# Parallel Construction of the Vietoris-Rips Complex in 3D

William J. Beksi and Nikolaos Papanikolopoulos

Department of Computer Science and Engineering, University of Minnesota

## The Goal

Speed up the construction of the Vietoris-Rips complex in three dimensions using multiple CPU cores and GPGPUs.

## Introduction

The Vietoris-Rips complex describes the topology of a point cloud. It is used in topological data analysis (TDA) to compute persistent homology, a method for finding topological features at different spatial resolutions. The construction of the Vietoris-Rips complex is the primary bottleneck in TDA due to the necessity of building the complex at different scales. Previous work has developed fast algorithms for creating complexes in arbitrary dimensions [1]. In this work, we focus on speeding up the construction of 3D complexes which have many relevant applications [2, 3, 4]. The key contribution of our work is:

- A novel set of algorithms and data structures to parallelize the creation of a 3D Vietoris-Rips complex.

## Mathematical Background

A topological space is composed of simplices:

- A $k$ dimensional simplex $\sigma$ is the convex hull of $k+1$ affinely independent points $v_0, \dots, v_k \in \mathbb{R}^n$.
- A face $\tau$ of $\sigma$ is the convex hull formed by the subset $\{v_0, \dots, v_k\}$ of the $k+1$ points.

Simplices can be joined to form simplicial complexes:

- A simplicial complex $K$ is a finite collection of simplices such that if $\sigma \in K$ and $\tau$ is a face of $\sigma$, then $\tau \in K$.
- Given a set of points $X = \{x_0, \dots, x_{m-1}\} \in \mathbb{R}^n$ in Euclidean $n$-space and a fixed radius $\epsilon$, the *Vietoris-Rips* complex of $X$ is an abstract simplicial complex whose $k$-simplices correspond to unordered $(k+1)$-tuples of points that are pairwise within $\epsilon$ distance of each other, Figure 1.

Persistent homology allows us to study homology (i.e. connected components, holes, and voids) at multiple scales [5]. It provides a framework to

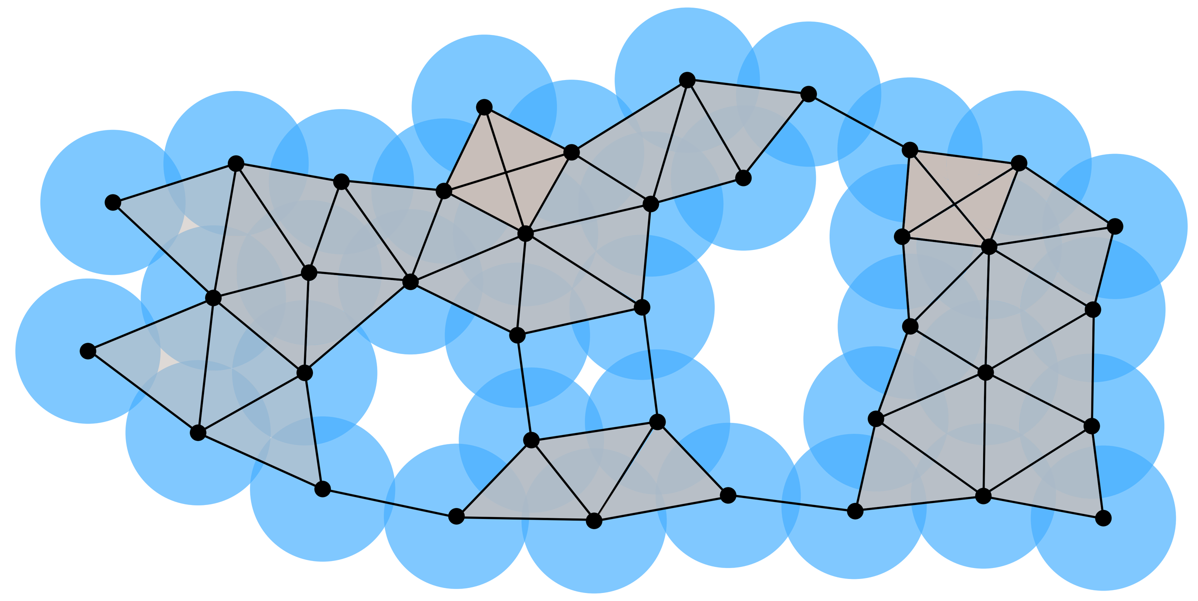

Figure 1: A Vietoris-Rips complex formed by joining together simplices of dimension 0 (points), 1 (edges), and 2 (triangles).

quantify the homological evolution of a parameterized sequence of topological spaces. Changes that occur as the scale parameter varies can be visualized through a barcode diagram, Figure 2. As the scale value changes, homology is *born* and *dies*. The pairing between births $b$ and deaths $d$ over varying spatial resolutions is known as a filtration. Topological features that exist for a long interval $d-b$ can be perceived as significant, while features with small intervals are indistinguishable from noise.

## Problem Statement

Let $X \in \mathbb{R}^3$ be a topological space. We construct the Vietoris-Rips complex, $K = \{\sigma \subset \{x_0, \dots, x_{m-1}\} \mid \text{dist}(x_i, x_j) \le \epsilon, \forall x_i \neq x_j \in \sigma\}$ where dist is the Euclidean metric, as follows. First, we perform a nearest neighbors search about each point. Next, we use the results of the nearest neighbors search to determine the edge and triangle relationships among the points. Finally, we construct a list of edges and triangles to be used for persistent homology computations.

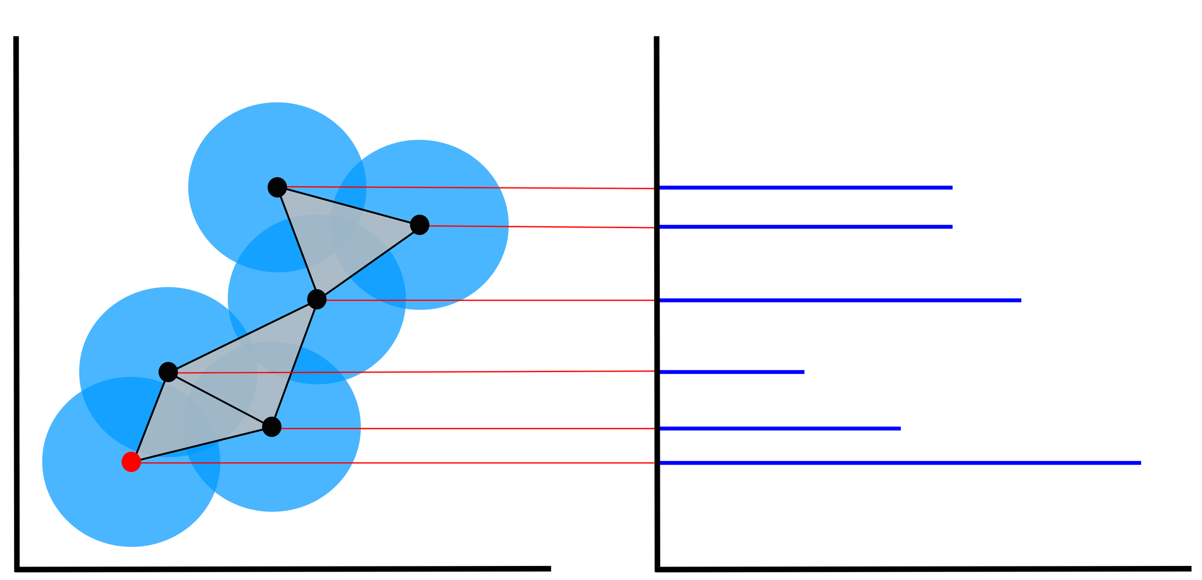

Figure 2: The evolution of a scale parameter defining the neighborhood radius about each point (left) and the corresponding barcode diagram (right).

## Nearest Neighbors Search

Given a 3D point cloud, we begin by sorting the points by their Cartesian coordinates. Next, we initialize a kd-tree and perform a parallel nearest neighbors search about each point. The nearest neighbors search is performed using nanoflann [6].

## Edge List Construction

After computing nearest neighbors we find the total number of edges by performing a parallel reduction on the vector of nearest neighbors. To find the offset of each edge a parallel prefix sum operation is done. An edge list is then created in parallel where an edge exists between two neighboring points.

## Triangle List Construction

Triangles are identified in parallel by finding all three cliques in the edge list. The total number of triangles and their offsets are computed using a parallel reduction and prefix sum, respectively. After the resulting operations a triangle list is constructed in parallel.

## Experimental Results

Figure 3 shows the performance of our parallel construction of the Vietoris-Rips complex in 3D. The results are averaged over ten independent runs. All experiments were performed using a single node with 24 CPU cores and an NVIDIA Tesla K40 GPU.

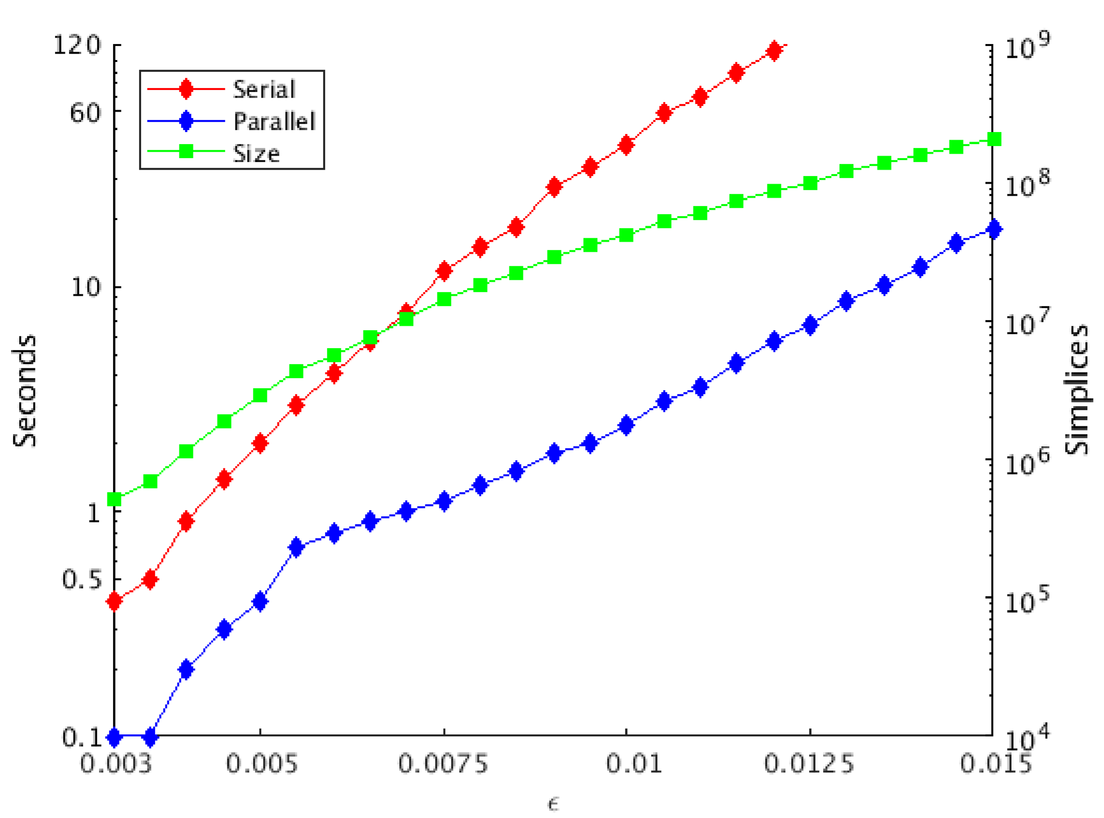


Figure 3: Construction time and complex size of a 3D Vietoris-Rips complex at scale $\epsilon$ for a point cloud consisting of 12,000 points.

## Conclusion

This work develops algorithms and data structures for the parallel assembly of a 3D Vietoris-Rips complex. Fast construction of the Vietoris-Rips complex, having many important applications, is a major hurdle in TDA due to its high computational costs. We've shown the increased performance of constructing a 3D Vietoris-Rips complex in parallel with a **13.1x** mean speedup over a serial implementation.

## References

[1] A. Zomorodian, "Fast Construction of the Vietoris-Rips Complex", *Computers & Graphics*, vol. 34, no. 3, pp. 263-271, 2010.

[2] W.J. Beksi and N. Papanikolopoulos, "3D Point Cloud Segmentation using Topological Persistence", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5046-5051, 2016.

[3] W.J. Beksi and N. Papanikolopoulos, "3D Region Segmentation using Topological Persistence", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1079-1084, 2016.

[4] W.J. Beksi and N. Papanikolopoulos, "Signature of Topologically Persistent Points for 3D Point Cloud Description", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3229-3234, 2018.

[5] H. Edelsbrunner, D. Letscher, A. Zomorodian, "Topological Persistence and Simplification", *Discrete and Computational Geometry*, vol. 28, no. 4, pp. 511-533, 2002.

[6] J.L. Blanco and P.K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees", 2014. [Online]. Available: https://github.com/jlblancoc/nanoflann

## Acknowledgements

The authors acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to these research results. This material is based upon work supported by the National Science Foundation through grants #CNS-1338042, #IIS-1427014, #CNS-1439728, #CNS-1531330 and #CNS-1544887.

University of Minnesota
Driven to Discover℠