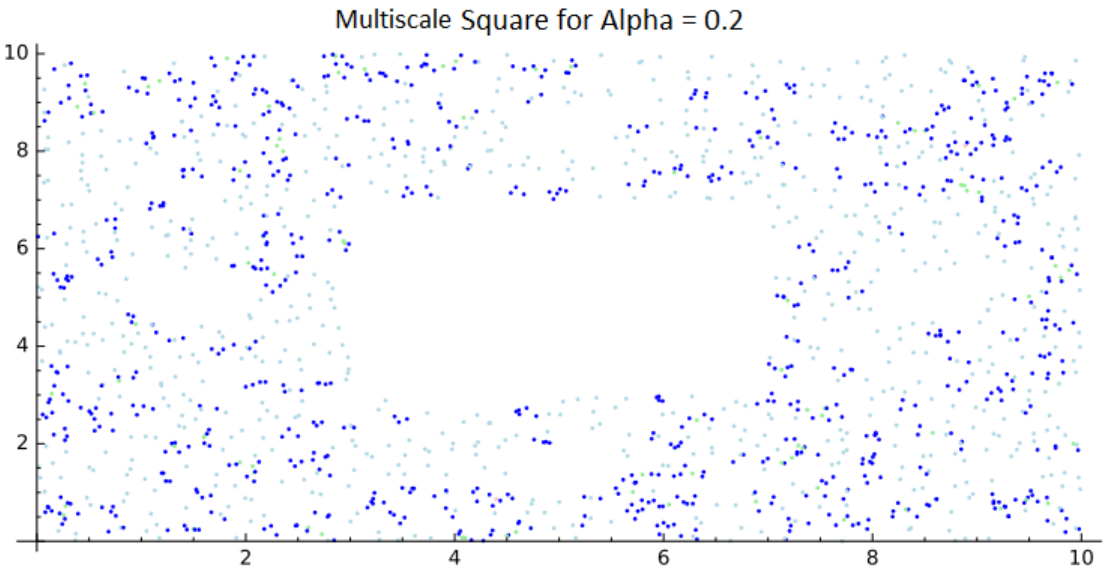


Figure 7.3.5: The homology-based coloring of the Multiscale Square at $\alpha = 0.2$.

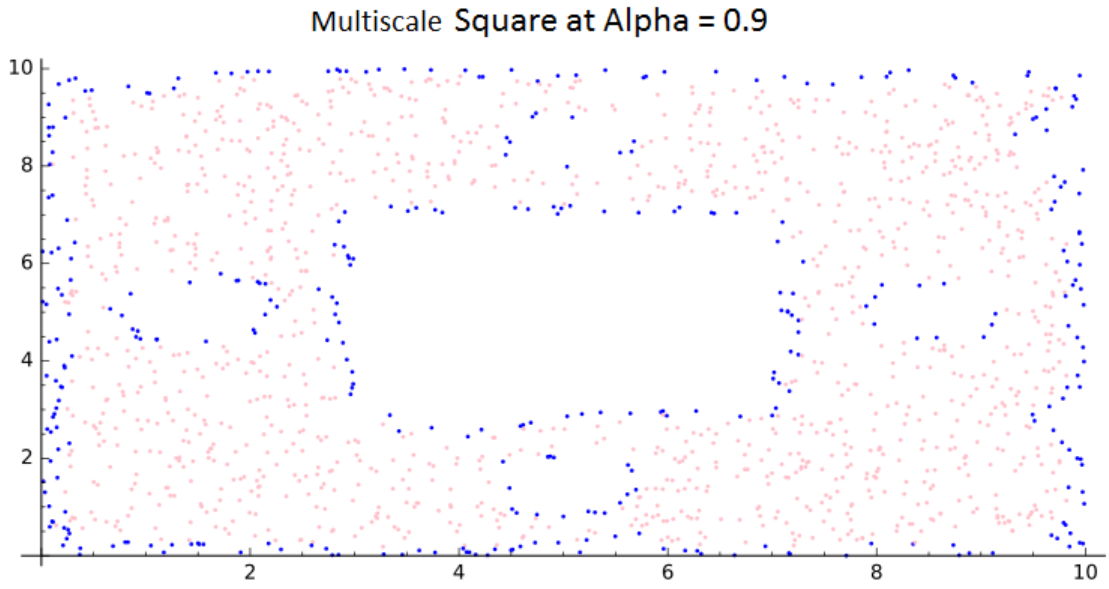


Figure 7.3.6: The homology-based coloring of the Multiscale Square at $\alpha = 0.9$.

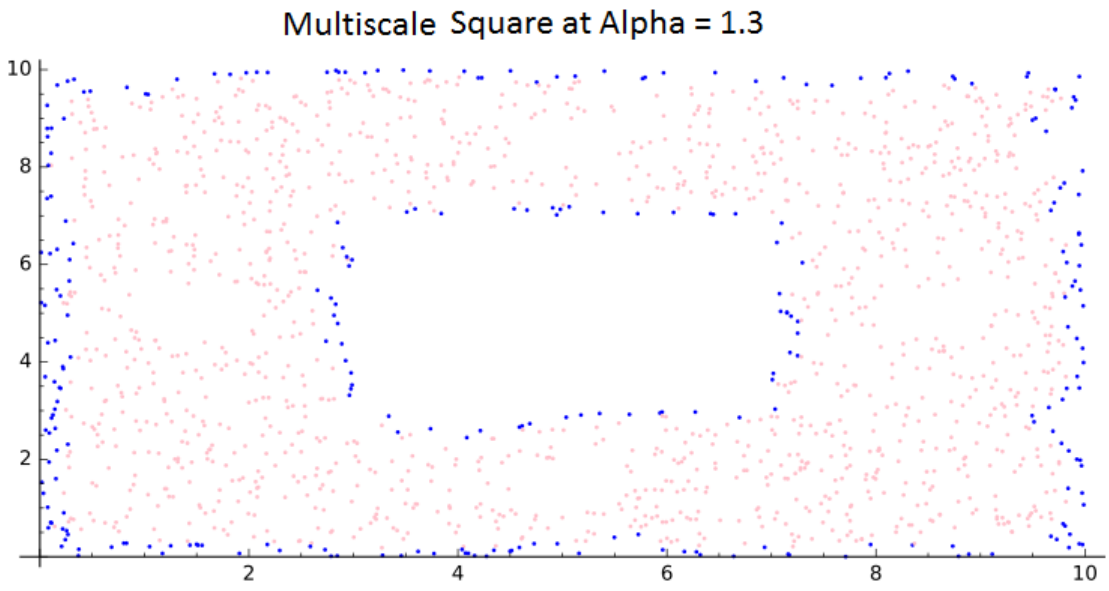


Figure 7.3.7: The homology-based coloring of the Multiscale Square at $\alpha = 1.3$.

from the labeling at β_{k-1} to the labeling at β_k, for a fixed value of α_i. Then, the set of β_k such that N_{β_k}/N are local minima, represents the ***β*-stable points** for a fixed α_i. For a dataset X, we denote this set as $\mathrm{STB}_\beta(X, \alpha_i)$, or, simply, $\mathrm{STB}_\beta(\alpha_i)$, if the dataset is clear from the context. We can similarly determine the ***α*-stable points** $\mathrm{STB}_\alpha\left(X, \beta_i\right)$.

The special interesting choices of both α and β, ***αβ*-stable points**, are then defined to be

$$\mathrm{STB}_{\alpha,\beta}(X) = \left\{\left(\alpha, \beta\right) | \alpha \text{ is stable for fixed } \beta \text{ and } \beta \text{ is stable for fixed } \alpha\right\}.$$

CHAPTER 8

RESULTS AND ANALYSIS

8.1 Introduction

We apply the algorithm described in Chapter 7, in particular, the Local Complex in Definition 5.2.1 with the constraint in Definition 5.2.7, to both artificial and real-world data sets. First, we consider three artificial sets: a capital letter A with a thick central bar, a capital letter A with a thin central bar, and a set of three balls joined by two thin lines.

8.2 A synthetic dataset: capital A

To construct the capital A with a thick central bar, we sample points uniformly randomly from a trapezoidal region with vertices $(0,0)$, $(1,4)$, $(3,4)$ and $(4,0)$, using the functions detailed in lines 24–47 of the Python code in the appendix. We then remove any points sampled from the two smaller black trapezoids pictured in Figure 8.2.1, using the function described in lines 33–39 of the code in the appendix. After determining the stable values of α and β by computing $\mathrm{STB}_{\alpha,\beta}(X)$ from Definition 7.3.3, we find that one of the combinations of $\alpha\beta$-stable points (where $\alpha = 0.21$ and $\beta = 0.21$) correctly identifies points which lie in two-dimensional regions and points which are on the boundary of two-dimensional regions (see Figure 8.2.2 in color).

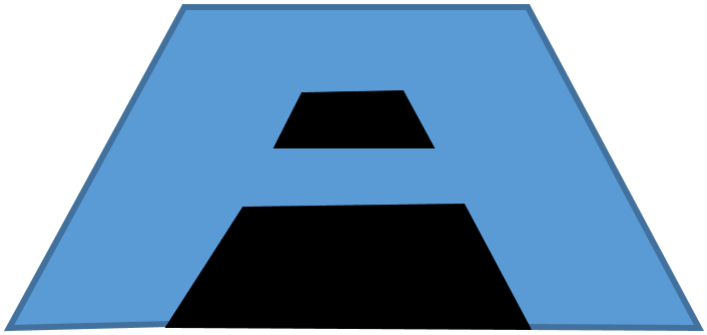


Figure 8.2.1: The capital A dataset with a thick central bar is randomly uniformly sampled from the blue region.

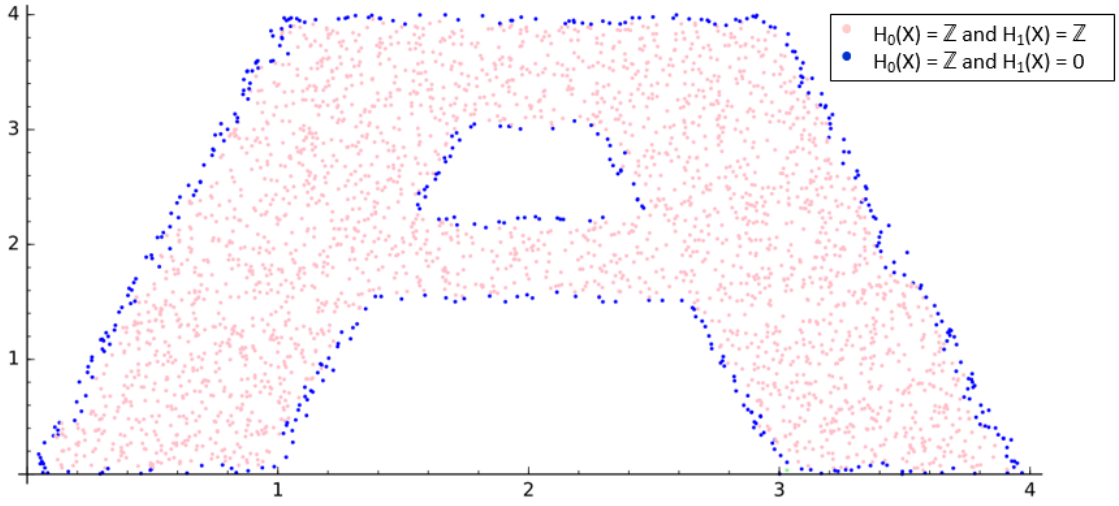


Figure 8.2.2: The resulting homology-based coloring of the thick A with $\alpha = 0.21$ and $\beta = 0.21$.

8.3 A synthetic dataset: capital A with thin central bar

The capital letter A with a thin central bar introduces a new challenge to correctly distinguish between points of the boundary of a two-dimensional region and points in a one-dimensional region. To construct the capital A with a thick central bar, we sample points uniformly randomly from a trapezoidal region with vertices $(0,0)$, $(1,4)$, $(3,4)$ and $(4,0)$. We then remove any points sampled from the two smaller black trapezoids pictured in Figure 8.3.1. As one can see in a color rendering of Figure 8.3.2, for one combination of $\alpha\beta$-stable points (where $\alpha = 0.21$ and $\beta = 0.21$), the central bar is distinguished from the boundary of the A.

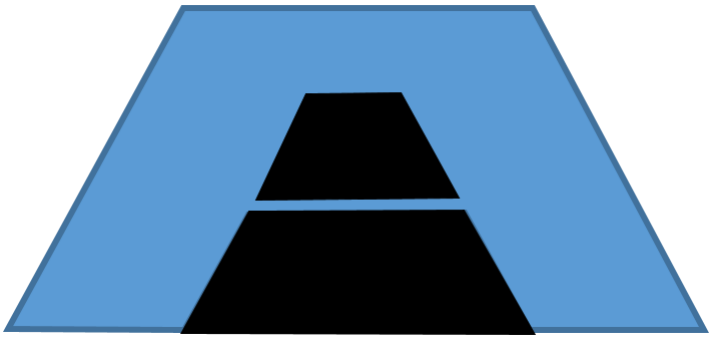


Figure 8.3.1: The capital A dataset with a thin central bar is randomly uniformly sampled from the blue region.

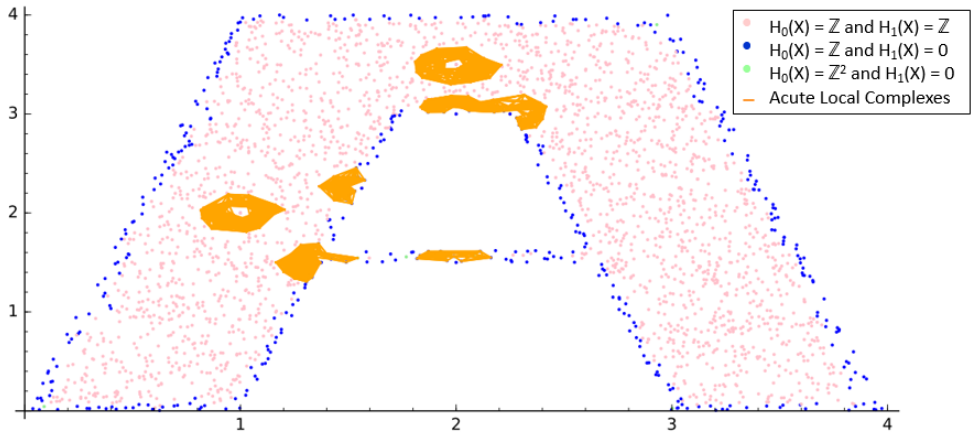


Figure 8.3.2: The resulting homology-based coloring of the thin A with $\alpha = 0.21$ and $\beta = 0.21$, as well as seven representative Acute Local Complexes.

8.4 A synthetic dataset: a flowchart

The flowchart dataset is constructed by uniformly randomly selecting points from the union of three octogons and two thin rectangles, illustrated in Figure 8.4.1. For the flowchart of three balls and two bars, the simplex projection-based homology coloring algorithm correctly identifies the centres of the balls as two-dimensional regions and can determine the points which form the boundary of the two-dimensional balls. It also establishes that the bars are formed of both boundary points and points which lie along a one-dimensional line (see Figure 8.4.2 in color). The values $\alpha = 0.9$ and $\beta = 0.7$ in Figure 8.4.2 are in the set of $\alpha\beta$-stable points for the flowchart.

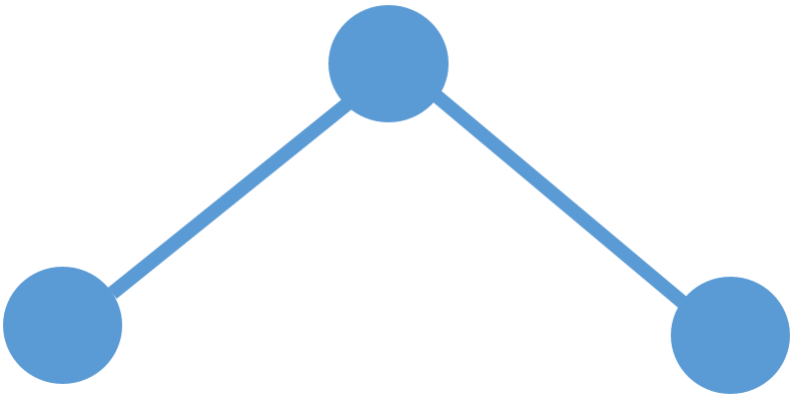


Figure 8.4.1: The flowchart dataset is randomly uniformly sampled from the blue region.

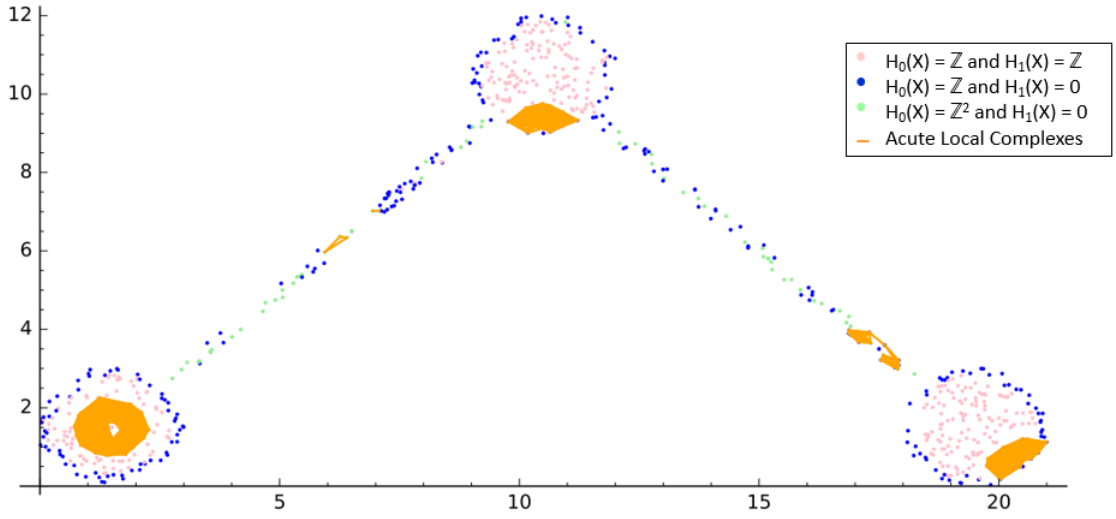


Figure 8.4.2: The resulting homology-based coloring of the flowchart with $\alpha = 0.9$ and $\beta = 0.7$, as well as five representative Acute Local Complexes.

8.5 A synthetic dataset: a uniformly random sample from a rectangle

In contrast, for a set of points randomly selected from a rectangular region, the stable α and β algorithm outputs choices of α and β that do not find any structure in the random data set. To construct the random dataset, we sample points uniformly randomly from a rectangular region with vertices $(0,0)$, $(0,4)$, $(4,4)$ and $(4,0)$ As one can see in a colored version of Figure 8.5.1, the randomly selected points are all identified as points on the interior of a two-dimensional region, except the boundary of the rectangle, which is colored blue.

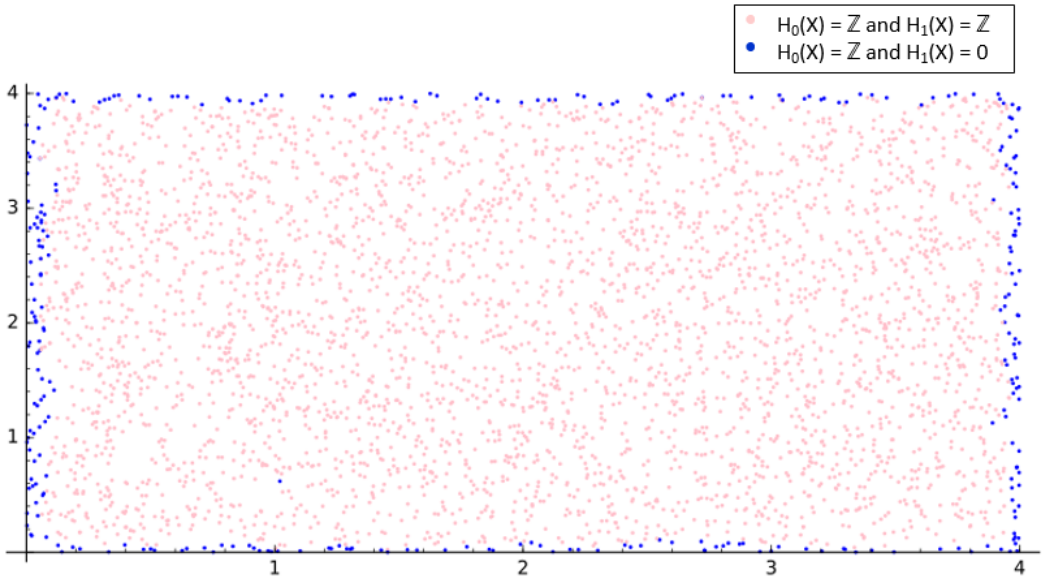


Figure 8.5.1: The resulting homology-based coloring of a random point set with $\alpha = 0.27$ and $\beta = 0.25$.

8.6 Real dataset: the SDSS galaxy data

We also consider a real-world application of the algorithm to the distribution of galaxies in the universe. For the data set, we analyze a thin slice of a Sloan Digital Sky Survey (SDSS) (described in Section 1.1.1). We project a thin three-dimensional slice of

the SDSS dataset, composed of 3357 galaxies, down to two dimensions and then apply the algorithm to determine the stable values of α and β.

A color rendering of Figure 8.6.1 depicts one choice of stable α and β. Note that the simplex projection-based coloring of the SDSS data varies greatly from the random dataset in Section 8.5 and Figure 8.5.1. The SDSS data is clearly not a random distribution of galaxies. The pink points indicate two-dimensional regions and are located in dense clusters. The blue points are boundaries of the two-dimensional regions and can perhaps outline voids. The green points lie on one-dimensional structures and may indicate filamentary regions.

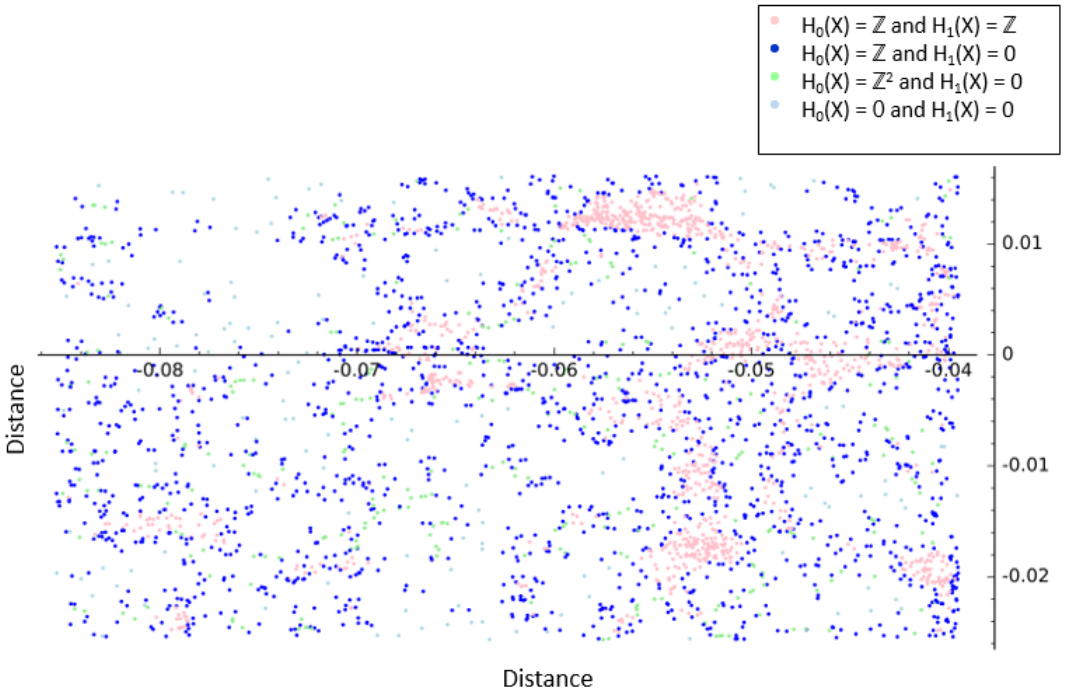


Figure 8.6.1: The resulting homology-based coloring of a slice of the SDSS galaxy survey with $\alpha = 0.002$ and $\beta = 0.0004$.

CHAPTER 9

FUTURE DIRECTIONS

Here we summarize the results and suggest areas of further research. The first two sections involve extending and strengthening the current results. The last section opens up an entirely new direction of research.

9.1 Higher dimensions

We wish to extend this work to higher dimensional analysis, primarily three-dimensional distributions. To do so, we could form the Acute Local Complex in three dimensions about each vertex p, and then project the 2-skeleton of the complex (triangles, edges and vertices) onto the unit sphere centered at p. In Example 9.1.1, we motivate how this technique could be applied in higher dimensions.

Example 9.1.1. Let U be the interior of a simplex, and $\overline{U}$ be the closure. Then the boundary $\mathrm{Bd}(U)$ is the closure set minus the interior. Take the following Figure 9.1.1 as an example.

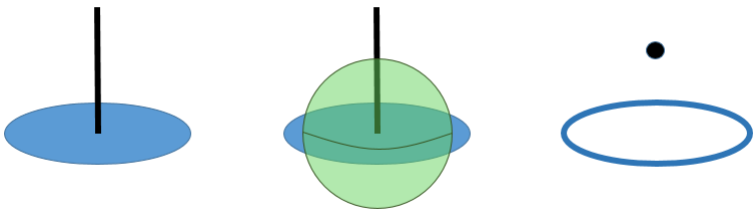


Figure 9.1.1: Extension of simplex arc projection to three dimensions. A set U on the left, and the result of the intersection of U with the surface of a sphere on the right.

If we intersect U with the surface of a sphere, the result is $\mathrm{Bd}(U)$. Then $H_0(\mathrm{Bd}(U)) = \mathbb{Z} \bigoplus \mathbb{Z}$ and $H_1(\mathrm{Bd}(U)) = \mathbb{Z}$. If this represented the galaxy data, it could

indicate that X is at the intersection of a two-dimensional wall and a one-dimensional filament.

9.2 Theory

Referring back to Figure 1.2.1, the local homology of the VR complex and the heuristic local homology of the simplex arc projections are related in a tenuous manner. In particular, we show two positive examples of their equivalency (see Examples 6.1.5 and 6.1.7), but we also show one counterexample (Example 6.1.3).

An important theoretical strengthening of this work involves the characterization of conditions under which exact equivalency holds.

9.3 Simplicial complex reconstruction

Throughout this thesis, we work under the assumption that the given point cloud is a sample of some true underlying space (which could be discrete, e.g. the galactic distribution, or continuous, e.g. the synthetic data sets used in this thesis, like the 'multiscale square').

A natural question then is: given the heuristic local dimension characterization of all points in the point cloud, recover a simplicial complex that fully represents the underlying space.

An important modern application of this line of work is the reconstruction of buildings, cars, street furniture and other urban artifacts from LIDAR point cloud data generated by autonomous self-driving cars.

BIBLIOGRAPHY

[BH11] Martin R Bridson and André Haefliger, *Metric spaces of non-positive curvature*, vol. 319, Springer Science & Business Media, 2011.

[BKP95] J Richard Bond, Lev Kofman, and Dmitri Pogosyan, *How filaments are woven into the cosmic web*, arXiv preprint astro-ph/9512141 (1995).

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva, *State of the art in surface reconstruction from point clouds*, EUROGRAPHICS star reports, vol. 1, 2014, pp. 161–185.

[CCDS06] Erik Carlsson, Gunnar Carlsson, and Vin De Silva, *An algebraic topological method for feature identification*, International Journal of Computational Geometry & Applications **16** (2006), no. 04, 291–314.

[DGGZ02] Tamal K Dey, Joachim Giesen, Samrat Goswami, and Wulue Zhao, *Shape dimension and approximation from samples*, Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2002, pp. 772–780.

[DSG06] Vin De Silva and Robert Ghrist, *Coordinate-free coverage in sensor networks with controlled boundaries via homology*, The International Journal of Robotics Research **25** (2006), no. 12, 1205–1222.

[DSG07] ———, *Coverage in sensor networks via persistent homology*, Algebraic & Geometric Topology **7** (2007), no. 1, 339–358.

[Hat02] Allen Hatcher, *Algebraic topology*, Cambridge University Press, 2002.

[Hun80] Thomas W Hungerford, *Algebra. reprint of the 1974 original*, Graduate Texts in Mathematics **73** (1980).

[ITA+08] Zeljko Ivezic, JA Tyson, B Abel, E Acosta, R Allsman, Y AlSayyad, SF Anderson, J Andrew, R Angel, G Angeli, et al., *LSST: from science drivers to reference design and anticipated data products*, arXiv preprint arXiv:0805.2366 (2008).

[KKM+11] Alexander Knebe, Steffen R Knollmann, Stuart I Muldrew, Frazer R Pearce, Miguel Angel Aragon-Calvo, Yago Ascasibar, Peter S Behroozi, Daniel Ceverino, Stephane Colombi, Juerg Diemand, et al., *Haloes gone mad: the halo-finder comparison project*, Monthly Notices of the Royal Astronomical Society **415** (2011), no. 3, 2293–2318.

[Ler45] Jean Leray, *Sur la forme des espaces topologiques et sur les points fixes des représentations*, J. Math. Pures Appl **24** (1945), no. 9, 95–167.

[Mun84] James R Munkres, *Elements of algebraic topology*, Addison-Wesley Menlo Park, 1984.

[Sou11] Thierry Sousbie, *The persistent cosmic web and its filamentary structure–i. theory and implementation*, Monthly Notices of the Royal Astronomical Society **414** (2011), no. 1, 350–383.

[XW14] Kelin Xia and Guo-Wei Wei, *Persistent homology analysis of protein structure, flexibility, and folding*, International Journal for Numerical Methods in Biomedical Engineering **30** (2014), no. 8, 814–844.

[ZC05] Afra Zomorodian and Gunnar Carlsson, *Computing persistent homology*, Discrete & Computational Geometry **33** (2005), no. 2, 249–274.

APPENDIX

```
# Imports and function definitions

import array
import csv
import matplotlib.pyplot as plt
import matplotlib
import numpy
import pprint
import random
import time
from datetime import datetime

from matplotlib.collections import LineCollection
from sage.plot.circle import Circle
from scipy.spatial import Delaunay
from sets import Set
from shapely.geometry import Point, Polygon

sage_server.MAX_OUTPUT_MESSAGES = 100000
sage_server.MAX_OUTPUT = 1000000

#
# General methods for polygons & points
# Used to create synthetic data sets
#

# Given a polygon poly and an integer n, return a set of n points
 ↪  contained within poly
def get_n_random_points_in_polygon(poly,n):
    i = n
    pts = []
    while i > 0:
```

```
32          pts.append(get_random_point_in_polygon(poly))
33          i-=1
34      return pts
35
36  # Given a set of points pts and a polygon poly, remove from pts all
     ↪ the points contained within poly and return the result
37  def remove_polygon_from_points(poly, pts):
38      good_pts = []
39      for p in pts:
40          if not poly.contains(Point(p)):
41              good_pts.append(p)
42      return good_pts
43
44  # Given a polygon poly, return a random point contained within poly
45  def get_random_point_in_polygon(poly):
46       (minx, miny, maxx, maxy) = poly.bounds
47       while True:
48           p = Point(random.uniform(minx, maxx), random.uniform(miny,
              ↪ maxy))
49           if poly.contains(p):
50               return (p.x, p.y)
51
52  # Given a radius, return a random point contained within a sphere of
     ↪ that radius
53  def generate_random_point_in_sphere(radius):
54      phi = uniform(0, 2*math.pi)
55      costheta = uniform(-1, 1)
56      u = uniform(0, 1)
57      theta = arccos(costheta)
58      r = radius * (u)^(1/3)
59      x = r * sin(theta) * cos(phi)
60      y = r * sin(theta) * sin(phi)
61      z = r * costheta
62      return (x, y, z)
63
```

```
64  # export points to csv
65  def export_points_csv(points, filename):
66      L = open(filename, "w", 0)
67      for p in points:
68          L.write(str(p[0]) + "," + str(p[1]) + "\n")
69      L.close()
70
71  #
72  # Compute the Vietoris-Rips Complex of a set of vertices, given a
     ↪ scale factor epsilon, based on Zomorodian's incremental algorithm
73  # First approach is to compute the entire VR complex and then find the
     ↪ homologies of the star of a vertex relative to the link
74  # and then color the points based on homology.  This is
     ↪ computationally expensive.
75  #
76
77  # Define Scale-based Edge Creation (Brute force O(n^2))
78  def FormEpsilonNearEdges(V,epsilon):
79      edges = {}
80      for v1 in V:
81          edges[v1]=Set()
82      for v1 in V:
83          for v2 in V:
84              if (v1 != v2):
85                  if sqrt((v2[1]-v1[1])^2+(v2[0]-v1[0])^2) < epsilon:
86                      edges[v1].add(v2)
87                      edges[v2].add(v1)
88      return edges
89
90  # Acute Local Complex: same as above; but, avoid edges crossing
     ↪ p-shell
91  def FormEpsilonNearEdges_NotCrossingPShell(p, V, epsilon):
92      edges = {}
93      for v1 in V:
94          edges[v1]=Set()
```

```
95      for v1 in V:
96          for v2 in V:
97              if (v1 != v2):
98                  dot_product = (v1[0] - p[0])*(v2[0] - p[0]) + (v1[1] -
                     ↪ p[1])*(v2[1] - p[1])
99                  if (dot_product > 0) and
                     ↪ (sqrt((v2[1]-v1[1])^2+(v2[0]-v1[0])^2) <=
                     ↪ epsilon):
100                     edges[v1].add(v2)
101                     edges[v2].add(v1)
102     return edges
103
104 # Define LowerNeighbors
105 def LowerNeighbors(E,V,u,OV):
106     N = Set()
107     for v in V[0:OV[u]]:
108         if (v in E[u]):
109             N.add(v)
110     return N
111
112 # Define AddCofaces
113 def AddCofaces(Edges, VertexSet, OrderedVertexSet, _V, tau, N, k):
114     _V.add(tau)
115     if len(tau)-1 >= k:
116         return
117     else:
118         for v in N:
119             sigma = tau | frozenset([v])
120             M = N & LowerNeighbors(Edges,VertexSet,v,OrderedVertexSet)
121             AddCofaces(Edges,VertexSet,OrderedVertexSet,_V,sigma,M,k)
122
123 # INCREMENTAL-VR Main Algorithm
124 def IncrementalVR(p, VertexSet, epsilon):
125     # For the Acute Local Complex, use
      ↪ FormEpsilonEdges_NotCrossingPShell
```

```
126     #Edges = FormEpsilonNearEdges(VertexSet, epsilon)
127     Edges = FormEpsilonNearEdges_NotCrossingPShell(p, VertexSet,
         ↪ epsilon)
128     # Totally order the vertex set
129     i = 0
130     OrderedVertexSet = {}
131     for v in VertexSet:
132         OrderedVertexSet[v] = i
133         i += 1
134     # Initialize VR complex to null
135     _V = Set()
136     for u in VertexSet:
137         N = LowerNeighbors(Edges,VertexSet,u,OrderedVertexSet)
138 
            ↪ AddCofaces(Edges,VertexSet,OrderedVertexSet,_V,frozenset([u]),N,3)
139     return _V
140 
141 #
142 # Local Shell Complex: find the VR complex of an annulus of points
     ↪ between beta - alpha and beta + alpha
143 # Alpha-beta shells & localized VR complexes
144 #
145 
146 def Bin(VertexSet, beta, alpha, GridLength):
147     # find bounding box
148     low_x = float('inf')
149     low_y = float('inf')
150     high_x = 0
151     high_y = 0
152     for p in VertexSet:
153         if (p[0] < low_x):
154             low_x = p[0]
155         if (p[0] > high_x):
156             high_x = p[0]
157         if (p[1] < low_y):
```

```
158             low_y = p[1]
159         if (p[1] > high_y):
160             high_y = p[1]
161 
162     # add buffer to bounding box
163     d = (beta + alpha)
164     low_x -= d
165     low_y -= d
166     high_x += d
167     high_y += d
168 #    print "here"
169 
170     #align low_x and low_y to a GridLength grid
171     low_x = math.floor((low_x) / GridLength) * GridLength
172     low_y = math.floor((low_y) / GridLength) * GridLength
173 
174     # initialize bins
175     binning = {}
176     y = low_y
177     while y <= high_y:
178         x = low_x
179         while x <= high_x:
180             binning[('%.2f' % x,'%.2f' % y)] = []
181             x += GridLength
182         y += GridLength
183 
184     # fill the bins
185     for p in VertexSet:
186         gridX = (math.floor((p[0]) / GridLength) * GridLength)
187         gridY = (math.floor((p[1]) / GridLength) * GridLength)
188         binning[('%.2f' % gridX,'%.2f' % gridY)].append(p)
189 
190     return binning
191 
```

```
# Given a binning and a point, return all "neighboring" points in grid cells that contain the beta+alpha disc
def AlphaBetaCandidatePoints(binning, p, beta, alpha, GridLength):
    d = (beta + alpha)
    # determine lower left grid cell
    g_ll = (math.floor((p[0] - d) / GridLength) * GridLength, math.floor((p[1] - d) / GridLength) * GridLength)
    # determine upper right grid cell
    g_ur = (math.floor((p[0] + d) / GridLength) * GridLength, math.floor((p[1] + d) / GridLength) * GridLength)

    # get all points in all containing grids
    neighbors = []
    y = g_ll[1]
    while y <= g_ur[1]:
        x = g_ll[0]
        while x <= g_ur[0]:
            neighbors.extend(binning[('%.2f' % x,'%.2f' % y)])
            x += GridLength
        y += GridLength
    return neighbors

def GetAlphaBetaShell(VertexSet, v, beta, alpha, gamma):
    Shell = []
    for p in VertexSet:
        dist = sqrt((p[1]-v[1])^2+(p[0]-v[0])^2)
        #For the Local Complex, use the first of the next two lines
        #For the Local Shell Complex and the Local Central Hole Complex, use the second of the next two lines
        if (dist <= beta/2 + alpha/2) and (dist > 0):
        # if ((dist <= beta/2 + alpha/2) and (dist >= max(gamma, beta/2 - alpha/2))):
            Shell.append(p)
    return Shell

```

```
222 def GetLinkVRHomologies(VertexSet, beta, alpha, gamma, GridLength, L,
    ↪ interesting_points):
223     homologies = {}
224     binning = Bin(VertexSet, beta, alpha, GridLength)
225     L.write("Number of points : " + str(len(VertexSet)) + "\n")
226     L.write("beta : " + str(beta) + "\n")
227     L.write("alpha : " + str(alpha) + "\n")
228     L.write("gamma : " + str(gamma) + "\n")
229     L.write("GridLength : " + str(GridLength) + "\n")
230     L.write("#bins : " + str(len(binning)) + "\n")
231     G = Graphics()
232     for p in VertexSet:
233         pShell = GetAlphaBetaShell(AlphaBetaCandidatePoints(binning,
            ↪ p, beta, alpha, GridLength), p, beta, alpha, gamma)
234         pVR = IncrementalVR(p, pShell, alpha)
235         L.write("Size of VR complex :  " + str(len(pVR)) + "\n")
236         pVR_SC = SimplicialComplex(pVR)
237         L.write("|\n")
238         if p in interesting_points:
239             G += point(p, color = "purple", size = 3)
240             for simplex in pVR:
241                 if (len(simplex) == 2):
242                     G += line(simplex, color="orange")
243     L.write("\n")
244     return (homologies,G)
245
246
247 #
248 # Homology coloring : Color the points based on their homology
249 #
250
251 # 0:Z is blue, 0:ZxZ is light green, 0:Z^3 is red, 0:Z^n is yellow and
    ↪ 1:Z is pink, default is light blue
252 def HomologyBasedColoring(homology):
253     # default color is light blue
```

```
254         _color = 'lightblue'
255         if (0 not in homology):
256             _color = 'lightblue'
257         elif ((str(homology[0]) == 'Z') and ((1 not in homology) or
             ↪  (str(homology[1]) == '0'))):
258             _color = 'blue'
259         elif ((str(homology[0]) == 'Z^2' or str(homology[0]) == 'Z x Z')
             ↪  and ((1 not in homology) or (str(homology[1]) == '0'))):
260             _color = 'lightgreen'
261         elif ((str(homology[0]) == 'Z^3' or str(homology[0]) == 'Z x Z x
             ↪  Z') and ((1 not in homology) or (str(homology[1]) == '0'))):
262             _color = 'red'
263         elif ((str(homology[0]) != 'Z' and str(homology[0]) != 'Z x Z' and
             ↪  str(homology[0]) != 'Z^2') and ((1 not in homology) or
             ↪  (str(homology[1]) == '0'))):
264             _color = 'yellow'
265         elif (str(homology[1] == 'Z')):
266             _color = 'pink'
267         return _color
268 
269 def ColorHomologies(VertexSet, homologies):
270     # define colorings
271     G = Graphics()
272     for p in VertexSet:
273         G += point(p, color = HomologyBasedColoring(homologies[p]),
             ↪  size = 2)
274     G.show()
275     return G
276 
277 # Plot alpha balls of varying radius around each point
278 def DrawAlphaBalls(VertexSet, alpha):
279     G = Graphics()
280     G += points(VertexSet)
281     for p in VertexSet:
282         G += circle((p), alpha/2, facecolor='pink', fill=True)
```

```
283     return G
284
285
286 def GetLinkVRHomologiesPlot(VertexSet, beta, alpha, gamma, GridLength,
    ↪ L, interesting_points):
287
288     Binning = Bin(VertexSet, beta, alpha, GridLength)
289     L.write("Number of points : " + str(len(VertexSet)) + "\n")
290     L.write("beta : " + str(beta) + "\n")
291     L.write("alpha : " + str(alpha) + "\n")
292     L.write("gamma : " + str(gamma) + "\n")
293     L.write("GridLength : " + str(GridLength) + "\n")
294     L.write("#bins : " + str(len(binning)) + "\n")
295     G = Graphics()
296     for p in VertexSet:
297         pShell = GetAlphaBetaShell(AlphaBetaCandidatePoints(binning,
            ↪ p, beta, alpha, GridLength), p, beta, alpha, gamma)
298         pVR = IncrementalVR(pShell, alpha)
299         L.write("Size of VR complex :  " + str(len(pVR)) + "\n")
300         pVR_SC = SimplicialComplex(pVR)
301         G += point(p, color =
            ↪ HomologyBasedColoring(pVR_SC._homology_()), size = 3)
302         L.write("|\n")
303         if p in interesting_points:
304             G += point(p, color = "purple", size = 2)
305             for simplex in pVR:
306                 if (len(simplex) == 2):
307                     G += line(simplex, color="orange")
308     L.write("\n\n\n")
309     return G
310
311 # Simplex projection: project edges and vertices in pShell onto circle
    ↪ centered at p and merge overlapping components,
312 # taking into account, (a) projections onto minor arcs only, and (b)
    ↪ the fact that arcs wrap around
```

```
313 def ComponentsBySimplexProjection(p, pShell, Edges):
314     components = set()
315     for v1 in pShell:
316         angle1 = atan2(float(v1[1] - p[1]), float(v1[0] - p[0]))
317         if v1 not in Edges or (len(Edges[v1]) == 0):
318             components.add((angle1, angle1))
319         elif v1 in Edges:
320             for v2 in Edges[v1]:
321                 angle2 = atan2(float(v2[1] - p[1]), float(v2[0] -
                     p[0]))
322                 if angle2 < angle1:
323                     temp = angle1
324                     angle1 = angle2
325                     angle2 = temp
326                 if (angle2 - angle1) > math.pi:
327                     components.add((angle2, math.pi))
328                     components.add((-math.pi, angle1))
329                 else:
330                     components.add((angle1, angle2))
331     sorted_components = sorted(components, key=lambda tup: tup[0])
332     merged = []
333     for higher in sorted_components:
334         if not merged:
335             merged.append(higher)
336         else:
337             lower = merged[-1]
338             if higher[0] <= lower[1]:
339                 upper_bound = max(lower[1],higher[1])
340                 merged[-1] = (lower[0], upper_bound)
341             else:
342                 merged.append(higher)
343     minus_pi_index = -1
344     plus_pi_index = -1
345     for i in range(len(merged)):
346         if merged[i][0] == -math.pi:
```

```
347             minus_pi_index = i
348         elif merged[i][1] == math.pi:
349             plus_pi_index = i
350     if (minus_pi_index != -1 and plus_pi_index != -1):
351 
352         merged.append((merged[plus_pi_index][0],merged[minus_pi_index][1]))
353         del merged[minus_pi_index]
354         del merged[plus_pi_index - 1]
355     return merged
356 
357 # circle is pink, 0 components is light blue, 1 component is blue, 2 components is green, 3 components is red, more is yellow
358 def SimplexArcProjectionsBasedColoring(components):
359     # default color is red
360     _color = 'red'
361     if (len(components) == 1 and components[0] == (-math.pi, math.pi)):
362         _color = 'pink'
363     elif (len(components) == 1):
364         _color = 'blue'
365     elif (len(components) == 2):
366         _color = 'lightgreen'
367     elif (len(components) == 3):
368         _color = 'red'
369     elif (len(components) == 0):
370         _color = 'lightblue'
371     else:
372         _color = 'yellow'
373     return _color
374 
375 def GetVRSimplexArcProjections(VertexSet, beta, alpha, gamma, GridLength, L, interesting_points):
376     binning = Bin(VertexSet, beta, alpha, GridLength)
377     L.write("Number of points : " + str(len(VertexSet)) + "\n")
```

```
378     L.write("alpha : " + str(alpha) + "\n")
379     L.write("gamma : " + str(gamma) + "\n")
380     L.write("GridLength : " + str(GridLength) + "\n")
381     L.write("#bins : " + str(len(binning)) + "\n")
382     G = Graphics()
383     ColoringsByVertex = {}
384     for p in VertexSet:
385         disc_candidate_points = AlphaBetaCandidatePoints(binning, p,
            ↪ beta, alpha, GridLength)
386         pShell = GetAlphaBetaShell(disc_candidate_points, p, beta,
            ↪ alpha, gamma)
387         # For the Acute Local Complex, use the second of the next two
            ↪ lines
388         #edges = FormEpsilonNearEdges(pShell, alpha)
389         edges = FormEpsilonNearEdges_NotCrossingPShell(p, pShell,
            ↪ alpha)
390         components = ComponentsBySimplexProjection(p, pShell, edges)
391         ColoringsByVertex[p] =
            ↪ SimplexArcProjectionsBasedColoring(components)
392         if p in interesting_points:
393             G += point(p, color = "purple", size = 3)
394             for p1 in pShell:
395                 if p1 in edges:
396                     for p2 in edges[p1]:
397                         G += line([p1, p2], color="orange", thickness
                            ↪ = 1)
398         G += point(p, color = ColoringsByVertex[p], size = 4)
399         L.write("|\n")
400     L.write("\n\n\n")
401     L.write("DONE GetVRSimplexArcProjections!\n")
402     return (G, ColoringsByVertex)
403
404 # Library of methods implementing the search for stable alpha and
    ↪ beta
405
```

```
406 # Compute the percentage of points whose colorings changed
407 def GetRelativeChange(OldColoringsMap, NewColoringsMap):
408     diff = 0
409     for k in OldColoringsMap.keys():
410         if OldColoringsMap[k] != NewColoringsMap[k]:
411             diff += 1
412     return float(diff)/float(len(OldColoringsMap))
413 
414 # Compute the matrix of relative changes in alpha, with fixed beta,
    ↪ and then changes in beta with fixed alpha
415 # Note that the alpha_start, alpha_end, alpha_change, beta_start,
    ↪ beta_end, beta_change variables are all scaled down by 100.  This
    ↪ may need to be adjusted depending on the data set.
416 def AlphaBetaGradient(VertexSet, interesting_points, GridLength,
    ↪ alpha_start, alpha_end, alpha_change, beta_start, beta_end,
    ↪ beta_change):
417     alpha_count = (alpha_end - alpha_start) / alpha_change
418     beta_count = (beta_end - beta_start) / beta_change
419     RelativeChanges_alpha = {}
420     for alpha_idx in range(0, alpha_count):
421         RelativeChanges_alpha[alpha_idx] = {}
422     for beta_idx in range(0, beta_count):
423         RelativeChanges_alpha[0][beta_idx] = 1.0
424 
425     RelativeChanges_beta = {}
426     for alpha_idx in range(0, alpha_count):
427         RelativeChanges_beta[alpha_idx] = {}
428     for alpha_idx in range(0, alpha_count):
429         RelativeChanges_beta[alpha_idx][0] = 1.0
430 
431     ColoringsByVertexByAlphaBeta = {}
432     for alpha_idx in range(0, alpha_count):
433         ColoringsByVertexByAlphaBeta[alpha_idx] = {}
434 
435     first_alpha = float(alpha_start)/float(10000)
```

```
436     for beta_idx in range(0, beta_count):
437         beta = float(beta_start + float(beta_idx) *
            ↪ beta_change)/float(10000)
438         _s = "alpha_" + str(first_alpha) + "_beta_" + str(beta)
439         L = open("log_" + _s + ".txt", "w", 0)
440         (G, ColoringsByVertex) = GetVRSimplexArcProjections(VertexSet,
            ↪ beta, first_alpha, 0, GridLength, L, interesting_points)
441         L.close()
442         G.save(_s + "_Combined_" + dataset + ".pdf")
443         ColoringsByVertexByAlphaBeta[0][beta_idx] = ColoringsByVertex
444 #         print("done beta : ", beta)
445 #         print("done alpha : ", first_alpha)
446
447     first_beta = float(beta_start)/float(10000)
448     for alpha_idx in range(0, alpha_count):
449         alpha = float(alpha_start + float(alpha_idx) *
            ↪ alpha_change)/float(10000)
450         _s = "alpha_" + str(alpha) + "_beta_" + str(first_beta)
451         L = open("log_" + _s + ".txt", "w", 0)
452         (G, ColoringsByVertex) = GetVRSimplexArcProjections(VertexSet,
            ↪ first_beta, alpha, 0, GridLength, L, interesting_points)
453         L.close()
454         G.save(_s + "_Combined_" + dataset + ".pdf")
455         ColoringsByVertexByAlphaBeta[alpha_idx][0] = ColoringsByVertex
456 #         print("done beta : ", first_beta)
457 #         print("done alpha : ", alpha)
458
459     for alpha_idx in range(1, alpha_count):
460         alpha = float(alpha_start + float(alpha_idx) *
            ↪ alpha_change)/float(10000)
461         beta_idx = 0
462         beta = float(beta_start + float(beta_idx) *
            ↪ beta_change)/float(10000)
463         _s = "alpha_" + str(alpha) + "_beta_" + str(beta)
464         L = open("log_" + _s + ".txt", "w", 0)
```

```
465             (G, ColoringsByVertex) = GetVRSimplexArcProjections(VertexSet,
                ↪  beta, alpha, 0, GridLength, L, interesting_points)
466             L.close()
467             G.save(_s + "_Combined_" + dataset + ".pdf")
468             ColoringsByVertexByAlphaBeta[alpha_idx][beta_idx] =
                ↪  ColoringsByVertex
469             RelativeChanges_alpha[alpha_idx][beta_idx] =
                ↪  GetRelativeChange(ColoringsByVertexByAlphaBeta[alpha_idx -
                ↪  1][beta_idx], ColoringsByVertex)
470 #           print("done beta : ", beta)
471 #           print("done alpha : ", alpha)
472
473         for beta_idx in range(1, beta_count):
474             beta = float(beta_start + float(beta_idx) *
                ↪  beta_change)/float(10000)
475             alpha_idx = 0
476             alpha = float(alpha_start + float(alpha_idx) *
                ↪  alpha_change)/float(10000)
477             _s = "alpha_" + str(alpha) + "_beta_" + str(beta)
478             L = open("log_" + _s + ".txt", "w", 0)
479             (G, ColoringsByVertex) = GetVRSimplexArcProjections(VertexSet,
                ↪  beta, alpha, 0, GridLength, L, interesting_points)
480             L.close()
481             G.save(_s + "_Combined_" + dataset + ".pdf")
482             ColoringsByVertexByAlphaBeta[alpha_idx][beta_idx] =
                ↪  ColoringsByVertex
483             RelativeChanges_beta[alpha_idx][beta_idx] =
                ↪  GetRelativeChange(ColoringsByVertexByAlphaBeta[alpha_idx][beta_idx
                ↪  - 1], ColoringsByVertex)
484 #           print("done beta : ", beta)
485 #           print("done alpha : ", alpha)
486
487         print("Finished edge cases!")
488
489         for alpha_idx in range(1, alpha_count):
```

```
490             alpha = float(alpha_start + float(alpha_idx) *
                ↪  alpha_change)/float(10000)
491             for beta_idx in range(1, beta_count):
492                 beta = float(beta_start + float(beta_idx) *
                    ↪  beta_change)/float(10000)
493                 _s = "alpha_" + str(alpha) + "_beta_" + str(beta)
494                 L = open("log_" + _s + ".txt", "w", 0)
495                 (G, ColoringsByVertex) =
                    ↪  GetVRSimplexArcProjections(VertexSet, beta, alpha, 0,
                    ↪  GridLength, L, interesting_points)
496                 L.close()
497                 G.save(_s + "_Combined_" + dataset + ".pdf")
498                 ColoringsByVertexByAlphaBeta[alpha_idx][beta_idx] =
                    ↪  ColoringsByVertex
499                 RelativeChanges_alpha[alpha_idx][beta_idx] =
                    ↪  GetRelativeChange(ColoringsByVertexByAlphaBeta[alpha_idx
                    ↪  - 1][beta_idx], ColoringsByVertex)
500                 RelativeChanges_beta[alpha_idx][beta_idx] =
                    ↪  GetRelativeChange(ColoringsByVertexByAlphaBeta[alpha_idx][beta_
                    ↪  - 1], ColoringsByVertex)
501 #               print("done beta : ", beta)
502 #               print("done alpha : ", alpha)
503
504
505     print("Alpha gradient")
506     for alpha_idx in range(0, alpha_count):
507         print(RelativeChanges_alpha[alpha_idx].values())
508
509     print("Alpha gradient - transpose")
510     for beta_idx in range(0, beta_count):
511         print([x[beta_idx] for x in RelativeChanges_alpha.values()])
512
513     print("Beta gradient")
514     for alpha_idx in range(0, alpha_count):
515         print(RelativeChanges_beta[alpha_idx].values())
```

```
516
517     return (RelativeChanges_alpha, RelativeChanges_beta)
518
519
520 # Given a 1d vector of relative changes, return stable indices,
    ↪ defined as the local minima
521 def GetStableIndices(RelativeChanges):
522     stable_idxs = []
523     i = 1
524     while true:
525         if i == len(RelativeChanges)-1:
526             break
527         if (RelativeChanges[i] <= RelativeChanges[i-1] and
            ↪ RelativeChanges[i] <= RelativeChanges[i+1]):
528             stable_idxs.append(i)
529         i += 1
530         if RelativeChanges[i] == min(RelativeChanges):
531             stable_idxs.append(i)
532     return stable_idxs
533
534 def GetStableAlphas(stable_idxs, alpha_start, alpha_change):
535     stable_alphas = []
536     for x in stable_idxs:
537         stable_alphas.append((x + 1)*(alpha_change) + alpha_start)
538     return stable_alphas
539
540 # Given two 2d matrices representing, respectively, relative changes
    ↪ in the alpha and beta dimension,
541 # compute and print the stable indices and their corresponding alpha
    ↪ and beta values.
542 def GetStableIndices_AlphaBeta(RelativeChanges_alpha,
    ↪ RelativeChanges_beta, label):
543     print("Stable indices in the alpha gradient")
544     stable_alpha_indices = Set([])
545     for beta_idx in range(0, len(RelativeChanges_beta)):
```

```
546             print("Main loop beta_idx = ", beta_idx)
547             alpha_grad = [x[beta_idx] for x in
                ↪ RelativeChanges_alpha.values()]
548             stable_indices = GetStableIndices(alpha_grad)
549             for alpha_idx in stable_indices:
550                 print("alpha_idx = ", alpha_idx)
551                 print("beta_idx = ", beta_idx)
552                 stable_alpha_indices.add(tuple([alpha_idx, beta_idx]))
553                 print("RelativeChanges_alpha[alpha_idx, beta_idx] = ",
                    ↪ RelativeChanges_alpha[alpha_idx][beta_idx])
554
555         print("Stable indices in the beta gradient")
556         stable_beta_indices = Set([])
557         for alpha_idx in range(0, len(RelativeChanges_alpha)):
558             beta_grad = RelativeChanges_beta[alpha_idx].values()
559             stable_indices = GetStableIndices(beta_grad)
560             for beta_idx in stable_indices:
561                 print("alpha_idx = ", alpha_idx)
562                 print("beta_idx = ", beta_idx)
563                 stable_beta_indices.add(tuple([alpha_idx, beta_idx]))
564                 print("RelativeChanges_beta[alpha_idx, beta_idx] = ",
                    ↪ RelativeChanges_beta[alpha_idx][beta_idx])
565
566         print("Stable indices in alpha beta")
567         print(stable_alpha_indices & stable_beta_indices)
568         with open(label + "_Interesting_alpha_beta.csv", "wb") as f:
569             writer = csv.writer(f)
570             writer.writerows(stable_alpha_indices & stable_beta_indices)
571
572 def Main(dataset, interesting_points, GridLength, alpha_start,
    ↪ alpha_end, alpha_change, beta_start, beta_end, beta_change):
573     label = dataset + '_a' + str(alpha_start)+ '_b' + str(beta_start)
574     data = list(csv.reader(open(dataset + ".csv", 'rU')))
575     VertexSet = map(lambda x: (float(x[0]), float(x[1])), data)
```

```
576     (RelativeChanges_alpha, RelativeChanges_beta) =
        ↪ AlphaBetaGradient(VertexSet, interesting_points, GridLength,
        ↪ alpha_start, alpha_end, alpha_change, beta_start, beta_end,
        ↪ beta_change)
577     GetStableIndices_AlphaBeta(RelativeChanges_alpha,
        ↪ RelativeChanges_beta, label)
```