Computational Geometry: Young Researchers Forum 2023

— book of abstracts —

This volume contains the abstracts of presentations given at "Computational Geometry: Young Researchers Forum" (CG:YRF), a satellite event of the 39th International Symposium on Computational Geometry, held in Dallas, Texas, USA, on June 12-15, 2023.

There were 21 abstracts submitted to CG:YRF. Of these, 19 were accepted with revisions after a limited refereeing process to ensure some minimal standards and to check for plausibility. One abstract was withdrawn during the revision process. The abstracts have been made public for the benefit of the community and should be considered preprints rather than formally reviewed papers. Thus, these works are expected to appear in conferences with formal proceedings and/or in journals. Copyrights of the works in this booklet are maintained by their respective authors. More information about the event and about previous and future editions is available online at

`http://www.computational-geometry.org`

CG:Young Researchers Forum — Programme

On the Fine-Grained Complexity of Small-Size Geometric Set Cover and Discrete k-Center for Small k

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, USA

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, USA

Yuancheng Yu
Department of Computer Science, University of Illinois at Urbana-Champaign, USA

Abstract

We prove a number of new conditional lower bounds on the time complexity of the discrete k-center problem and related (exact) geometric set cover problems when k or the size of the cover is small. Our lower bounds include the following (under certain popular hypotheses in fine-grained complexity such as the APSP Hypothesis and the Sparse Triangle Hypothesis, for an arbitrarily small $\delta > 0$):

- $\Omega(n^{3/2-\delta})$ for weighted size-3 set cover for axis-aligned unit squares in 2D;
- $\Omega(n^{4/3-\delta})$ for unweighted size-3 set cover for boxes in 3D;
- $\Omega(n^{4/3-\delta})$ for rectilinear discrete 3-center in 4D and unweighted size-3 set cover for unit hypercubes in 4D.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric set cover, discrete k-center, conditional lower bounds

Related Version A full version of the paper is available at https://arxiv.org/abs/2305.01892 and is also accepted to ICALP'23.

Funding *Timothy M. Chan*: Work supported by NSF Grant CCF-2224271.

1 Introduction

The rectilinear discrete k-center problem for small k. Various versions of the *k-center* problem have been extensively studied in the computational geometry literature. In this presentation, we consider specifically the *rectilinear discrete k-center* problem: given a set P of n points in $\mathbb{R}^d$ and a number k, we want to find k congruent hypercubes covering P, while minimizing the side length, with the extra constraint that the centers of the chosen hypercubes are from P.

The rectilinear discrete 2-center problem can be solved in $\widetilde{O}(n)$ time in any constant dimension d, by a straightforward application of orthogonal range searching, as reported in several papers [3, 4, 10]. The approach does not seem to work for the rectilinear discrete 3-center problem. Naively, rectilinear discrete 3-center can be reduced to n instances of (some version of) rectilinear discrete 2-center, and solved in $\widetilde{O}(n^2)$ time. However, no better results have been published. In contrast, the *continuous* version of the rectilinear 3-center problem, where the the centers are unrestricted, admits an $O(n \log n)$-time algorithm by Cabello et al. [5] in any constant dimension. We would like to address the following question:

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

Main Question. *Are there conditional lower bounds to show that the rectilinear discrete 3-center problem does not have near-linear-time algorithm (and is thus strictly harder than rectilinear discrete 2-center, or rectilinear continuous 3-center)?*

Similar questions may be asked about rectilinear discrete k-center for $k \geq 4$. As k gets larger compared to d, an upper bound of $n^{O(dk^{1-1/d})}$ is known for both the continuous and discrete k-center problem under the Euclidean and rectilinear metric [1, 8, 9]. Recently, in SoCG'22, Chitnis and Saurabh [7] (extending earlier work by Marx [11] in the $\mathbb{R}^2$ case) proved a nearly matching conditional lower bound for discrete k-center in $\mathbb{R}^d$, ruling out $n^{o(k^{1-1/d})}$-time algorithms under ETH. However, these bounds do not answer our questions concerning very small k's. On the other hand, in SODA'08, Cabello et al. [5] proved a conditional lower bound for rectilinear continuous 4-center in $\mathbb{R}^d$, ruling out $n^{o(\sqrt{d})}$-time algorithms under ETH, but this does not apply to the discrete version of the problem.

The geometric set cover problem with small size k. The decision version of the discrete k-center problem reduces to a *geometric set cover* problem: given a set P of n points and a set R of n objects, find the smallest subset of objects in R that cover all points of P. Geometric set cover has been extensively studied in the literature, particularly from the perspective of approximation algorithms, but here we are interested in *exact* algorithms for the case when the optimal size k is a small constant. For the application to rectilinear k-center, the objects are congruent hypercubes, or by rescaling, unit hypercubes, but other types of objects may be considered, such as arbitrary rectangles or boxes.

We can also consider the *weighted* version of the problem: here, given a set P of n points, a set R of n weighted objects, and a small constant k, we want to find a subset of k objects in R that cover all points of P, while minimizing the total weight of the chosen objects.

For rectangles in $\mathbb{R}^2$ or boxes in $\mathbb{R}^d$, size-2 geometric set cover (unweighted or weighted) can be solved in $\widetilde{O}(n)$ time, like discrete rectilinear 2-center [3, 4, 10], by orthogonal range searching. Analogs to our Main Question may be asked for size-3 geometric set cover for rectangles/boxes.

Surprisingly, the complexity of exact geometric set cover of small size k has not received as much attention, but very recently in SODA'23, Chan [6] initiated the study of similar questions for geometric independent set with small size k, for example, providing subquadratic algorithms and conditional lower bounds for size-4 independent set for boxes.

2 New results

We prove the first nontrivial conditional lower bounds on the time complexity of rectilinear discrete 3-center and related size-3 geometric set cover problems:[1]

- $\Omega(n^{3/2-\delta})$ for weighted size-3 set cover for unit squares in $\mathbb{R}^2$, assuming the APSP Hypothesis [12];
- $\Omega(n^{4/3-\delta})$ for rectilinear discrete 3-center in $\mathbb{R}^4$ and unweighted size-3 set cover for unit hypercubes in $\mathbb{R}^4$, assuming the Sparse Triangle Hypothesis [2];
- $\Omega(n^{4/3-\delta})$ for unweighted size-3 set cover for boxes in $\mathbb{R}^3$, assuming the Sparse Triangle Hypothesis [2].

[1] Here, δ denotes an arbitrarily small positive constant.

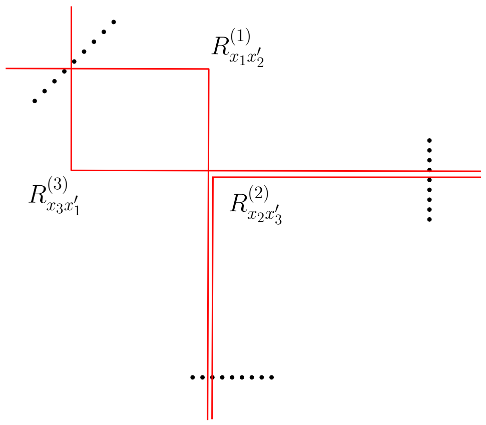


Figure 1 Reduction from the minimum-weight triangle problem (known to be equivalent to the APSP Hypothesis [13]) to weighted size-3 set cover for orthants in $\mathbb{R}^2$. Create $O(n)$ points of 3 types, and $O(m)$ orthants of 3 types $R^{(1)}_{x_1x'_2}$, $R^{(2)}_{x_2x'_3}$ and $R^{(3)}_{x_3x'_1}$ each representing an edge. The weight of each orthant is set to be the number of points it covers plus the weight of the edge it represents. A triangle $x_1x_2x_3$ would correspond to a set cover solution using the orthants $R^{(1)}_{x_1x_2}$, $R^{(2)}_{x_2x_3}$ and $R^{(3)}_{x_3x_1}$; on the other hand, if a triangle exists then the optimal size-3 set cover $R^{(1)}_{x_1x'_2}$, $R^{(2)}_{x_2x'_3}$ and $R^{(3)}_{x_3x'_1}$ must satisfy $x_1 = x'_1$, $x_2 = x'_2$ and $x_3 = x'_3$ because of the setting of weights.

The first bullet contrasts with some new subquadratic algorithms that we have obtained in $\mathbb{R}^2$ (which will not be covered in the presentation due to time constraints). The second bullet answers our Main Question, at least when the dimension is 4 or higher.

Techniques. Our conditional lower bounds for rectilinear discrete 3-center and the corresponding set cover problem for unit hypercubes are proved by reduction from unweighted or weighted triangle finding in graphs. It turns out there is a simple reduction that works in $\mathbb{R}^2$ if we exploit weights (see Fig. 1 for an example). However, lower bounds in the unweighted case (and thus the original rectilinear discrete 3-center problem) are trickier. We are able to find a simple reduction that works in $\mathbb{R}^6$ by hand, but reducing the dimension further to 4 is more challenging (because the number of choices is much larger), and we end up employing a *computer-assisted search*, interestingly. (The final construction is still simple, and so is easy to verify by hand.)

References

1 Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002. doi:10.1007/s00453-001-0110-y.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.

3 Sergei Bespamyatnikh and David G. Kirkpatrick. Rectilinear 2-center problems. In *Proc. 11th Canadian Conference on Computational Geometry (CCCG)*, 1999. URL: http://www.cccg.ca/proceedings/1999/fp55.pdf.

4 Sergei Bespamyatnikh and Michael Segal. Rectilinear static and dynamic discrete 2-center problems. In *Proc. 6th Workshop on Algorithms and Data Structures (WADS)*, pages 276–287, 1999. doi:10.1007/3-540-48447-7_28.

5 Sergio Cabello, Panos Giannopoulos, Christian Knauer, Dániel Marx, and Günter Rote. Geometric clustering: Fixed-parameter tractability and lower bounds with respect to the

dimension. *ACM Trans. Algorithms*, 7(4):43:1–43:27, 2011. Preliminary version in SODA'08. doi:10.1145/2000807.2000811.

6 Timothy M. Chan. Finding triangles and other small subgraphs in geometric intersection graphs. In *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023. To appear. URL: https://arxiv.org/abs/2211.05345.

7 Rajesh Chitnis and Nitin Saurabh. Tight lower bounds for approximate & exact k-center in $\mathbb{R}^d$. In *Proc. 38th International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:15, 2022. doi:10.4230/LIPIcs.SoCG.2022.28.

8 R. Z. Hwang, R. C. Chang, and Richard C. T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993. doi:10.1007/BF01228511.

9 R. Z. Hwang, Richard C. T. Lee, and R. C. Chang. The slab dividing approach to solve the Euclidean p-center problem. *Algorithmica*, 9(1):1–22, 1993. doi:10.1007/BF01185335.

10 Matthew J. Katz, Klara Kedem, and Michael Segal. Discrete rectilinear 2-center problems. *Comput. Geom.*, 15(4):203–214, 2000. doi:10.1016/S0925-7721(99)00052-8.

11 Dániel Marx. Efficient approximation schemes for geometric problems? In *Proc. 13th Annual European Symposium of Algorithms (ESA)*, pages 448–459, 2005. doi:10.1007/11561071_41.

12 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018. URL: https://people.csail.mit.edu/virgi/eccentri.pdf.

13 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. Preliminary version in FOCS'10. doi:10.1145/3186893.

Geodesic complexity of convex polyhedra

Ezra Miller
Department of Mathematics, Duke University, Durham, USA

Jiaxi Zhang
Department of Mathematics, Duke University, Durham, USA

Abstract

Geodesic complexity of the d-dimensional boundary S of a convex polytope of dimension $d+1$ is intimately related to the combinatorics of nonoverlapping unfolding of S into a Euclidean space $\mathbb{R}^d$ following Miller and Pak (2008). This combinatorics is based on facet sequences, which are lists of adjacent facets traversed by geodesics in S. Our main result bounds the geodesic complexity of S from above by the number of distinct maximal facet sequences traversed by shortest paths in S. For $d = 2$, results from the literature on nonoverlapping unfolding imply that this bound is polynomial in the number of facets. In arbitrary dimension d, a reinterpretation of conjectures by Miller and Pak (2008) leads to the conjecture that the geodesic complexity of S is polynomial in the number of facets. The theory and results developed here hold more generally for convex polyhedral complexes.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geodesic complexity, convex polyhedron, facet sequence, unfolding

1 Introduction

Recio-Mitter [11] introduced the concept of geodesic complexity, an extension of Farber's topological complexity [6]. The geodesic complexity $GC(X)$ of a metric space (X, d) measures the complexity of optimal motion planning on X, in the sense of decomposing the product $X \times X$ as a union $X \times X = \bigcup_i E_i$ such that on each subset E_i there is a map that assigns to each pair of endpoints a shortest path that varies continuously with the endpoints.

No known upper bounds exist for the geodesic complexity of a convex polyhedral boundary in general. This paper establishes the first such upper bound by making a heretofore unrealized connection between geodesic complexity and polyhedral nonoverlapping unfolding. Our approach enables theorems and conjectures concerning polyhedral nonoverlapping unfolding to translate well into bounds for geodesic complexity. Based on [9, Conjecture 9.3], we conjecture our upper bound to be polynomial; we prove it in dimension $d = 2$.

1.1 Geodesic complexity

▶ **Definition 1.1.** For a metric space (X, d), a path $\gamma \subseteq X$ with endpoints a and b is a *geodesic* if it locally minimizes length. A geodesic is a *shortest path* if its length equals the distance between its endpoint.

▶ **Definition 1.2** ([11, Definition 1.7])**.** The *geodesic complexity* $GC(X)$ of a metric space $(X,\ \mathrm{d})$ is the smallest k for which there exists a partition $X \times X = \bigcup_{i=0}^{k} E_i$ into pairwise disjoint locally compact sets each of which has a local continuous section $s_i : E_i \to GX$ of π, where GX is the shortest path space of X equipped with the compact-open topology by considering paths as continuous maps $\gamma : [0, 1] \to X$.

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

1.2 Polytope boundaries and convex polyhedral complexes

This extended abstract focuses on the boundary $S = \partial P$ of a convex polytope P of dimension $d+1$ in $\mathbb{R}^{d+1}$. But the proofs hold verbatim for convex polyhedral complexes in general.

1.3 Main results

The main result of the paper, Theorem 1.3, establishes a combinatorial upper bound on the geodesic complexity of a convex polyhedral boundary S in terms of the number of maximal *facet sequences* (Definitions 2.5, 2.8) of shortest paths in S. Such a sequence $\phi = (F_1, \ldots, F_\ell)$ is the ordered list of facets *visited* by a shortest path (Corollary 2.4).

▶ **Theorem 1.3.** *$GC(S) \leq \frac{1}{2}|\Phi_S| - 1$, where Φ_S is the set of maximal facet sequences in S.*

▶ **Remark 1.4.** We expect our bound to approximate the actual value of $GC(S)$. Our reasoning: almost all endpoint pairs are connected by a unique shortest path [9, Theorem 2.9]. So for any pair with nonunique shortest paths, there is a neighborhood of pairs whose facet sequences do not vary continuously through the given pair. This means maximal facet sequences cannot be easily patched together to yield a section of the endpoint map.

▶ **Example 1.5.** Let S be the boundary of a 3-cube. Theorem 1.3 yields that $GC(S) \leq 35$ as S has 72 maximal facet sequences. Indeed, count as follows. Any shortest path in S starts in some facet, by symmetry say the Top facet of S. Every maximal facet sequence that starts with Top ends with Bottom, because Top–Front–Left–Bottom, Top–Front–Right–Bottom, Top–Front–Bottom are all realized by shortest paths, but Top–Left–Front–Right is not: Top–Right shortcuts it. The count of 72 is factored as $6 \times 4 \times 3$, where $6 = \#$initial facets, $4 = \#$second facets, $3 = \#$ways to complete the sequence once the second facet is fixed.

▶ **Theorem 1.6.** *If $d = 2$, then $GC(S)$ is polynomial in the number of facets of S.*

2 Definitions and other background

▶ **Definition 2.1.** As S has dimension d, a face of dimension $d-1$ is a *ridge.* A point x is *warped* if x lies in the union S_{d-2} of all faces of dimension at most $d-2$; otherwise, x is *flat.*

▶ **Proposition 2.2.** *If γ is a geodesic in S, then γ has no warped points in its relative interior unless both endpoints are warped and lie in a single face.*

2.1 Facet sequences

▶ **Definition 2.3.** A geodesic γ of positive length in S *visits* a facet F if the relative interior of γ has a point in F.

▶ **Proposition 2.4** (cf. [9, Corollary 1.4])**.** *Fix a compact geodesic $\gamma \subseteq S$ not contained in a single face. Then γ visits (in order) a well defined sequence of facets.*

▶ **Definition 2.5.** Fix a compact geodesic $\gamma \subseteq S$. If γ is not contained in a single facet then it has *facet sequence* $\phi_\gamma = (F_1, \ldots, F_\ell)$, where $F_1, \ldots, F_\ell$ are, in order, all of the facets that γ visits. If γ is contained in a facet F then $\phi_\gamma = (F)$ is a *facet sequence* for γ.

▶ **Definition 2.6** ([9, Definition 1.5])**.** Suppose two facets F and F' share a ridge $R = F \cap F'$. For each facet F of S, let T_F be the affine span of F in $\mathbb{R}^{d+1}$. The *folding map* $f_{F,F'} : T_F \to T_{F'}$ is the isometry that identifies the copy of R in T_F with the one in $T_{F'}$ in such a way that the image of F does not intersect the interior of F'.

▶ **Definition 2.7** ([9, Definition 1.6])**.** Given a facet sequence $\phi = (F_1, \ldots, F_\ell)$ such that F_i shares a (unique) ridge with F_{i+1} whenever $1 \le i < \ell$, write

$$f_\phi^{-1} = f_{F_1,F_2}^{-1} \circ \cdots \circ f_{F_{\ell-1},F_\ell}^{-1}$$

for the *unfolding of* T_{F_ℓ} *onto* T_{F_1}. Setting $\phi_i = (F_1, \ldots, F_i)$, the *sequential unfolding* of a subset $\Gamma \subseteq F_1 \cup \cdots \cup F_\ell$ *along* ϕ is the set

$$\Upsilon_\phi(\Gamma) = (\Gamma \cap F_1) \cup f_{\phi_2}^{-1}(\Gamma \cap F_2) \cup \cdots \cup f_{\phi_\ell}^{-1}(\Gamma \cap F_\ell) \subset T_{F_1}.$$

2.2 Maximal facet sequences and type

▶ **Definition 2.8.** A facet sequence ϕ is *maximal* if it is not a contiguous proper subsequence of any longer facet sequence. A shortest path γ has *type* ϕ if some facet sequence ϕ_γ of γ is a contiguous subsequence of the maximal facet sequence ϕ.

▶ **Proposition 2.9.** *Given a facet sequence ϕ and two endpoints x and y, there is at most one shortest path between them with type ϕ.*

3 Sketch of the proof of Theorem 1.3

Sketch of proof. Define $E_\phi \subseteq S \times S$ for $\phi \in \Phi_S$ to be $E_\phi = \{\pi(\gamma) \mid \gamma \text{ has type } \phi\}$.

▶ **Claim 3.1.** *The endpoint map $\pi : GS \to S \times S$ has local sections $s_\phi : E_\phi \to GS$ for $\phi \in \Phi_S$.*

By Proposition 2.9, every pair $(a, b) \in E_\phi$ is joined by a unique shortest path of type ϕ. Let s_ϕ map (a, b) to this path. Continuity is trivial after sequentially unfolding along ϕ.

▶ **Claim 3.2.** *Fix $\phi \in \Phi_S$ and the reverse facet sequence ϕ'. Any two continuous sections $s_\phi : E_\phi \to GS$ and $s_{\phi'} : E_{\phi'} \to GS$ glue to a continuous section $s_\phi \cup s_{\phi'} : E_\phi \cup E_{\phi'} \to GS$.*

Conclusion. Thus $E_\phi \cup E_{\phi'}$ is a domain of continuity for $\phi \in \Phi_S$. ◀

References

1 Pankaj K Agarwal, Boris Aronov, Joseph O'Rourke, and Catherine A Schevon. Star Unfolding of a Polytope with Applications. *SIAM Journal on Computing*, 26(6):1689–1713, 1997.

2 Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A Course in Metric Geometry*, volume 33. American Mathematical Society, 2001.

3 Jindong Chen and Yijie Han. Shortest Paths on a Polyhedron, Part I: Computing Shortest Paths. *International Journal of Computational Geometry & Applications*, 6(02):127–144, 1996.

4 Donald M Davis. Geodesics in the Configuration Spaces of Two Points in $\mathbb{R}^n$. *Tbilisi Mathematical Journal*, 14(1):149–162, 2021.

5 Donald M Davis and David Recio-Mitter. The Geodesic Complexity of N-dimensional Klein Bottles. *The New York Journal of Mathematics*, 27:296–318, 2021.

6 Michael Farber. Topological Complexity of Motion Planning. *Discrete & Computational Geometry*, 29(2):211–221, 2003.

7 Stephan Mescher and Maximilian Stegemeyer. Geodesic Complexity of Homogeneous Riemannian Manifolds. *arXiv preprint arXiv:2105.09215*, 2021.

8 Stephan Mescher and Maximilian Stegemeyer. Geodesic Complexity Via Fibered Decompositions of Cut Loci. *Journal of Applied and Computational Topology*, pages 1–29, 2022.

9 Ezra Miller and Igor Pak. Metric Combinatorics of Convex Polyhedra: Cut Loci and Nonoverlapping Unfoldings. *Discrete & Computational Geometry*, 39:339–388, 2008.

10 Susan B Niefield. A Note on the Locally Hausdorff Property. *Cahiers de topologie et géométrie différentielle catégoriques*, 24(1):87–95, 1983.
11 David Recio-Mitter. Geodesic Complexity of Motion Planning. *Journal of Applied and Computational Topology*, 5(1):141–178, 2021.

Improved Algorithms for Distance Selection and Related Problems

Haitao Wang
School of Computing, University of Utah, Salt Lake City, UT 84112, USA.

Yiming Zhao[1]
Department of Computer Science, Utah State University, Logan, UT 84322, USA

Abstract

We propose new techniques for solving geometric optimization problems involving interpoint distances of a point set in the plane. Given a set P of n points in the plane and an integer $1 \leq k \leq \binom{n}{2}$, the distance selection problem is to find the k-th smallest interpoint distance among all pairs of points of P. The previously best deterministic algorithm solves the problem in $O(n^{4/3}\log^2 n)$ time [Katz and Sharir, SIAM J. Comput. 1997]. We improve their algorithm to $O(n^{4/3}\log n)$ time. Using similar techniques, we also give improved algorithms for several other problems.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric optimization, distance selection, Fréchet distance, range searching

1 Introduction

We propose new techniques for solving geometric optimization problems involving interpoint distances in a point set in the plane. More specifically, the optimal objective value of these problems is equal to the (Euclidean) distance of two points in the set. Our techniques usually yield improvements over the previous work by at least a logarithmic factor (and sometimes a polynomial factor).

The main problem we consider is the *distance selection* problem: Given a set P of n points in the plane and an integer $1 \leq k \leq \binom{n}{2}$, the problem asks for the k-th smallest interpoint distance among all pairs of points of P. The problem can be easily solved in $O(n^2)$ time. The first subquadratic time algorithm was given by Chazelle [5]; the algorithm runs in $O(n^{9/5}\log^{4/5} n)$ time and is based on Yao's technique [13]. Later, Agarwal, Aronov, Sharir, and Suri [1] gave a better algorithm of $O(n^{3/2}\log^{5/2} n)$ time and subsequently Goodrich [6] solved the problem in $O(n^{4/3}\log^{8/3} n)$ time. Katz and Sharir [7] finally presented an $O(n^{4/3}\log^2 n)$ time algorithm. All above are deterministic algorithms. Several randomized algorithms have also been proposed for the problem. The randomized algorithm of [1] runs in $O(n^{4/3}\log^{8/3} n)$ expected time. Matousek [9] gave another randomized algorithm of $O(n^{4/3}\log^{2/3} n)$ expected time. Very recently, Chan and Zheng proposed a randomized algorithm of $O(n^{4/3})$ expected time (see the arXiv version of [4]). Also, the time complexity can be made as a function of k. In particular, Chan's randomized techniques [3] solved the problem in $O(n\log n + n^{2/3}k^{1/3}\log^{5/3} n)$ expected time and Wang [10] recently improved the algorithm to $O(n\log n + n^{2/3}k^{1/3}\log n)$ expected time; these algorithms are particularly interesting when k is relatively small.

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

[1] Corresponding author.

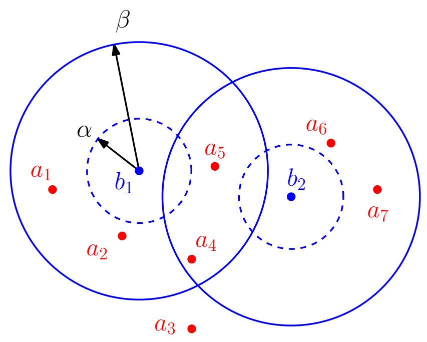


Figure 1 An example of Problem 1 with $\Gamma = \{\{a_1, a_2\} \times \{b_1\}, \{a_4, a_5\} \times \{b_1, b_2\}\}$ and $\Pi = \{\{a_3, a_6, a_7\} \times \{b_2\}\}$. Note that the uncertain pair (a_3, b_2) has a distance $\|a_3b_2\| \notin (\alpha, \beta]$.

2 Our result

We present a new deterministic algorithm that solves the distance selection problem in $O(n^{4/3} \log n)$ time. Albeit slower than the randomized algorithm of Chan and Zheng [4], our algorithm is the first progress on the deterministic solution since the work of Katz and Sharir [7]. One technique we introduce is an algorithm for solving the following *partial batched range searching problem*.

▶ **Problem 1. (Partial batched range searching)** *Given a set A of m points and a set B of n points in the plane and an interval $(\alpha, \beta]$, one needs to construct two collections of edge-disjoint complete bipartite graphs $\Gamma(A, B, \alpha, \beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ and $\Pi(A, B, \alpha, \beta) = \{A'_s \times B'_s \mid A'_s \subseteq A, B'_s \subseteq B\}$ so that the following two conditions are satisfied (see Fig. 1 for an example):*

1. *For each pair $(a, b) \in A_t \times B_t \subseteq \Gamma(A, B, \alpha, \beta)$, the (Euclidean) distance $\|ab\|$ between points $a \in A$ and $b \in B$ is in $(\alpha, \beta]$.*
2. *For any two points $a \in A$ and $b \in B$ with $\|ab\| \in (\alpha, \beta]$, either $\Gamma(A, B, \alpha, \beta)$ has a unique graph $A_t \times B_t$ that contains (a, b) or $\Pi(A, B, \alpha, \beta)$ has a unique graph $A'_s \times B'_s$ that contains (a, b).*

In other words, the two collections Γ and Π together record all pairs (a, b) of points $a \in A$ and $b \in B$ whose distances are in $(\alpha, \beta]$. While all pairs of points recorded in Γ have their distances in $(\alpha, \beta]$, this may not be true for Π. For this reason, we sometimes call the point pairs recorded in Π uncertain pairs*.*

In the traditional BRS, which has been studied with many applications, e.g.,[11, 8, 2], the collection Π is $\emptyset$ (and thus Γ itself satisfies the two conditions in Problem 1); for differentiation, we refer to this case as the *complete BRS*. The advantage of the partial problem over the complete problem is that the partial problem can usually be solved faster, with a sacrifice that some uncertain pairs (i.e., those recorded in Π) are left unresolved. In typical applications the number of those uncertain pairs can be made small enough so that they can be handled easily without affecting the overall runtime of the algorithm. More specifically, we derive an algorithm to compute a solution for the partial BRS, whose runtime is controlled by a parameter (roughly speaking, the runtime increases as the graph sizes of Π decreases). Previously, Katz and Sharir [7] gave an algorithm for the complete problem. Our solution, albeit for the more general partial problem, even improves their algorithm by roughly a logarithmic factor when applied to the complete case.

On the one hand, our partial BRS solution helps achieve our new result for the distance selection problem. On the other hand, combining some techniques for the latter problem, we propose a general algorithmic framework that can be used to solve any geometric optimization problem that involves interpoint distances of a set of points in the plane. Consider such a problem whose optimal objective value (denoted by δ^*) is equal to the distance of two points of a set P of n points in the plane. Assume that the decision problem (i.e., given δ, decide whether $\delta \geq \delta^*$) can be solved in T_D time. A straightforward algorithm for computing δ^* is to use the distance selection algorithm and the decision algorithm to perform binary search on interpoint distances of all pairs of points of P; the algorithm runs in $O(\log n)$ iterations and each iteration takes $O(n^{4/3}\log n + T_D)$ time (if we use our new distance selection algorithm). As such, the total runtime is $O((n^{4/3}\log n + T_D)\log n)$. Using our new framework, the runtime can be bounded by $O((n^{4/3} + T_D)\log n)$, which is faster when $T_D = o(n^{4/3}\log n)$.

Our new techniques lead to improved results on several geometric optimization problems, including *discrete Fréchet distance with shortcuts* [2] (both the one-sided and the two-sided versions) and *reverse shortest paths in unit-disk graphs* [8, 12] (both the weighted and the unweighted cases). Our techniques are quite general and we believe they will find many other applications in future.

References

1 Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting distances in the plane. *Algorithmica*, 9(5):495–514, 1993.

2 Rinat B. Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Transactions on Algorithms*, 11(4):Article No. 29, 2015.

3 Timothy M. Chan. On enumerating and selecting distances. *International Journal of Computational Geometry and Application*, 11:291–304, 2001.

4 Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022. Full version with new results available at https://arxiv.org/pdf/2111.03744.pdf.

5 Bernard Chazelle. New techniques for computing order statistics in Euclidean space. In *Proceedings of the 1st Annual Symposium on Computational Geometry (SoCG)*, pages 125–134, 1985.

6 Michael T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proceedings of the 9th Annual Symposium on Computational Geometry (SoCG)*, pages 73–82, 1993.

7 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997.

8 Matthew J. Katz and Micha Sharir. Efficient algorithms for optimization problems involving semi-algebraic range searching. *arXiv:2111.02052*, 2021.

9 Jiří Matoušek. Randomized optimal algorithm for slope selection. *Information Processing Letters*, 39:183–187, 1991.

10 Haitao Wang. Unit-disk range searching and applications. In *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 32:1–32:17, 2022.

11 Haitao Wang and Yiming Zhao. Reverse shortest path problem in weighted unit-disk graphs. In *Proceedings of the 16th International Conference and Workshops on Algorithms and Computation (WALCOM)*, pages 135–146, 2022.

12 Haitao Wang and Yiming Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023.

13 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Approximate Hausdorff Distance in Doubling Metrics

Oliver A. Chubet
North Carolina State University

Parth Parikh
North Carolina State University

Donald R. Sheehy
North Carolina State University

Siddharth S. Sheth
North Carolina State University

Abstract

The Hausdorff distance is a metric commonly used to compute set similarity. We present an algorithm to compute $(1+\varepsilon)$-approximate directed Hausdorff distance in $\left(\frac{1+\varepsilon}{\varepsilon}\right)^{O(d)} n$ time after preprocessing the n input points into a metric tree data structure, where the space has doubling dimension d. The preprocessing step takes $\left(\frac{1}{\varepsilon}\right)^{O(d)} n \log \Delta$ time, for input with spread Δ.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases data structures, algorithms, ball trees, Hausdorff distance

Funding This research was supported by the NSF under grant CCF-2017980.

1 Introduction

The Hausdorff distance is a metric on compact subsets of a metric space. Let $(X, \mathbf{d})$ be a metric space and let A and B be compact subsets of X. The distance from a point $x \in X$ to the set B is $\mathbf{d}(x, B) := \min_{b \in B} \mathbf{d}(x, b)$. The *directed Hausdorff distance* is $\mathbf{d}_h(A, B) := \max_{a \in A} \mathbf{d}(a, B)$, and the (undirected) *Hausdorff distance* is $\mathbf{d}_H(A, B) := \max\{\mathbf{d}_h(A, B), \mathbf{d}_h(B, A)\}$. This definition leads directly to a quadratic time algorithm for finite sets.

We focus on general metrics spaces that have finite doubling dimension. The *doubling dimension* of X is the smallest real number d such that any metric ball in X can be covered by at most 2^d balls of half the radius. In the general setting, one may not expect a subquadratic algorithm for computing the Hausdorff distance, however, there are classes of metric spaces for which the Hausdorff distance can be computed more quickly. For example, given point sets in the plane, it is possible to use fast nearest neighbor search data structures [1] to give an $O(n \log n)$ time algorithm. If one allows for approximate answers, $O(n \log n)$ time algorithms are possible in low-dimensional Euclidean spaces [2]. In practice, speed-ups to the naïve algorithm use heuristics [3, 10, 11] or geometric data structures [7, 12]. We use linear-size greedy trees [4].

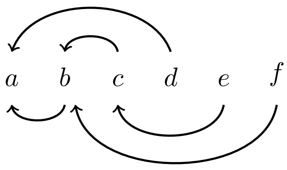
Given a permutation $(p_0, \ldots, p_{n-1})$ of a set $P \subseteq X$, let $P_i := \{p_0, \ldots, p_{i-1}\}$ denote the ith *prefix* of the permutation. A permutation of P is a *greedy permutation* if for all i, the ith

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

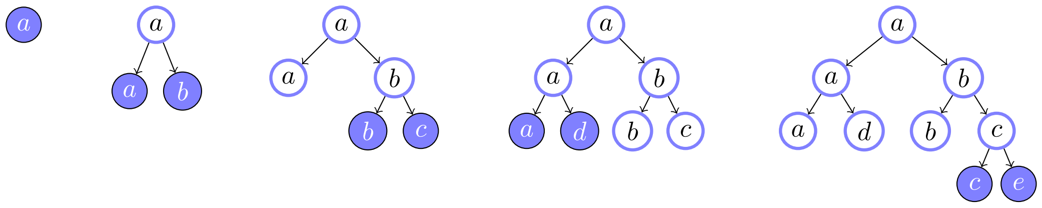
point is the farthest point in the set from the ith prefix of the permutation. That is, for all i,

$$\mathbf{d}(p_i, P_i) = \max_{p \in P} \mathbf{d}(p, P_i).$$

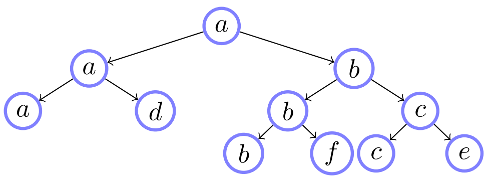
A *greedy tree* on set A is a ball tree [8] with packing guarantees on the nodes. A ball tree on A is a binary tree formed by recursively partitioning A. A ball tree node $\mathbf{a}$ has center $a \in A$, and radius r_a. A greedy tree uses the greedy permutation on A and a mapping from points of A to their predecessors in the greedy permutation to guide the partition. To construct a greedy tree, start with a node for the seed and incrementally construct the tree topology first. Create nodes for every point in the permutation and its predecessor, and attach them as the right and left child respectively, of the predecessor's node in the tree (see Figure 1). Once the tree is constructed, compute the radii for all nodes. The points of A are stored as the leaves of the greedy tree. The radius of a node is the distance from the center to the farthest leaf in its subtree. The radius of a child is never greater than the radius of its parent in a greedy tree. A subset of greedy tree nodes such that every point of A is contained as a leaf in exactly one node can be used to approximate A.



(a) The permutation (a, b, c, d, e, f) is depicted with arrows representing a predecessor mapping.



(b) The tree is constructed incrementally. Each new point creates two new nodes.



(c) The completed greedy tree.

Figure 1 A ball tree is computed for a given permutation and predecessor pairing.

Given two greedy trees, our algorithm maintains approximations for either set and tracks which nodes of B are close to nodes of A. Approximating all the leaves in a subtree by the node center batches the searches and results in fewer distance computations. Nutanong et al. [7] take a similar dual-tree approach, studied more in the machine learning literature[5, 6, 9].

Our algorithm computes a $(1 + \varepsilon)$-approximation of the Hausdorff distance between two sets with a total of n points in $\left(\frac{1+\varepsilon}{\varepsilon}\right)^{O(d)} n$ time after preprocessing the input sets individually into greedy trees [4]. The preprocessing takes $\left(\frac{1}{\varepsilon}\right)^{O(d)} n \log \Delta$ time. Here Δ is the *spread*, defined as the ratio of the largest to smallest pairwise distances, of the input sets, and d is the doubling dimension of the metric space. The preprocessing is especially useful when the same sets are involved in multiple distance computations.

2 Approximate Hausdorff Distance Algorithm

The input to HAUSDORFF is a list of the nodes of greedy trees G_A and G_B sorted by non-increasing radii such that no child precedes its parent, and a parameter $\varepsilon > 0$. The output is a $(1+\varepsilon)$-approximation of $\mathbf{d}_h(A, B)$. The main data structure is a bipartite graph on the nodes of G_A and G_B, called the *neighbor graph* N. The neighbor graph satisfies the following invariants:

1. Every point of $A \cup B$ is a leaf in some vertex of N.
2. If $\mathbf{d}(a, B) = \mathbf{d}(a, b)$ then there is an edge between the nodes containing a and b.

Let $N(\mathbf{a})$ denote the set of neighbors of $\mathbf{a}$ in N. If for some $\mathbf{a} \in N$ we have $\mathbf{a} \in G_A$, then the neighbor graph also maintains $l(a)$, a lower bound on $\mathbf{d}(a, B)$ defined as,

$$l(a) := \min_{\mathbf{b} \in N(\mathbf{a})} \{\mathbf{d}(a, b) - r_b\}.$$

Initialize N to contain the roots of G_A and G_B connected by an edge and then iterate over the input list. For every node, $\mathbf{p}$, replace it in N by its children and connect them to the same vertices that were adjacent to $\mathbf{p}$.

We prevent N from becoming complex by pruning edges that are too long. Let the vertex set of N be $A' \sqcup B'$ at the start of some iteration. An edge between $\mathbf{a}$ and $\mathbf{b}$ can be safely pruned if $\mathbf{d}(a, b') + r_a < \mathbf{d}(a, b) - r_a - r_b$ for some $\mathbf{b}' \in B'$, without violating the neighbor invariant (see Figure 3). In HAUSDORFF, if $\mathbf{p} \in G_A$, then prune long edges adjacent to the newly added vertices and update their lower bounds. Otherwise, $\mathbf{p} \in G_B$ and so, prune edges adjacent to $\mathbf{a}$ for all $\mathbf{a} \in N(\mathbf{p})$ and update $l(a)$. A lower bound update is shown in Figure 2.

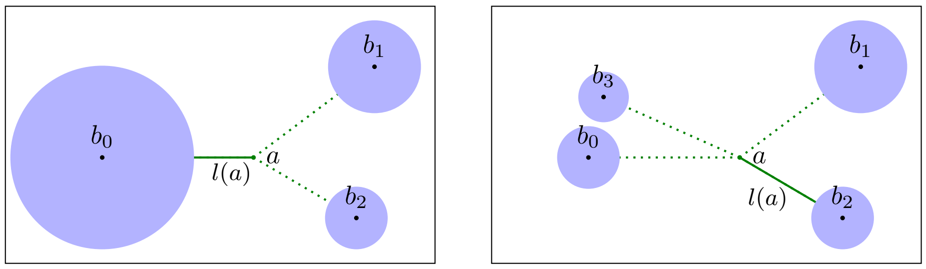


Figure 2 This figure depicts an update of $l(a)$ after replacing a node $\mathbf{b_0}$ with its children.

We stop once the unprocessed nodes have radii too small to significantly affect the lower bounds. HAUSDORFF maintains L as the greatest local lower bound. Figure 4 depicts how L is updated when local lower bounds change. If the radius r of the largest node yet to process satisfies $r \leq \frac{\varepsilon}{2} L$, then the algorithm returns L as a $(1+\varepsilon)$-approximation of $\mathbf{d}_h(A, B)$.

If the radius of every vertex in the neighbor graph is non-zero, then this fact, along with pruning and some properties of the greedy tree, allows one to bound the degree of a vertex by a packing argument. This leads to constant time graph updates and so, we have linear running time. We conclude the following theorem.

▶ **Theorem 1.** *Given two greedy trees for sets A and B of total cardinality n, HAUSDORFF computes a $(1+\varepsilon)$-approximation of $\mathbf{d}_h(A, B)$ in $\left(\frac{1+\varepsilon}{\varepsilon}\right)^{O(d)} n$ time.*

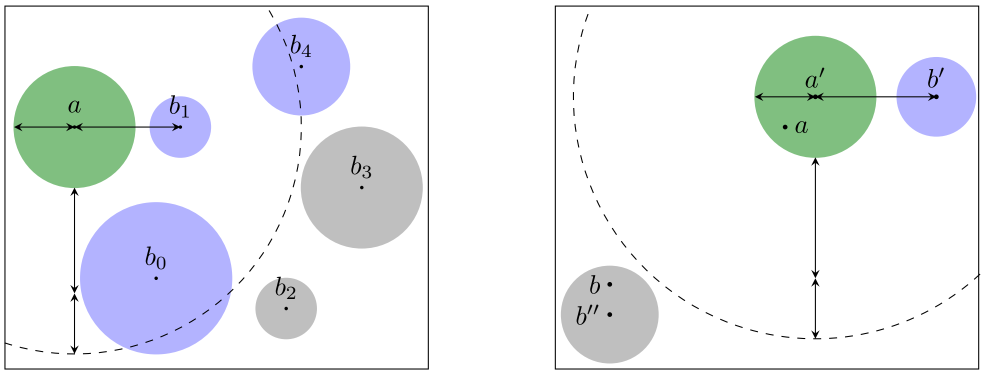


Figure 3 This figure shows the role of the pruning condition. On the left, the nodes $\mathbf{b_2}$ and $\mathbf{b_3}$ are too far away to contain the nearest neighbor of any point in $\mathbf{a}$, so edges $(\mathbf{a}, \mathbf{b_2})$ and $(\mathbf{a}, \mathbf{b_3})$ can be pruned from the neighbor graph. The pruning condition respects the neighbor invariant and does not prune edge $(\mathbf{a}, \mathbf{b_4})$. In the right image let $\mathbf{a}' \in A'$. For any point a in $\mathbf{a}'$, by the triangle inequality, $\mathbf{d}(a, B) \leq r_{a'} + \mathbf{d}(a', b')$. If edge $(\mathbf{a}, \mathbf{b}'')$ has been pruned, then no point b in $\mathbf{b}''$ can be the nearest neighbor of a, because $\mathbf{d}(a, b') \leq \mathbf{d}(a, b)$ for any such b.

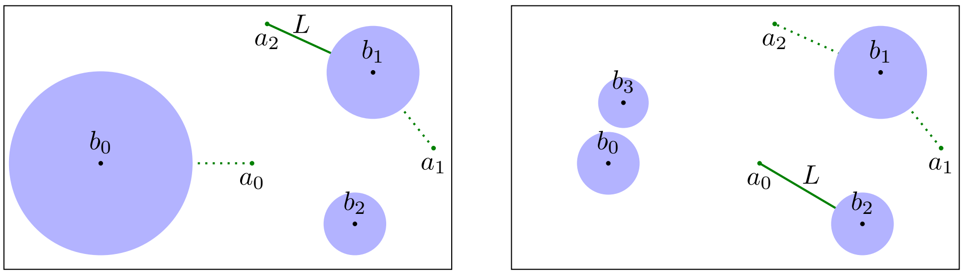


Figure 4 This figure depicts an update of L after replacing the node $\mathbf{b_0}$ with its children.

References

1 Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3-4):251–265, September 1995. URL: `http://link.springer.com/10.1007/BF01530830`, `doi:10.1007/BF01530830`.

2 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, November 1998. URL: `https://dl.acm.org/doi/10.1145/293347.293348`, `doi:10.1145/293347.293348`.

3 Yilin Chen, Fazhi He, Yiqi Wu, and Neng Hou. A local start search algorithm to compute exact Hausdorff Distance for arbitrary point sets. *Pattern Recognition*, 67:139–148, July 2017. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0031320317300559`, `doi:10.1016/j.patcog.2017.02.013`.

4 Oliver Chubet, Parth Parikh, Donald R. Sheehy, and Siddharth Sheth. *Proximity Search in the Greedy Tree*, pages 332–342. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611977585.ch29`, `arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611977585.ch29`, `doi:10.1137/1.9781611977585.ch29`.

5 Ryan Curtin, William March, Parikshit Ram, David Anderson, Alexander Gray, and Charles Jr. Tree-independent dual-tree algorithms. *30th International Conference on Machine Learning, ICML 2013*, 04 2013.

6 Alexander Gray and Andrew Moore. `n-body'problems in statistical learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL: `https://proceedings.neurips.cc/paper/2000/file/7385db9a3f11415bc0e9e2625fae3734-Paper.pdf`.

7 Sarana Nutanong, Edwin H. Jacox, and Hanan Samet. An incremental Hausdorff distance calculation algorithm. *Proceedings of the VLDB Endowment*, 4(8):506–517, May 2011. URL: `https://dl.acm.org/doi/10.14778/2002974.2002978`, `doi:10.14778/2002974.2002978`.

8 Stephen M. Omohundro. Five balltree construction algorithms. Technical Report 562, ICSI Berkeley, 1989.

9 Parikshit Ram, Dongryeol Lee, William March, and Alexander Gray. Linear-time algorithms for pairwise statistical problems. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL: `https://proceedings.neurips.cc/paper/2009/file/2421fcb1263b9530df88f7f002e78ea5-Paper.pdf`.

10 Jegoon Ryu and Sei-ichiro Kamata. An efficient computational algorithm for Hausdorff distance based on points-ruling-out and systematic random sampling. *Pattern Recognition*, 114:107857, June 2021. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0031320321000443`, `doi:10.1016/j.patcog.2021.107857`.

11 Abdel Aziz Taha and Allan Hanbury. An Efficient Algorithm for Calculating the Exact Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2153–2163, November 2015. URL: `http://ieeexplore.ieee.org/document/7053955/`, `doi:10.1109/TPAMI.2015.2408351`.

12 Dejun Zhang, Fazhi He, Soonhung Han, Lu Zou, Yiqi Wu, and Yilin Chen. An efficient approach to directly compute the exact Hausdorff distance for 3D point sets. *Integrated Computer-Aided Engineering*, 24(3):261–277, July 2017. URL: `https://www.medra.org/servlet/aliasResolver?alias=iospress\&doi=10.3233/ICA-170544`, `doi:10.3233/ICA-170544`.

Kd-trees Work with Separable Bregman Divergences

Tuyen Pham
University of Florida, Gainesville, US

Hubert Wagner
University of Florida, Gainesville, US

Abstract

We report work in progress on generalizing Kd-trees to a broad family of Bregman divergences. Our focus is on separable Bregman divergences, which include practical divergences such as the Kullback–Leibler divergence (relative entropy) and the Itakura–Saito divergence.

2012 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems; Theory of computation → Computational geometry; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Kd-tree, nearest neighbor search, Bregman divergence, separable Bregman divergence, KL-divergence, relative entropy

Funding 2022 Google Research Scholar Award in Algorithms and Optimization.

1 Introduction

Many techniques in computational geometry were developed with the Euclidean distance – or a more general metric distance – in mind. However, in many applied settings – particularly modern machine learning – non-metric distances play an important role. One example is the Kullback–Leibler divergence (KL), commonly used to compare discrete probability distributions [8]. In practice, it often serves as a loss to be minimized [15, 9, 10] – often under the name of relative entropy or the related cross entropy. KL is one member of the family of Bregman divergences – on which we center our attention.

A range of computational geometry techniques have been extended to the setting of Bregman divergences. This includes vantage-point trees [12], ball-trees [13], k-means clustering [1], Voronoi diagrams and Delaunay triangulations [3], Čech complexes [6]. We show that Kd-trees work for a class of Bregman divergences called separable Bregman divergences [11].

2 Background

We begin by setting up the definitions for Bregman divergences [4], which will serve as measures of distance. We note that they are usually not symmetric and never satisfy the triangle inequality – and as such do not define a proper metric.

A *function of Legendre type* [14] is a function $F : \Omega \to \mathbb{R}$ where $\Omega \subseteq \mathbb{R}^n$ is a convex set and F is differentiable, strictly convex, and satisfies $\lim_{x \to \partial \Omega} \|\nabla F(x)\| = \infty$ if $\partial \Omega$ is nonempty. Given a function F of Legendre type, the Bregman divergence is the difference between F and the best linear approximation of F at y, both evaluated at x:

$$D_F(x \| y) = F(x) - (F(y) + \langle \nabla F(y), x - y \rangle).$$

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

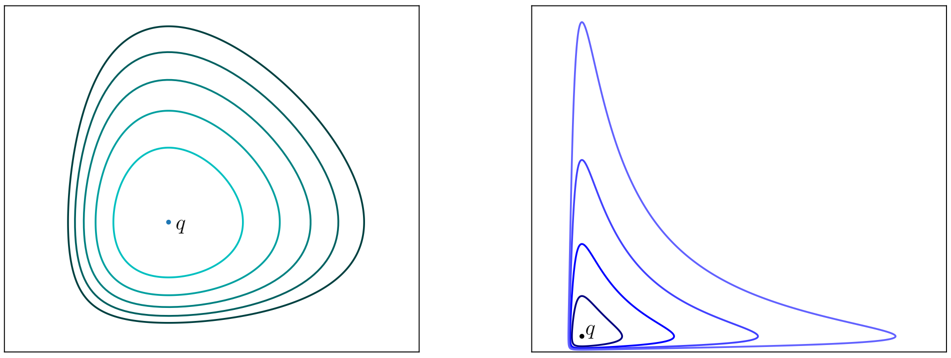


Figure 1 Left: primal generalized Kullback–Leibler balls. Right: Itakura–Saito balls

Separable Bregman divergences. In particular, a separable Bregman divergence can be decomposed into univariate divergences, i.e., $D_F(x\|y) = \sum_{i=1}^n D_{f_i}(x_i\|y_i)$ [11]. These divergences have seen many uses in machine learning. One popular example of separable Bregman divergences is the Kullback–Leibler (KL) divergence, $D_{\mathrm{KL}}(x\|y) = \sum_{i=1}^n x_i \log\left(\frac{x_i}{y_i}\right)$, which is a standard distance between probability distributions [8] and is commonly used as a loss function in algorithms such as t-SNE [15], UMAP [9] and in deep learning [10]. We work with the generalized KL divergence $D_{\mathrm{GKL}}(x\|y) = \sum_{i=1}^n (x_i \log\left(\frac{x_i}{y_i}\right) + x_i - y_i)$ defined on $\mathbb{R}_+^n$ (which reduces to the usual KL for points on the open standard simplex). Another example is the Itakura–Saito (IS) divergence [7], $D_{\mathrm{IS}}(x\|y) = \sum_{i=1}^n \left(\frac{x_i}{y_i} - \log\left(\frac{x_i}{y_i}\right) - 1\right)$ defined on $\mathbb{R}_+^n$, which is useful for speech and sound data [5].

We define the *primal Bregman ball* of radius $r \geq 0$ and center q as

$$B_F(q;r) = \{y \in \Omega : D_F(q\|y) \leq r\}.$$

Namely, it is the collection of points in the domain with Bregman divergence measured *from* the center not exceeding r. See Figure 1 for an illustration. Primal Bregman balls have a particularly nice geometric interpretation: given a light at point $(q, F(q) - r)$, the ball $B_F(q;r)$ is the *illuminated* part of the graph of F projected onto Ω. Due to the asymmetry we could define dual Bregman balls, but we limit our attention to the primal ones.

Kd-trees. We briefly overview a simple version of the Kd-tree [2], which is a useful data-structure for nearest neighbor queries. We construct it as a binary tree encoding repeated partitions of Ω with axis-aligned hyperplanes. In finding the nearest neighbor of $q \in \Omega$ among $X \subset \Omega$, the crucial step is checking if the points on the other side of a hyperplane can be safely pruned. This reduces to checking if the hyperplane intersects the ball centered at q of radius equal to the distance to the current nearest neighbor candidate. With the Euclidean metric, this check is trivial. We now consider a version of this problem for separable Bregman divergences. We focus on finding $\operatorname{argmin}_{x\in X} D_F(q\|x)$; the other case is analogous.

3 Hyperplane Intersection Problem

Given an axis-aligned hyperplane $P \subset \mathbb{R}^n$ and a point $q \in \Omega \subset \mathbb{R}^n$, we check if the hyperplane intersects a Bregman ball of radius $r \geq 0$ centered at q. This can be solved by finding the Bregman projection of q onto P, namely the point $q_P = \operatorname{arginf}_{p\in P\cap\Omega} D_F(q\|p)$.

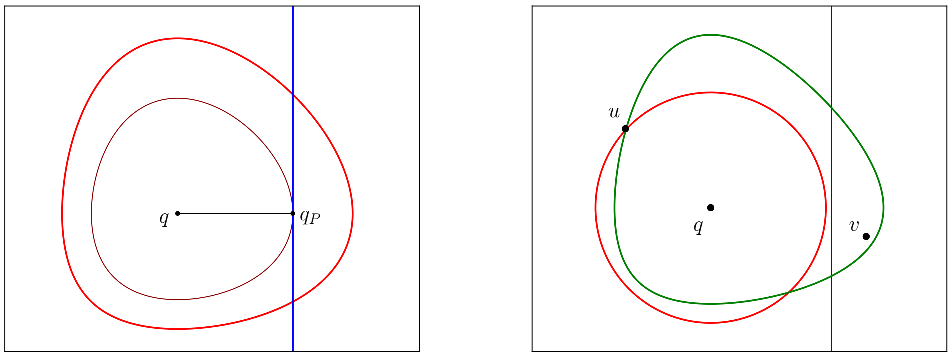


Figure 2 Left: A GKL ball intersects the hyperplane and contains the ball of radius $\inf_{p\in P} D_{GKL}(q\|p)$. Right: An IS ball intersects the hyperplane while the Euclidean ball does not, resulting in different nearest neighbors.

▶ **Lemma 1** (Hyperplane projection lemma). *Let $P \subset \mathbb{R}^n$ be an axis-aligned hyperplane such that $P \cap \Omega \neq \emptyset$, and let D_F be a separable Bregman divergence. Given $q \in \Omega$, the Bregman projection of q onto P is well-defined and coincides with the Euclidean projection of q onto P.*

Proof. Without loss of generality, we consider P with fixed $p_1 = c$ for each $p \in P$. Let $q = (q_1, q_2, \ldots, q_n)$ and $D_F(x\|y) = \sum_{i=1}^n D_{f_i}(x_i\|y_i)$. Since each D_{f_i} is a Bregman divergence, $D_{f_i}(x\|y) \geq 0$ with $D_{f_i}(x_i\|y_i) = 0$ if and only if $x_i = y_i$. As q is fixed and $p_1 = c$ for each $p \in P$, $\min_{p\in P\cap\Omega} D_F(q\|p) \geq D_{f_1}(q_1\|p_1)$, with equality if and only if $p_i = q_i$ for $i = 2, 3, \ldots, n$. With $f_i : \omega_i \to \mathbb{R}$ being a function of Legendre type, D_F is well-defined on Ω^2 where $\Omega = \prod_{i=1}^n \omega_i$ and so $q_P = (p_1, q_2, q_3, \ldots, q_n) \in \Omega$ and $D_F(q\|q_P)$ is well defined.

Thus the Bregman projection of q onto an arbitrary axis-aligned hyperplane P with the i^{th} coordinate fixed at p_i is given by $q_P = (q_0, q_1, \ldots, q_{i-1}, p_i, q_{i+1}, \ldots, q_n)$. ◀

With this projection, we simply compare $D_F(q\|q_P)$ with r to determine if the hyperplane intersects the Bregman ball $B_F(q; r)$. Despite the lack of the triangle inequality, this allows us to decide if the points on the side of the hyperplane opposite to q can be safely pruned. Indeed, these points are in the complement of the Bregman ball and thus have greater Bregman divergence. See Figure 2 for an illustration.

4 Summary

We showed that Kd-trees work with separable Bregman divergences. In particular, this result allows for efficient queries of probability distributions measured with the Kullback–Leibler divergence. We intend to sharpen this result for the important special case of points living on the simplex, which differs significantly from the Euclidean case.

We plan to test the efficiency of Kd-trees for various divergences and compare it to the alternatives mentioned in the introduction. We stress that, unlike most other methods, Kd-trees allow the divergence to be chosen *after* the data-structure is constructed. It makes the method an interesting choice for situations in which queries with respect to multiple divergences are performed or when the divergence changes over time.

References

1 Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6(58):1705–1749, 2005. URL: http://jmlr.org/papers/v6/banerjee05b.html.

2 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, 1975.

3 Jean-Daniel Boissonnat, Frank Nielsen, and Richard Nock. Bregman Voronoi diagrams. *Discrete and Computational Geometry*, 44:281–307, 2010. doi:10.1007/s00454-010-9256-1.

4 Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967. doi:https://doi.org/10.1016/0041-5553(67)90040-7.

5 Minh N Do and Martin Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Transactions on Image Processing*, 11(2):146–158, 2002.

6 Herbert Edelsbrunner and Hubert Wagner. Topological data analysis with Bregman divergences. In *33th International Symposium on Computational Geometry (SoCG)*, pages 67–86, 2017. doi:https://doi.org/10.20382/jocg.v9i2a6.

7 Fumitada Itakura. Analysis synthesis telephony based on the maximum likelihood method. *Reports of the* 6^{th} *Int. Cong. Acoust.*, 1968.

8 Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

9 Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi:10.21105/joss.00861.

10 Kevin P. Murphy. *Machine learning: A Probabilistic Perspective*. MIT press, 2012.

11 Frank Nielsen and Richard Nock. The Bregman chord divergence. In *4th International Conference on Geometric Science of Information (GSI)*, pages 299–308. Springer, 2019.

12 Frank Nielsen, Paolo Piro, and Michel Barlaud. Bregman vantage point trees for efficient nearest neighbor queries. pages 878–881, 2009. doi:10.1109/ICME.2009.5202635.

13 Frank Nielsen, Paolo Piro, and Michel Barlaud. Tailored Bregman ball trees for effective nearest neighbors. In *Proceedings of the 25th European Workshop on Computational Geometry (EuroCG)*, pages 29–32, 2009.

14 Ralph T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970. doi:10.1515/9781400873173.

15 Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

A Quadtree for Hyperbolic Space

Sándor Kisfaludi-Bak
Department of Computer Science, Aalto University, Finland

Geert van Wordragen
Department of Computer Science, Aalto University, Finland

Abstract

We propose a data structure in d-dimensional hyperbolic space that can be considered a natural counterpart to quadtrees in Euclidean spaces. Based on this data structure we propose a so-called L-order for hyperbolic point sets, which is an extension of the Z-order defined in Euclidean spaces. We demonstrate the usefulness of our hyperbolic quadtree data structure by giving an algorithm for constant-approximate closest pair and dynamic constant-approximate nearest neighbours in hyperbolic space of constant dimension d.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Computational geometry

Keywords and phrases Hyperbolic geometry, quadtree, approximate nearest neighbour

Related Version A full version of the paper is available at `https://arxiv.org/abs/2305.01356`.

1 Introduction

Hyperbolic geometry has applications in several fields, including special relativity, topology, visualisation, machine learning, complex network modelling, etc. [8–12]. With the growing interest in the larger scientific community, there are growing computational and graphical/visualisation needs. It is becoming increasingly important to develop basic data structures and algorithms for hyperbolic spaces.

Quadtrees in Euclidean spaces [4] are among the few geometric data structures that have proven to be useful both in practical algorithms and in theory [1,3,5]. They form the basis of various data structures by being able to 'zoom in' efficiently. Quadtrees provide a hierarchical structure, as well as a way to think of ordering points of the plane (or higher-dimensional spaces) using the so-called Z-order curve. They can be used as a basis for nearest neighbour algorithms [5]. The central question addressed by this article is as follows.

Is there a natural hyperbolic equivalent to Euclidean quadtrees?

Given a point set P in the Euclidean plane (henceforth denoted by $\mathbb{R}^2$), a quadtree of P can be defined as follows. Let σ_0 be a minimal axis-parallel square containing P, and let T be a tree graph whose root corresponds to σ_0. Then consider a square σ and the corresponding vertex v_σ of T where $|\sigma \cap P| \geq 2$ (starting with $\sigma = \sigma_0$). We subdivide σ into four squares of half the side length of σ. Each smaller square is associated with a new vertex that is connected to v_σ. This procedure is repeated for each square σ where $\sigma \cap P \geq 2$ exhaustively, until all leaves of T correspond to squares that contain at most one point from P. The squares are called the *cells* of the quadtree, and we can speak of parent/child and ancestor/descendant relationships of cells by applying the terms of the corresponding vertices in T. The *level* of a cell or vertex is its distance to the root of T along the shortest path in T (i.e., the root has level 0).

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

Some crucial properties of Euclidean quadtrees are used by several algorithms.

1. If C' is a child cell of C, then $c_1 < \operatorname{diam}(C')/\operatorname{diam}(C) < c_2$ where $0 < c_1 < c_2 < 1$ are fixed constants, and $\operatorname{diam}(C)$ denotes the diameter of the cell C.
2. Each cell C contains a ball that has diameter at least constant times the diameter of C. Thus cells are so-called *fat* objects in $\mathbb{R}^2$.
3. Each cell has at most k children cells for some fixed constant k.
4. Cells of the same level are isometric, that is, any cell can be obtained from any other cell of the same level by using a distance-preserving transformation.

Could the above four properties be replicated by a quadtree in the hyperbolic plane? Unfortunately this is not possible: the volume of a ball in hyperbolic space grows exponentially with its radius (thus hyperbolic spaces are not *doubling spaces*). Consequently, for large cells, a cell of constant times smaller diameter than its parent will only cover a vanishingly small volume of its parent. This rules out having properties 1, 2, and 3 together. Property 4 also poses a unique challenge: while the hyperbolic plane provides many types of *tilings* one could start with, there is no transformation that would be equivalent to scaling in Euclidean spaces. This is unlike the scaling-invariance exhibited by Euclidean quadtrees. Moreover, in small neighbourhoods hyperbolic spaces are locally Euclidean, meaning that a small ball in hyperbolic space can be embedded into a Euclidean ball of the same radius with distortion infinitesimally close to 1. Thus a hyperbolic quadtree needs to operate at two different scales: at small distances we need to work with an almost-Euclidean metric, while at larger distances we need to work on a non-doubling metric.

2 Our contribution

In the full version of the paper, we propose a hyperbolic quadtree that satisfies properties 1, 2, as well as property 4 in case of cells of super-constant diameter. Moreover, our hyperbolic quadtree resembles a Euclidean quadtree for cells of sub-constant diameter. Based on the quadtree we are able to construct a new order and space-filling curve, named the L-order, which serves as a hyperbolic extension of the Euclidean Z-order. We show that a few hyperbolic quadtrees (and corresponding L-orders) can create a useful cover of $\mathbb{H}^d$ in the following sense.

▶ **Theorem 1.** *For any $\Delta \in \mathbb{R}_+$, there is a set of at most $3d+3$ infinite hyperbolic quadtrees such that any two points $p, q \in \mathbb{H}^d$ with $\operatorname{dist}_{\mathbb{H}^d}(p,q) \le \Delta$ are contained in a cell with diameter $\mathcal{O}\left(d\sqrt{d}\right) \cdot \operatorname{dist}_{\mathbb{H}^d}(p,q)$ in one of the quadtrees.*

The above theorem matches the Euclidean result given by Chan, Har-Peled and Jones [2, Lemma 3.7]. We demonstrate the usefulness of our new data structure by presenting two applications related to approximate nearest neighbours. A pair $p, p' \in P$ of distinct points is a *c-approximate closest pair* in P if $\operatorname{dist}(p,p') \le c \cdot \min_{a,a' \in P,\ a \ne a'} \operatorname{dist}(a,a')$. Given the point set P and some query point q in the ambient space, we say that $p \in P$ is a *c-approximate nearest neighbour* of q if $\operatorname{dist}(q,p) \le c \cdot \min_{a \in P} \operatorname{dist}(q,a)$.

▶ **Theorem 2.** *Let $P \subset \mathbb{H}^d$ be a given set of n points.*

- *We can find an $\mathcal{O}\left(d\sqrt{d}\right)$-approximate closest pair of P in $\mathcal{O}\left(d^2 n \log n\right)$ time.*
- *We can construct a data structure in $\mathcal{O}\left(d^2 n \log n\right)$ time that uses $\mathcal{O}\left(d^2 n\right)$ space and can answer queries for an $\mathcal{O}\left(d\sqrt{d}\right)$-approximate nearest neighbour in P in $\mathcal{O}\left(d^2 \log n\right)$ time, and perform updates (point insertions and removals) in $\mathcal{O}\left(d^2 \log n\right)$ time.*

A natural next step is to extend the above to $(1+\varepsilon)$-approximate nearest neighbours as in [2]. Unfortunately this fails to give $o(n)$ query times when $\operatorname{diam}(P) = \log n$, so the problem is left for future work.

3 Related work

Approximate nearest neighbour search in hyperbolic spaces has been studied by Krauthgamer and Lee [7] and by Wu and Charikar [13]. Krauthgamer and Lee describe a data structure of size $\mathcal{O}(n^2)$ that allows queries for a $\mathcal{O}(1)$-additive approximation of the nearest neighbour in $\mathcal{O}(\log^2 n)$ time. They note that this can be further improved to a $(1+\varepsilon)$-approximation using [6]. Wu and Charikar describe various practical methods of using black-box algorithms for Euclidean nearest neighbour search to find exact and $(1+\varepsilon)$-approximate nearest neighbours.

References

1 Srinivas Aluru. Quadtrees and octrees. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004. `doi:10.1201/9781420035179.ch19`.

2 Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM Journal on Computing*, 49(3):583–600, 2020. `doi:10.1137/19M1246493`.

3 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

4 Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974. `doi:10.1007/BF00288933`.

5 Sariel Har-peled. *Geometric Approximation Algorithms*. American Mathematical Society, USA, 2011.

6 Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 798–807. SIAM, 2004. URL: `http://dl.acm.org/citation.cfm?id=982792.982913`.

7 Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 119–132. IEEE Computer Society, 2006. `doi:10.1109/FOCS.2006.9`.

8 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

9 John Lamping, Ramana Rao, and Peter Pirolli. A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, 1995.

10 Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR, 2018. URL: `http://proceedings.mlr.press/v80/nickel18a.html`.

11 William P Thurston. Three dimensional manifolds, kleinian groups and hyperbolic geometry. *Bulletin (New Series) of the american mathematical society*, 6(3):357–381, 1982.

12 Abraham A Ungar. Einstein's special relativity: The hyperbolic geometric viewpoint. *arXiv preprint arXiv:1302.6961*, 2013.

13 Xian Wu and Moses Charikar. Nearest neighbor search for hyperbolic embeddings. *CoRR*, abs/2009.00836, 2020. URL: `https://arxiv.org/abs/2009.00836`, `arXiv:2009.00836`.

From Curves to Words

Hsien-Chih Chang
Department of Computer Science, Dartmouth College, USA.

Brittany Terese Fasy
School of Computing & Department of Mathematical Sciences, Montana State University, USA.

Bradley McCoy
School of Computing, Montana State University, USA.

David L. Millman
Blocky, USA.

Carola Wenk
Department of Computer Science, Tulane University, USA.

Abstract

Blank, in his Ph.D. thesis on determining whether a planar closed curve γ is self-overlapping, constructed a combinatorial word *geometrically* over the faces of γ by drawing cuts from each face to a point at infinity and tracing their intersection points with γ. Independently, Nie, in an unpublished manuscript, gave an algorithm to determine the minimum area swept out by any homotopy from a closed curve γ to a point. Nie constructed a combinatorial word *algebraically* over the faces of γ inspired by ideas from geometric group theory, followed by dynamic programming over the subwords. In this paper, we examine the definitions of the two words and prove the equivalence between Blank's word and Nie's word under the right assumptions.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph theory

Keywords and phrases curve representation, crossing sequence, homotopy area, self-overlapping curve, fundamental group, punctured plane, change of basis

Funding *Brittany Terese Fasy*: NSF DMS 1664858
Bradley McCoy: NSF DMS 1664858

1 Introduction

A *closed curve* in the plane is a continuous map γ from the circle $\mathbb{S}^1$ to the plane $\mathbb{R}^2$. In this paper, we are given a *generic* planar curve meaning there are finitely many self-crossing points in the curve, and each of them is a *transverse* intersection.

In order to work with planar curves, one must choose a *representation*. Blank [1] determines if a curve is *self-overlapping*—boundary of an immersed disk—by checking whether a property holds on a combinatorial word, constructed by drawing cables from each face to infinity then traversing the curve and recording the signed intersection sequence of the curve and the cables. Nie [5] computes the minimum null homotopy area by performing dynamic programming on a combinatorial word. Nie represents a curve algebraically as a word in $\pi_1(\gamma)$. Motivated by simplifying Nie's proof of correctness, we show that Blank and Nie's word constructions are equivalent under modest assumptions.

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

2 Curves and Words

We now describe Blank's word construction [1, page 5]. Let γ be a generic closed curve in the plane. Pick a point in the unbounded face of γ and call it the *basepoint* p_0. From each bounded face f_i, pick a *representative point* p_i. Now connect each p_i to p_0 by a simple path in such a way that no two paths intersect each other. We call the collection of such simple paths a *cable system*, denoted as Π, and each individual path π_i from p_i to p_0 as a *cable*.

Orient each π_i from p_i to p_0. Now traverse γ from an arbitrary *starting point* of γ and construct a cyclic word by writing down the indices of γ crossing the cables π_i in the order they appear on γ; each index i has a *positive* sign if we cross π_i from right to left and a *negative* sign if from left to right. We denote negative crossing with an overline $\overline{\mathtt{i}}$. We call the resulting combinatorial word over the faces a *Blank word* of γ with respect to Π, denoted as $[\gamma]_B(\Pi)$. Figure 1 provides an example. Note that changing the starting point corresponds to a cyclic permutation of the word. Words with this property are called *cyclic words*. We make additional assumptions to organize the cable system onto a tree, an organized cable system is said to be *managed*.

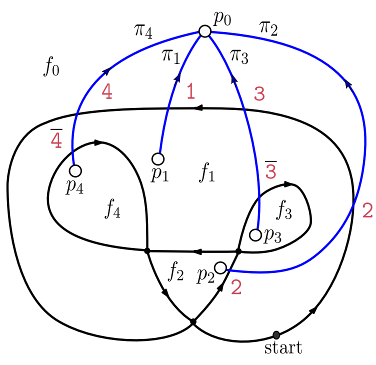


Figure 1 A curve γ with labeled faces and edges, Π_a is drawn in blue. The Blank word of γ corresponding to Π_a is $[\gamma]_B(\Pi_a) = [\mathtt{2314\overline{2}\overline{3}\overline{4}}]$.

We next describe Nie's word construction [5]. Choose a point p_i for each bounded face f_i of γ; denote the collection of points as P. Consider the punctured plane $X := \mathbb{R}^2 \setminus P$ and its fundamental group $\pi_1(X)$. Choose a basepoint x_0 in X and a set of generators Σ for $\pi_1(X)$, where each x_i in Σ represents the generator of $\pi_1(\mathbb{R}^2 \setminus \{p_i\}, x_0) \cong \mathbb{Z}$. These choices, along with a choice of a path from x_0 to each p_i, determine a map from each generator of $\pi_1(\mathbb{R}^2 \setminus \{p_i\})$ into $\pi_1(X)$ [6]. The fundamental group $\pi_1(X)$ is a free group over such generators, and the curve γ can be represented as a word over generators of $\pi_1(X)$. Elements of $\pi_1(X)$ are *free words*.

Consider the curve γ as a four-regular plane graph. Decompose γ into a spanning tree T and the complementary cotree (spanning tree of the dual graph) T^*, the trees (T, T^*) are a *tree-cotree* pair [4]. There are two natural sets of generators for $\pi_1(X)$: (1) the set of all cotree edges, and (2) the set of all face boundaries. We describe the change-of-basis between the two sets of generators in graph-theoretic terms. Traverse γ from some arbitrary starting point and orient each edge of γ accordingly. For each face f_i, define the *boundary operator* ∂ by mapping face f_i to the signed cyclic sequence of edges around face f_i, where each edge is signed positively if it is oriented counter-clockwise and negatively otherwise.

Now, write the curve γ as a cyclic word over the cotree edges T^* by traversing γ, ignoring all tree edges in T. We perform the following procedure inductively on the cotree T^* to construct another cyclic word, this time as an element in the free group over the faces of γ. Starting from the leaves f of T^*, rewrite each edge e bounding the face f as a singleton word, with positive sign if edge e is oriented counter-clockwise, or with negative sign otherwise. Next, for any internal node f of T^*, the boundary ∂f consists of a sequence of (1) tree edges, (2) cotree edges to children of f in T^* denoted as $e_1, e_2, \ldots, e_r$, and (3) (a unique) cotree edge to parent of f denoted as e_f : $\partial f = [e_f e_1 e_2 \ldots e_r]$.

We inductively rewrite each child cotree edge e_i as a free word w_i over the faces. Such words are *face words*. We emphasize that each word for the child cotree edge constructed inductively is a free word, not a cyclic word. Choose a particular but arbitrary way to break the cyclic sequence of faces and rewrite the equation: $e_f = \bar{w}_r \cdot \cdots \cdot \bar{w}_{j+1} \cdot (\bar{w}_j)' \cdot \partial f \cdot (\bar{w}_j)'' \cdot \bar{w}_{j-1} \cdot \cdots \cdot \bar{w}_1$, where $\bar{w}_j = (\bar{w}_j)'(\bar{w}_j)''$ is a particular way of breaking the face word $\bar{w}_j$ in two.

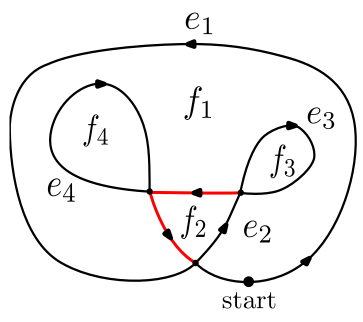


Figure 2 A curve γ with labeled faces and edges and tree in red. One cycle flattening of the boundaries gives $\partial(f_1) = e_3\bar{e}_2e_1e_4, \partial(f_2) = e_2, \partial(f_3) = \bar{e}_3$ and $\partial(f_4) = \bar{e}_4$. Write $\gamma = e_1e_2e_3e_4$ then use the boundaries to change the basis. The Nie word of γ is $[\gamma]_N(\Sigma) = [2314\overline{2}\overline{3}\overline{4}]$.

This gives us a free word over the faces for edge e_f, and, by induction, we have rewritten γ as a free word over the faces. Finally, we can turn the free word back into a cyclic word, by observing that the cyclic permutation of the constructed free word over the faces does not affect the element we are getting in $\pi_1(X)$ (but as a side effect of choosing the basepoint p_0 of γ). We call the resulting signed sequence of faces the *Nie word* and denoted as $[\gamma]_N(\Sigma)$, where Σ is the choices we made when breaking up the cyclic word at each cotree edge, referred to as a *cycle flattening*. Notice that the definition of $[\gamma]_N$ depends on how we choose to break the cyclic edge sequences, and thus is not well-defined without specifying the choices. The proof follows by induction.

▶ **Theorem 1** (Word Equivalence). *Let γ be any plane curve. For a Nie word $[\gamma]_N(\Sigma)$ with a fixed cycle flattening Σ, there is a managed cable system Π such that the Blank word $[\gamma]_B(\Pi)$ is equal to $[\gamma]_N(\Sigma)$. Conversely, any managed cable system Π induces a cycle flattening Σ such that $[\gamma]_B(\Pi)$ and $[\gamma]_N(\Sigma)$ are equal.*

References

1 Samual Joel Blank. *Extending immersions and regular homotopies in codimension 1*. PhD thesis, Brandeis University, 1967.

2 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. *Discrete Comput. Geom.*, 58:889, 2017.

3 Dennis Frisch. Extending immersions into the sphere. 2010. URL: http://arxiv.org/abs/1012.4923.

4 David Eppstein. Dynamic generators of topologically embedded graphs. In *ACM-SIAM Symp. on Discrete Algorithms*, pages 599–608, 2003.

5 Zipei Nie. On the minimum area of null homotopies of curves traced twice. 2014. URL: http://arxiv.org/abs/1412.0101.

6 Valentin Poénaru. Extension des immersions en codimension 1 (d'après Samuel Blank). *Séminaire N. Bourbaki (1966–1968)*, 10:473–505, 1968.

7 G. Toussaint. On separating two simple polygons by a single translation. *Discrete Comput. Geom.*, 4(3):265–278, June 1989.

A Distance for Geometric Graphs via the Labeled Merge Tree Interleaving Distance

Erin Wolf Chambers
Saint Louis University

Elizabeth Munch
Michigan State University

Sarah Percival
Michigan State University

Xinyi Wang
Michigan State University

Abstract

Geometric graphs appear in many real world datasets, such as road networks, sensor networks, and molecules. We investigate the notion of distance between graphs and present a semi-metric to measure the distance between two geometric graphs via the directional transform combined with the labeled merge tree distance. Our distance is not only reflective of the information from the input graphs, but also can be computed in polynomial time. We illustrate its utility by implementation on a *Passiflora* leaf dataset.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Mathematics of computing → Graphs and Surfaces

Keywords and phrases Geometric graphs, merge trees, directional transform

1 Introduction

Many real world datasets materialize as geometric graphs, from road networks to consumer preferences to our dataset of study: outlines of leaves. Our work is motivated by the question of how we can define and efficiently compute a distance between any two graphs in a way that encodes information about the embedding itself. We turn to the idea of the directional transform [2, 4, 7, 8], which encodes a shape $\mathbb{X} \subseteq \mathbb{R}^d$ as a family of $\mathbb{R}$-valued functions $f_\omega : \mathbb{X} \to \mathbb{R}$ for each unit vector $\omega \in \mathbb{S}^{d-1}$. Because many of the standard tools of Topological Data Analysis (such as persistent homology, the Euler characteristic transform, merge trees, and Reeb graphs) all take an $\mathbb{R}$-valued function as input, we can convert this information into a signature providing our favorite representation for each $\omega \in \mathbb{S}^{d-1}$. In this paper, our contribution is to combine the directional transform and the merge tree, with the goal of creating a distance to compare the embedded graphs. Due to the intractability of many available merge tree distances, we focus on the case of labeled merge trees [6, 3] where computation is possible. We construct an algorithm to determine the merge tree of an embedded graph for any direction ω while only needing to compute finitely many merge trees. We show that the resulting distance has potential for use in practical applications by applying it to a real data set of *Passiflora* leaves [1].

2 Methods

Our inputs are two topological graphs G_1 and G_2, each with a map into $\mathbb{R}^2$, $f_i : G_i \to \mathbb{R}^2$. For a fixed direction ω in circle $\mathbb{S}^1$, we define a function on each graph using the inner product, $f_{i,\omega} : G_i \to \mathbb{R}$ by $f_{i,\omega}(x) = \langle f_i(x), \omega \rangle$. Based on this, we can construct a merge tree by encoding the sublevel set components, $\pi_0(f_i^{-1}(-\infty, a])$. Combinatorially, a merge tree is

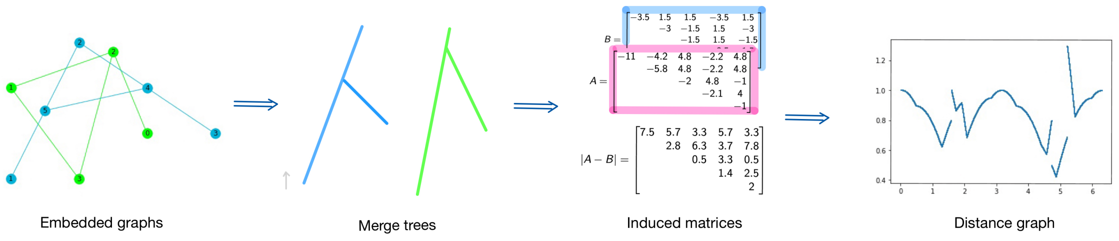


Figure 1 A sketch of the distance calculation procedure for two graphs.

a pair (T, f) consisting of a rooted tree T, along with a function $f : V \to \mathbb{R} \cup \{\infty\}$. This function is finite except for the root, where $f(r) = \infty$ and every vertex v has exactly one neighbor u with $f(u) \geq f(v)$. For a more detailed definition of merge trees, see [5].

We are actually interested in the case of *labeled merge trees*, where the input data is a triple (T, f, Π). Letting $[k] := \{1, 2, \cdots, k\}$, the third entry is the label map $\Pi : [k] \to V(T)$, which we require to be surjective on the leaves of T. This labeled merge tree can be stored as a matrix $\mathcal{M}(T, f, \Pi)_{ij} = f(\mathrm{LCA}(\Pi(i), \Pi(j)))$, where $LCA(v, w)$ is the lowest common ancestor of both v and w. Given two labeled merge trees with the same label set $[k]$, we can obtain two matrices of the same size and then determine the distance between the trees by taking the pointwise L_∞ norm, given by $\|A\|_\infty = \max |a_{ij}|$. Specifically, our distance between labeled merge trees is given by $d((T_1, f_1, \Pi_1), (T_2, f_2, \Pi_2)) := \|\mathcal{M}(T_1, f_1, \Pi_1) - \mathcal{M}(T_2, f_2, \Pi_2)\|_\infty$. See Figure 1 for a sketch of the process.

Given the two input graphs which we equate with their 1-dimensional images in the plane, we define the following labeling scheme to ensure that we end up with merge trees having the same label set. Let V_1 and V_2 denote the set of vertices of G_1 and G_2. We define the subsets $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$ to be the sets of vertices that are *not* in the convex hull of their neighbors. These sets contain all vertices which give birth to a leaf in a merge tree for some direction. For indexing, denote $V_1' = \{v_1, \cdots, v_n\}$ and $V_2' = \{w_{n+1}, \cdots, w_{n+m}\}$. For each $v_i \in V_1'$, let p be the closest point in G_2 via $\|f_1(v_i) - f_2(w_i)\|$ (note this might be internal to an edge), splitting the edge of G_2 to create a vertex if needed. Similarly, find the points $q \in G_1$ from $w_j \in V_2'$, adding a vertex if necessary. This gives a labeling from the set $[n + m]$ for each graph: $\Pi_1(i) = v_i \in G_1$ and $\Pi_2(j) = w_j \in G_2$.

Each choice of direction $\omega \in \mathbb{S}^1$ leads to different merge tree T_i^ω for the same graph G_i, and with each pair (T_1^ω, T_2^ω) we can project the labels as described above. We define a finite set of angles from which computing the topological signatures is sufficient to faithfully represent the data. This set of *critical angles* is the set of normal vectors to all edges in the two graphs. While at the moment this involves computing $O(n^2)$ labeled merge trees, our future work will seek to improve this number. We then integrate the distance $d(T_1^\omega, T_2^\omega)$ over $\mathbb{S}^1$ to obtain the distance between G_1 and G_2. We conjecture that this distance is a semi-metric, in other words, it satisfies finiteness, identity, symmetry, and separability properties, but not the triangle inequality.

3 Application

We use our distance to visualize the structural differences in *Passiflora* leaves. Our data sets consists of 3319 leaves across 40 species. Each data point is constructed of 15 Procrustes-

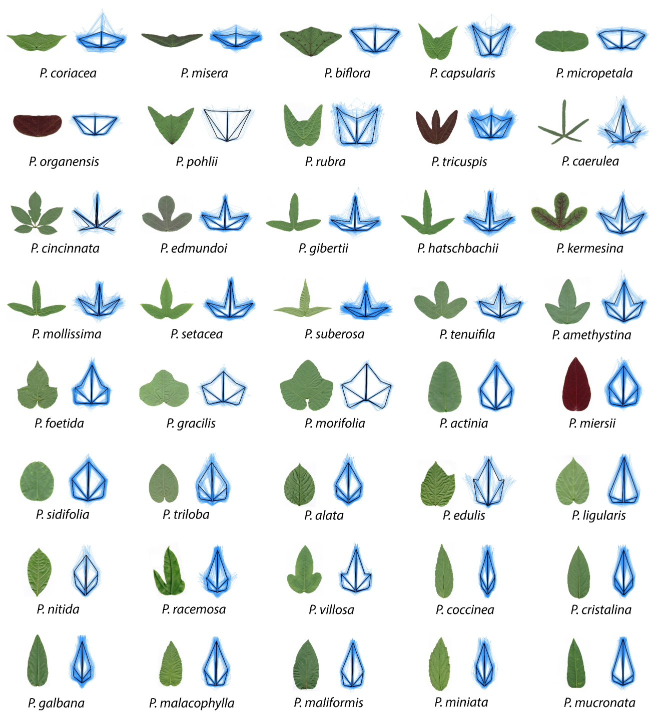


Figure 2 The shapes of *Passiflora* leaves measured using landmarks.

aligned (x, y)-coordinate pairs representing 15 landmarks on each leaf. See Figure 2. We sample one leaf from each of the 216 plants that represents the mature shape of the leaves on that plant. We then compute pairwise distances, constructing a 216×216 distance matrix with the leaves as both the rows and columns. Finally, we use Multi-Dimensional Scaling (MDS) to visualize the relationships between the structures of the leaves. Compared to a traditional method such as Principal Component Analysis, our MDS has a similar global structure but with clearer separation. See Figure 3.

4 Conclusion and Discussion

Our work displays a potentially useful measurement between two embedded graphs. We described its theoretical properties and utilized this for optimizing the implementation. We finally showed its utility by application to a *Passiflora* leaf data set. Our method comes with its own strengths and limitations. While it summarizes the outer shape of the graphs, this distance cannot distinguish internal structures that do not show up in the merge tree. Moreover, in the current form, our method requires the data to be pre-aligned, which is a difficult problem in its own right. Further, our computation, though polynomial, is still relatively expensive; we are looking to further data structures that can provide better performance. Additionally in future work, we aim to understand more theoretical properties of the labeling scheme as non-unique closest points can lead to jumps in the distance, resulting in instability.

References

1 Daniel H Chitwood and Wagner C Otoni. Morphometric analysis of passiflora leaves: the relationship between landmarks of the vasculature and elliptical fourier descriptors of the

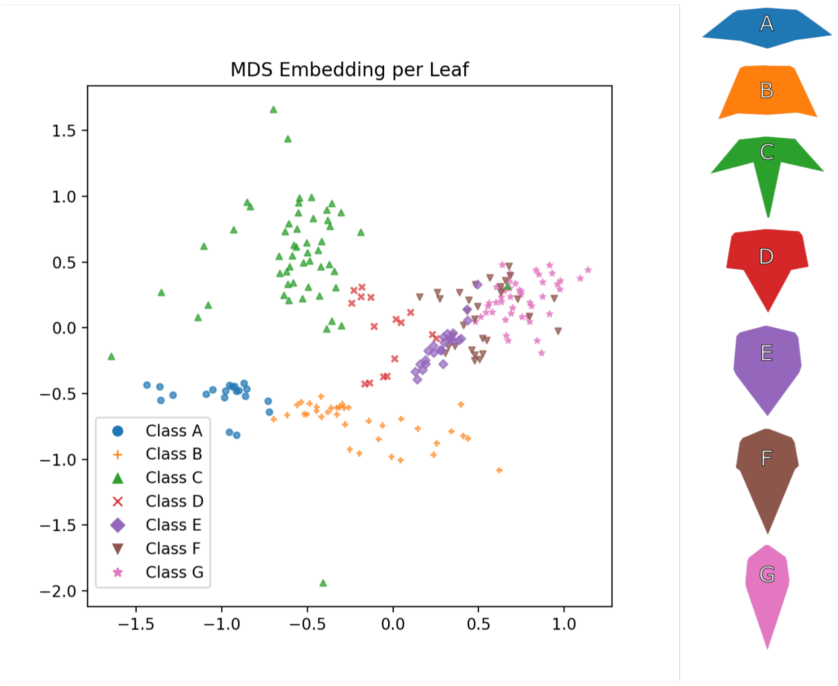


Figure 3 MDS plot with points colored by morphotype.

blade. *Gigascience*, 6(1):1–13, January 2017.
2 Justin Curry, Sayan Mukherjee, and Katharine Turner. How many directions determine a shape and other sufficiency results for two topological transforms. *Transactions of the American Mathematical Society, Series B*, 2018.
3 Ellen Gasparovic, Elizabeth Munch, Steve Oudot, Katharine Turner, Bei Wang, and Yusu Wang. Intrinsic interleaving distance for merge trees. *CoRR*, abs/1908.00063, 2019. URL: `http://arxiv.org/abs/1908.00063`, `arXiv:1908.00063`.
4 Robert Ghrist, Rachel Levanger, and Huy Mai. Persistent homology and Euler integral transforms. *Journal of Applied and Computational Topology*, 2(1):55–60, Oct 2018. `doi:10.1007/s41468-018-0017-1`.
5 Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 01 2013.
6 Elizabeth Munch and Anastasios Stefanou. The l_∞-cophenetic metric for phylogenetic trees as an interleaving distance. In Ellen Gasparovic and Carlotta Domeniconi, editors, *Research in Data Science*, pages 109–127. Springer International Publishing, Cham, 2019. `doi:10.1007/978-3-030-11566-1_5`.
7 Pierre Schapira. Operations on constructible functions. *Journal of Pure and Applied Algebra*, 72(1):83–93, 1991. URL: `https://www.sciencedirect.com/science/article/pii/002240499190131K`, `doi:https://doi.org/10.1016/0022-4049(91)90131-K`.
8 Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 12 2014. `arXiv:https://academic.oup.com/imaiai/article-pdf/3/4/310/2152818/iau011.pdf`, `doi:10.1093/imaiai/iau011`.

Unsmoothing of Contour Trees [*]

Erin Chambers
Saint Louis University

Samira Jabry
Saint Louis University

Elizabeth Munch
Michigan State University

Abstract

In this work, we define an operation on scalar field data which we call the “unsmoothing operator". This operator is essentially an attempt to formulate an inverse operation to the well studied smoothing functor. After introducing, we demonstrate this operation on contour trees, and explore the feasibility of computing it using branch decomposition.

2012 ACM Subject Classification cs.CG

Keywords and phrases Reeb graphs, smoothing, branch decomposition, unsmoothing contour trees

1 Introduction

An $\mathbb{R}$ -space, or a scalar field, is a pair (X, f) in which X is a topological space and $f : X \to \mathbb{R}$ is a continuous, scalar-valued function. Such input spaces are seen across many fields, and hence there is a great need for analysis of their shapes and properties. Reeb graphs are one commonly used tool in topological data analysis for scalar fields; we refer to a recent survey for a more exhaustive discussion of their many uses [9]. Essentially, these are one-dimensional summaries of the connectivity and some of the topology in the input spaces. Given a scalar field (X, f) where X is simply connected, the resulting Reeb graph R_f is simply connected; this results in a *contour trees*. See Figure 1 for an example of a contour tree. We denote the contour tree defined on (X, f) as $\mathbb{C}_f$. In this paper, we draw upon recent work for smoothing a Reeb graph or contour tree [3, 1] to explore the inverse operation: unsmoothing a contour tree. We show that our unsmoothing operation generates a new contour tree $T'_\epsilon(X, f)$ for every $\epsilon > 0$ and simplifies the tree. We then explore one way to compute such unsmoothings using branch decompositions [6, 5, 7, 2, 8], which were introduced in prior work to compute metrics to compare contour trees. Our eventual goal for this investigation is to better understand the nature of vineyards, and in particular determine what types of time varying data sets will allow for Reeb flows. As mentioned previously, earlier work [1] studies this for smoothing, and our work seeks to understand if there is an inverse operation, unsmoothing.

2 Background and definitions

Formally, a Reeb graph is a pair (X, f) where X is topological space and $f : X \to \mathbb{R}$ is a continuous real-valued function. We assume that all Reeb graphs R_f are generic so for that all vertices have degrees 1 or 3 and occur at distinct heights which can appear in this case, with their (down-, up-) degrees specified, are local minima $(0, 1)$, up-forks $(1, 2)$, down-forks $(2, 1)$, and local maxima $(1, 0)$. Note that this assumption is not necessary for correctness, as

[*] This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

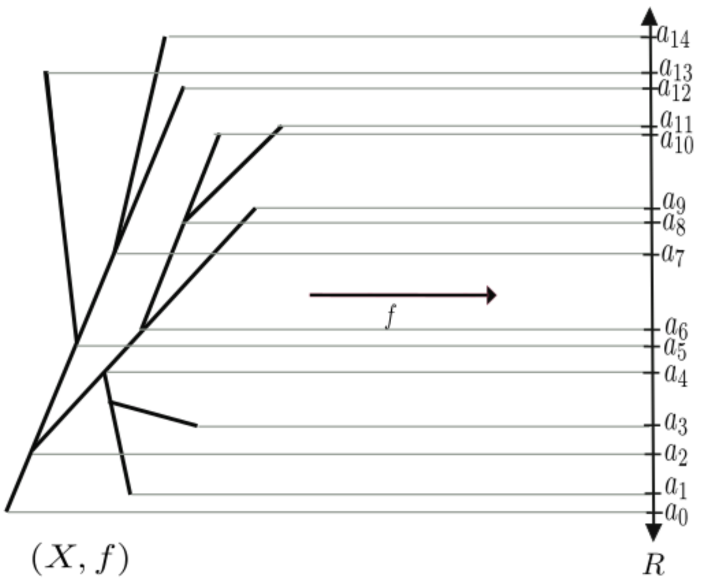


Figure 1 A contour tree (left) with real valued function (right) defined by height.

degenerate Reeb graphs can symbolically be made generic by perturbation: a $(2,2)$ vertex is really a $(2,1)$ vertex that is identified with a $(2,1)$ vertex, for example. However, this assumption greatly simplifies cases and is true in many contexts (generic Morse functions on manifolds, for example), so we keep it for simplicity.

The smoothing operation on Reeb graphs is defined as follows: For $\epsilon \geq 0$, let $(f + Id) : X \times [-\epsilon, \epsilon] \to \mathbb{R}$ be defined as $(x,t) \to f(x) + t$. We define the ϵ-smoothing $\mathrm{S}_\epsilon(\mathrm{X, f})$ to be the Reeb graph of $(X \times [-\epsilon, \epsilon], f + Id)$. Note that there this operator was originally defined and studied through the lens of cosheaves, in the context of defining interleavings for Reeb graphs and proving functorial properties [3]. Our definition is equivalent, but takes a more combinatorial and geometric view of the operation. In the space $X \times [-\epsilon, \epsilon]$, f_ϵ is the induced height function where $f_\epsilon(x,t) = f(x) + t$; we abuse notation slightly and also use f_ϵ to be the induced height function in $S_\epsilon(X, f)$.

Recent work [1] completely characterizes the combinatorial effect of smoothing on a graph, which we briefly describe here. Essentially, there is a natural bijection Φ which takes a subset of the vertices in Reeb graph R_f to vertices in $S_\epsilon(X, f)$. First, cycles disappear (or are "smoothed away") when the length of the cycle is shorter than 2ϵ, where the length of a cycle is the difference in height between the highest and the lowest vertices. We say a vertex u in $V(X)$ undergoes a ϵ-shift upwards if $f(u) + \epsilon = f_\epsilon(\Phi(u))$, and that u undergoes a ϵ-shift downwards if $f(u) - \epsilon = f_\epsilon(\Phi(u))$. The bijection Φ shifts up forks and maximums up by ϵ, and down forks and minimums downwards by ϵ.

Define an *(ascending) branch* in (X, f) to be a path $\gamma : [0,1] \to X$ so that $\gamma(0)$ and $\gamma(1)$ are vertices and $f(\gamma(t))$ is strictly increasing; this is a monotone path in a contour tree traversing a sequence of nodes with no decreasing or (non-increasing) values of f. Define the length of a branch as $f(\gamma(1)) - f(\gamma(0))$. A branch decomposition of a contour tree (X, f) is a hierarchical decomposition of branches into a tree structure which consists of a root branch b that connects two leaves of X, together with, for each component of $X - b$, a branch decomposition whose root branch starts or ends at a (interior) vertex of b. Note that branch

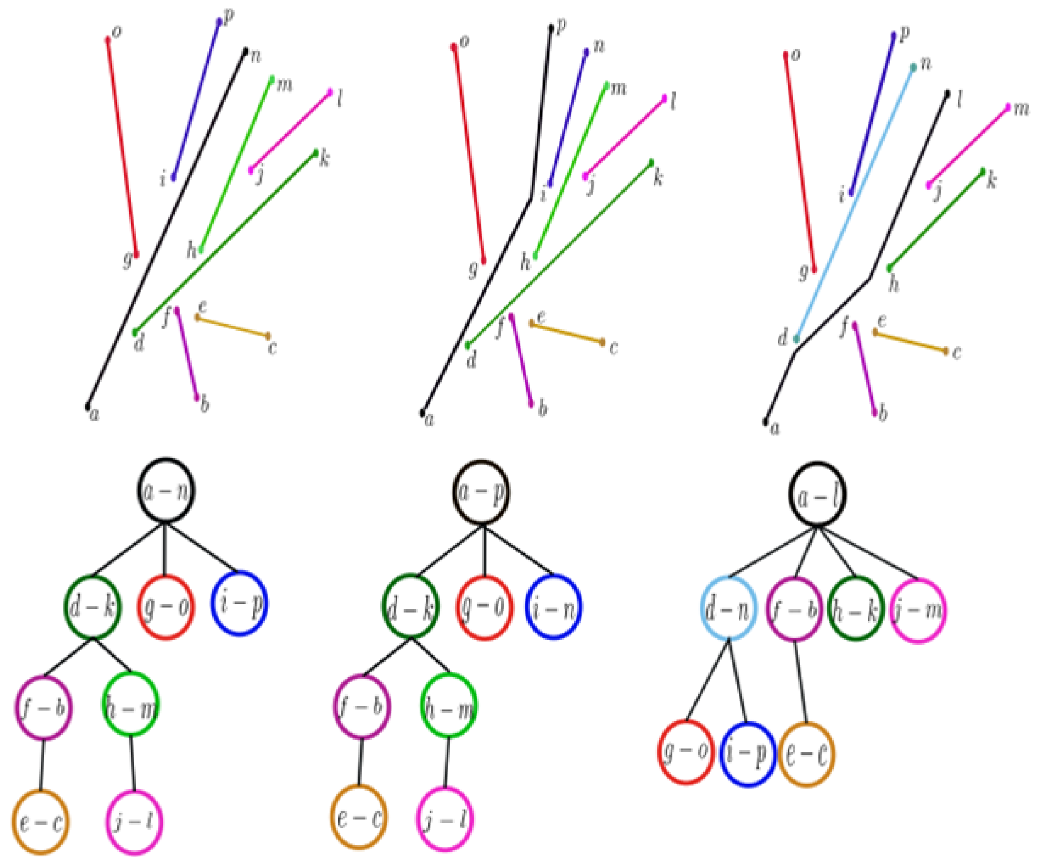


Figure 2 Top: The contour tree from Figure 1 can be segmented into many different possible collections of branches. Bottom: Each such segmentation results in a different branch decomposition tree; every node in the branch decomposition tree corresponds to a path in the original contour tree.

decomposition trees are not unique; for example, if we consider the contour tree in Figure 1, we can build several different branch decomposition trees, as shown, in Figure 2.

3 Unsmoothing contour trees

Finally, we are ready to consider unsmoothing, in effort to better generalize the study of flows on Reeb graphs [4].

We say that (Y, g) is an ϵ-unsmoothing of (X, f) if $S_\epsilon(Y, g) = (X, f)$. We define $\alpha(X, f)$ to be the minimum of the lengths of all ascending branches γ for which $\gamma(0)$ is a local minimum or a down-fork of f, and $\gamma(1)$ an up-fork or a local minimum of f, respectively. See Figure 3. Our main results begin an exploration of when unsmoothing is possible:

▶ **Proposition 1.** *Let (X, f) be a contour tree, and let $\epsilon > 0$.*
- *If $\epsilon \leq \alpha(X, f)/2$ then there exists an ϵ-unsmoothing (Y, g) of (X, f).*
- *If $\epsilon > \alpha(X, f)/2$ then there does not exist an ϵ-unsmoothing of (X, f).*

In the remaining space, we briefly sketch the high-level idea of the proof in our remaining space: we build a branch decomposition that guides how to build a particular unsmoothed contour tree. Each node in the branch decomposition tree is a monotone path in the contour tree; this will correspond to a (possibly longer) path in the unsmoothed tree which

4 Unsmoothing of Contour Trees

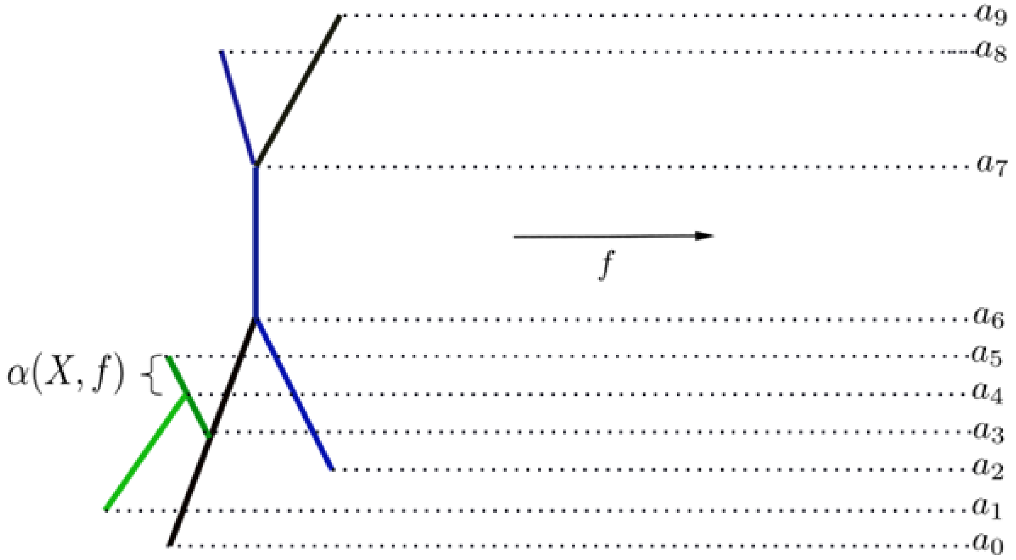


Figure 3 The graph has an up fork and down (green edges), and showing that α is difference of function value.

we construct. By placing vertices for each path at the correct height along their parent branch's path, we can guarantee the existence of paths if $\epsilon < \alpha(X, f)/2$, and demonstrate an obstruction to such a tree's existence when $\epsilon > \alpha(X, f)/2$, as the unsmoothed tree will have combinatorial differences with the original. (Note that when the distance is equal to $\alpha(X, f)/2$, the tree exists but is degenerate.)

This initial result only begins our exploration of unsmoothing. We are also working towards using branch decompositions to construct canonical unsmoothings, which ideally keep functorial properties and hence can be used more broadly for studying the inverse problem on this type of data.

References

1 Rehab Alharbi, Erin Wolf Chambers, and Elizabeth Munch. Realizable piecewise linear paths of persistence diagrams with reeb graphs, 2021.

2 Brian Bollen, Erin Chambers, Joshua A. Levine, and Elizabeth Munch. Reeb graph metrics from the ground up, 2021.

3 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified Reeb graphs. *Discrete & Computational Geometry*, pages 1–53, 2016.

4 Vin de Silva, Elizabeth Munch, and Anastasios Stefanou. Theory of interleavings on categories with a flow. *Theory and Applications of Categories*, 33(21):583–607, 2018.

5 Himangshu Saikia, Hans-Peter Seidel, and Tino Weinkauf. Extended Branch Decomposition Graphs: Structural Comparison of Scalar Data. *Computer Graphics Forum*, 2014.

6 Giorgio Scorzelli, Valerio Pascucci, and Kree Cole-McLaughlin. Multi-resolution computation and presentation of contour trees. 01 2004.

7 Florian Wetzels, Heike Leitte, and Christoph Garth. Branch decomposition-independent edit distances for merge trees. *Computer Graphics Forum*, 41(3):367–378, 2022.

8 Lin Yan, Talha Bin Masood, Farhan Rasheed, Ingrid Hotz, and Bei Wang. Geometry-aware merge tree comparisons for time-varying data with interleaving distances. *CoRR*, abs/2107.14373, 2021.

9 Lin Yan, Talha Bin Masood, Raghavendra Sridharamurthy, Farhan Rasheed, Vijay Natarajan, Ingrid Hotz, and Bei Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *CoRR*, abs/2106.00157, 2021.

Image Shape Classification with the Weighted Euler Curve Transform [*]

Dhanush Giriyan
Arizona State University. United States.

Jessi Cisewski-Kehe
Department of Statistics, University of Wisconsin-Madison, United States.

Brittany Terese Fasy
School of Computing & Dept. of Mathematical Sciences, Montana State University, United States.

Abstract

The weighted Euler curve transform (WECT) was recently introduced as a tool to extract meaningful information from shape data, when the shape is equipped with a weight function. In this extended abstract, we provide an experimental investigation on using the WECT for image classification.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Geometric topology

Keywords and phrases Topological data analysis, Euler characteristic, classification, image data

Funding The authors are supported by a collaborative grant through the National Science Foundation, NSF DMS 1854336 and DMS 2038556.

1 Introduction

One of the primary goals of topological shape analysis is to summarize the shape of data using topological invariants. The *Euler Characteristic* (EC) is a simple invariant that is easily computed from data. For a filtered topological space, this can be extended to an *Euler Characteristic Curve* (ECC). This extended abstract considers a variant of the ECC for a filtered and weighted simplicial complex, called the *Weighted Euler Curve Transform* (WECT) [6].

2 Background

Given a simplicial complex K and a filter function $f\colon K \to \mathbb{R}$, the ECC is the function $\chi_f\colon \mathbb{R} \to \mathbb{Z}$ that maps $x \in \mathbb{R}$ to the EC of the sublevel set $f^{-1}((-\infty, r])$. A common filter function is the *directional filter function*, which, for K embedded in $\mathbb{R}^2$, is the lower-star filter for a direction $s \in \mathbb{S}^1$. Using an ensemble of directional filter functions, one for each direction in $\mathbb{S}^1$, we obtain a parameterized collection of ECCs, called the *Euler curve transform (ECT)*.[1] No two distinct shapes have the same ECT [9]. Since its introduction, the ECT has developed both in theory and in practice [1, 3, 5–7].

The *WECT* is a generalization of the ECT for weighted complexes [6]. Let K be a simplicial complex in $\mathbb{R}^2$ and let $g\colon K \to \mathbb{R}$ be its weight function (g need not be a filter

[*] This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

[1] The ECT can be defined more generally; however, for brevity, we restrict the definition here to our specific problem setting.

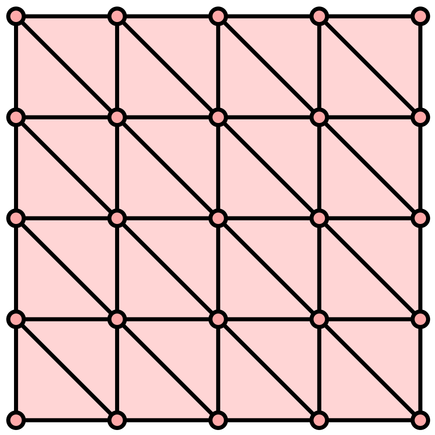


Figure 1 Freudenthal Triangulation of 5×5 image.

function). The WECT for K assigns to each $s \in \mathbb{S}^1$ the function $\omega_{s,g} \colon \mathbb{R} \to \mathbb{Z}$ defined by:

$$\omega_{s,g}(t) = \sum_{d=0}^{2}(-1)^d \sum_{\substack{\sigma \in K_d \\ h_s(\sigma) \leq t}} g(\sigma) = \sum_{\substack{\sigma \in K \\ h_s(\sigma) \leq t}} (-1)^{\dim(\sigma)} g(\sigma),$$

where $h_s \colon K \to \mathbb{R}$ is the lower-star filter in direction s.

3 Experiments

3.1 Data Sets

Synthetic image datasets are generated using the following parameters: shape, pixel intensity, and function extension. The domain of an image is a 64×64 grid, with vertices on an integer lattice, triangulated using the Freudenthal triangulation [4]; see Figure 1. The *shapes* are the domain of the image pixels with positive intensity. Two shapes are considered: the disc (D) and the square (S). The disc D is centered at $(32, 32)$ with a radius of 20 pixels. The square S is also centered at $(32, 32)$ and has an edge length of 35, ensuring that both D and S have approximately the same number of pixels. Let K be the triangulation of either D or S.

Pixel intensities are sampled from Gaussian (G) and uniform (U) density shapes. A bivariate Gaussian density with a peak at the middle of the image and diagonal covariance matrix with standard deviations 100 is used. Points are sampled from this distribution and then assigned to the nearest grid cell to create a 2D histogram. This histogram is rescaled so that the maximum intensity is 255. In the uniform distribution, each pixel intensity is sampled from $[0, 255]$. In what follows, $g \colon K_0 \to \mathbb{R}$ is the function on the vertices of K.

Together, the shape and distribution define four image classes: square-Gaussian (SG), square-uniform (SU), disc-Gaussian (DG), and disc-uniform(DU), with 250 sampled images in each; see Figure 2. Given an image (a function $g \colon K_0 \to \mathbb{R}$), we consider two extensions (functions $F \colon K \to \mathbb{R}$). First, the maximum extension is defined by mapping a simplex to the maximum value on its vertices. Second, the average extension takes average instead of the maximum. For each image class, we test both function extensions, creating eight data sets. We refer to an image, along with its extension to all simplices an *extended image*.

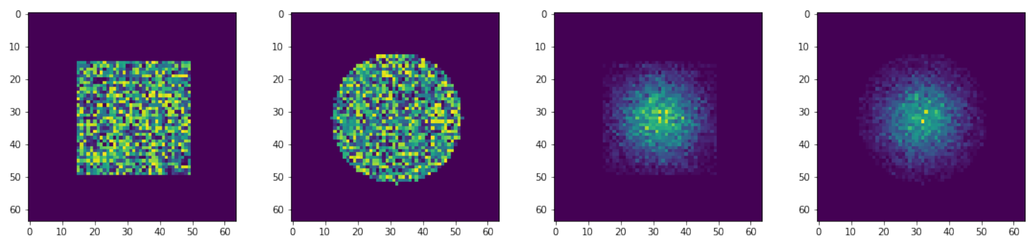

(a) SU **(b)** DU **(c)** SG **(d)** DG

Figure 2 Four image classes.

Table 1 Pairwise classification results with maximum weight function extension.

	SG	SU	DG	DU
SG	0.500 ± 0.000			
SU	1.000 ± 0.000	0.500 ± 0.000		
DG	0.902 ± 0.023	0.998 ± 0.006	0.500 ± 0.000	
DU	1.000 ± 0.000	1.000 ± 0.000	0.998 ± 0.006	0.500 ± 0.000

3.2 Methods

We evaluate the WECT on the task of binary classification. Given two data sets, we first compute the WECT for each extended image. To approximate the WECT (a function over $\mathbb{S}^1$), we sample eight equally-spaced directions from $\mathbb{S}^1$, which was sufficient given the simplicity of the image features. To discretize each WECC, 50 threshold values are sampled, thus each image results in a vectorized WECT of length 400. Next, we train a Support Vector Classifier (SVC) [2] separately for each binary classification model. The set of WECTs is partitioned into training and test sets, with an 80/20 train-test split. All SVCs in this work use a regularization parameter of value 20. This was implemented in Python, using data structures from the Dionysus 2 library [8]. Experiments were conducted on an Intel Xeon E5-2680 (128 GB RAM).

3.3 Results and Conclusion

For each pair of image classes, we compute the WECTs for both the maximum and the average function extensions. Then, we fit SVC models on 10 unique collections of 250 images of each class and report the average accuracies and the standard errors. Table 1 and Table 2 provide the percentage of correctly classified test images.

In the simple data sets that we considered, we found the WECT to be suitable for the task of binary classification. The weakest classification result was comparing the SG and DG classes under the average weight function extension. Because Gaussian density decreases

Table 2 Pairwise classification results with average weight function extension.

	SG	SU	DG	DU
SG	0.500 ± 0.000			
SU	1.000 ± 0.000	0.500 ± 0.000		
DG	0.830 ± 0.048	1.000 ± 0.000	0.500 ± 0.000	
DU	1.000 ± 0.000	0.948 ± 0.034	1.000 ± 0.000	0.500 ± 0.000

toward zero in the extremes, the corners of SG have weights that are close to zero making SG resemble the DG. Why the maximum weight function extension performed better than the average is not immediately clear. We plan to extend the set of shapes, distributions, and function extensions in order to explore this observation.

References

1 Eric Berry, Yen-Chi Chen, Jessi Cisewski-Kehe, and Brittany Terese Fasy. Functional summaries of persistence diagrams. *Journal of Applied and Computational Topology*, 4(2):211–262, 2020.

2 Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

3 Lorin Crawford, Anthea Monod, Andrew X Chen, Sayan Mukherjee, and Raúl Rabadán. Predicting clinical outcomes in glioblastoma: an application of topological and functional data analysis. *Journal of the American Statistical Association*, 115(531):1139–1150, 2020.

4 Hans Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, pages 580–582, 1942.

5 Robert Ghrist, Rachel Levanger, and Huy Mai. Persistent homology and Euler integral transforms. *Journal of Applied and Computational Topology*, 2(1-2):55–60, 2018.

6 Qitong Jiang, Sebastian Kurtek, and Tom Needham. The weighted Euler curve transform for shape and image analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 844–845, 2020.

7 Lewis Marsh, Felix Y Zhou, Xiao Quin, Xin Lu, Helen M Byrne, and Heather A Harrington. Detecting temporal shape changes with the euler characteristic transform, 2022. arXiv preprint arXiv:2212.10883.

8 Dmitriy Morozov. Dionysus 2. `https://mrzv.org/software/dionysus2/`.

9 Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.

Revisiting the Fréchet distance between piecewise smooth curves[*]

Jacobus Conradi
Department of Computer Science, University of Bonn, Germany

Anne Driemel
Hausdorff Center for Mathematics, University of Bonn, Germany

Benedikt Kolbe
Hausdorff Center for Mathematics, University of Bonn, Germany

Abstract

We extend algorithmic tools for the computation of the Fréchet distance for polygonal curves to that of piecewise smooth curves in $\mathbb{R}^d$. We show that the decision problem can be solved in $O(mn)$ time, for two curves consisting of m, resp., n smooth algebraic curves, of uniformly bounded degree.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Fréchet distance, Simplification, Smooth curves, Approximation algorithms

1 Introduction

The wealth of work surrounding the analysis of algorithms for computing the Fréchet distance is centered primarily on polygonal curves. However, more complicated curves and especially splines have become commonplace in industrial applications for, e.g., computer graphics, robotics and to represent motion tracking data. In such applications, the number of dimensions corresponds to the number of tracked parameters, leading to a high-dimensional ambient space. Devising algorithms for the computation of the Fréchet distance between smooth curves in $\mathbb{R}^d$ is a natural and fundamental task in computational geometry. As far as we know, there is no known approach to the computation of $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2)$ for smooth curves in $\mathbb{R}^d$. For smooth curves in the plane ($d = 2$), Rote [3] showed how to do this. However, his approach does not easily generalize to higher dimensions. We revisit this problem and present a new, simpler approach with the same time complexity that works for higher dimensions.

Problem definition Throughout the paper, γ_1 and γ_2 will be used to denote two piecewise smooth curves in $\mathbb{R}^d$ with d fixed, that is, continuous maps $\gamma_1, \gamma_2 : [0,1] \to \mathbb{R}^d$ that are comprised of m and n smooth pieces, each of class C^2. Let $A_{[0,1]}$ be the set of continuous and bijective maps $\alpha : [0,1] \to [0,1]$ that are increasing. For two continuous curves γ_1 and γ_2, the **Fréchet distance** is defined as $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) := \inf_{\alpha,\beta \in A_{[0,1]}} \max_{t\in[0,1]} \|\gamma_1(\alpha(t)) - \gamma_2(\beta(t))\|$. We consider different choices of ℓ_p norm for the norm $\|\cdot\|$.

Results Our main contribution is that we establish a solution to the decision problem of determining whether $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) \le \delta$ for a given $\delta > 0$ and a fixed ambient space $\mathbb{R}^d$. Assuming that the curves are algebraically bounded curves, i.e., piecewise algebraic curves where the maximal degree of the curves is bounded by a constant, our algorithm runs in $O(mn)$ time. The solution of the decision problem forms the basis for the framework for algorithms for computing the Fréchet distance, similarly to the polygonal case.

[*] This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

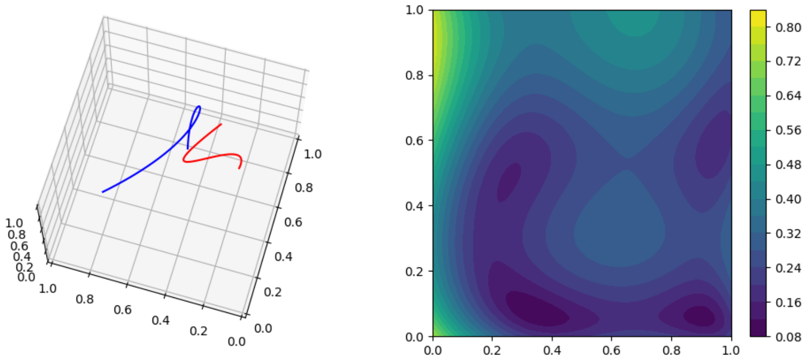


Figure 1 Two smooth curves in $\mathbb{R}^3$ and a contour plot of the associated free space diagram.

Methods and difficulties Similarly to the classical polygonal case, to solve the decision problem, we ascertain the existence of a path from $(0,0)$ to $(1,1)$ in the **free space** $\mathcal{D}_\delta$ that is monotone (in both coordinates). Unlike the polygonal case, the free space $\mathcal{D}_\delta$ can be very complicated (Figure 1 and 4), so we propose a decomposition of the **free space diagram** FSD_δ into a controlled number of pieces, for which the existence of a monotone path connecting two intervals on edges can be read off. Here, monotonicity of the boundary curves replaces the role of convexity of the free space for polygonal curves. A key technical result we show is that the boundary of $\mathcal{D}_\delta$ in FSD_δ is smooth (almost everywhere), for most values of δ and choices of ℓ_p norm in the definition of the Fréchet distance.

To arrive at an algorithm for the decision problem for smooth planar curves, Rote also uses a decomposition of FSD_δ to obtain pieces that are easier to handle. His approach relies on analyzing the curves γ_1 and γ_2 directly, introducing cuts at points with a certain value for the curvature, and to control the turning angle. In contrast to this, instead of working with the curves directly, our approach is based on analyzing the free diagram FSD_δ.

2 A combinatorial description of the free space diagram

Consider the boundary B_δ of $\mathcal{D}_\delta$. We treat B_δ as a set of curves, as opposed to the boundary of a region and define the three sets E_h, E_v and I_s of points on B_δ, as illustrated in Figure 2:

1. The set E_h of points in B_δ that have a horizontal tangent to the free space,
2. The set E_v of points in B_δ that have a vertical tangent to the free space,
3. The set I_s of singularities of B_δ, consisting of points where B_δ is non-smooth.

It turns out that I_s is empty for most values of δ and ℓ_p norms with $1 < p < \infty$.Furthermore, for algebraically bounded curves, the number of values for δ for which I_s is non-empty is bounded by a constant depending on the maximal degree of the curves.Figure 3 illustrates the effect of a small perturbation of δ on the free space in FSD_δ. Assume a δ and ℓ_p norm where I_s is empty and fix a cell C of FSD_δ. For every point $x \in C$ in E_h (E_v), trace the vertical (horizontal) line inside C. The result is a refinement of the partitioning FSD_δ into a collection $\{C_{i,j}\}_{i,j}$ of smaller cells. Figure 4 depicts the ensuing decomposition within a cell of FSD_δ. The decomposition $\{C_{i,j}\}$ has the property that B_δ is a collection of monotone arcs within each cell $C_{i,j}$ and that B_δ is essentially uniquely determined by its intersections with $C_{i,j}$.

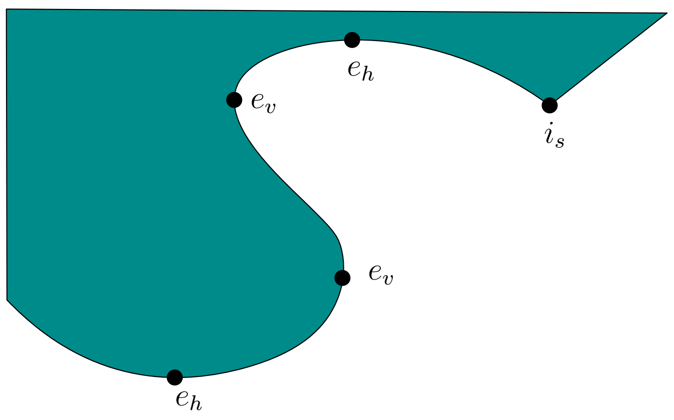


Figure 2 Illustration of points $e_v \in E_v$, $e_h \in E_h$, and $i_s \in I_s$ for a region of FSD$_\delta$.

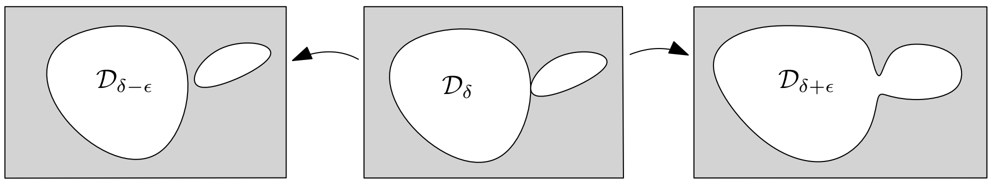

$\mathcal{D}_{\delta-\epsilon}$
$\mathcal{D}_{\delta}$
$\mathcal{D}_{\delta+\epsilon}$

Figure 3 Illustration of singular points and changes of the boundary as δ changes.

Decision problem To answer the decision problem, we examine the **reachable space** in FSD$_\delta$ (or $\{C_{i,j}\}$), which are those points reachable by a monotone path in the free space, starting from $(0,0)$. By monotonicity of B_δ in every cell in $\{C_{i,j}\}$, there is at most one interval on each of the top and right edges of a cell that are reachable by a monotone path starting from the left (or bottom) edge of that cell, and the points reachable by monotone paths within each cell can be readily identified. We compute the intervals on the top and right edges reachable from the left-most (and bottom) point of a reachable interval on the bottom (and left) edges inductively. The set-up resembles that of Alt and Godau [1] in the polygonal case, where each cell wall has at most one reachable interval, in contrast to the situation for smooth curves. Similarly to, e.g., Driemel, Har-Peled and Wenk [2], we use a BFS on grid cells to compute a representation as a graph of the intervals on edges of cells reachable by a monotone path. Each node in the graph represents a maximal reachable interval on the edges of a cell, and these are joined by an edge if they belong to the same cell and there is a monotone path in the cell from one interval to the other. One can show that there is at most a constant number of reachable intervals in each reachable cell, so we can proceed similarly to [2, Lemma 3.1] to obtain the following result.

▶ **Proposition 1.** *Given two algebraically bounded piecewise smooth curves γ_1, γ_2 in $\mathbb{R}^d$ comprised of m and n pieces, and a value of $\delta \geq 0$, one can decide if $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) \leq \delta$. The running time is bounded by $O(mn)$.*

In practice, the performance of the algorithm for the decision problem will depend on the dimension d of the ambient space of the curves. Note that this dependence stems from the algebraic complexity of the operations used to execute the algorithm. On the other hand, the number of cells in our decomposition of FSD$_{delta}$ depends on the degree of the

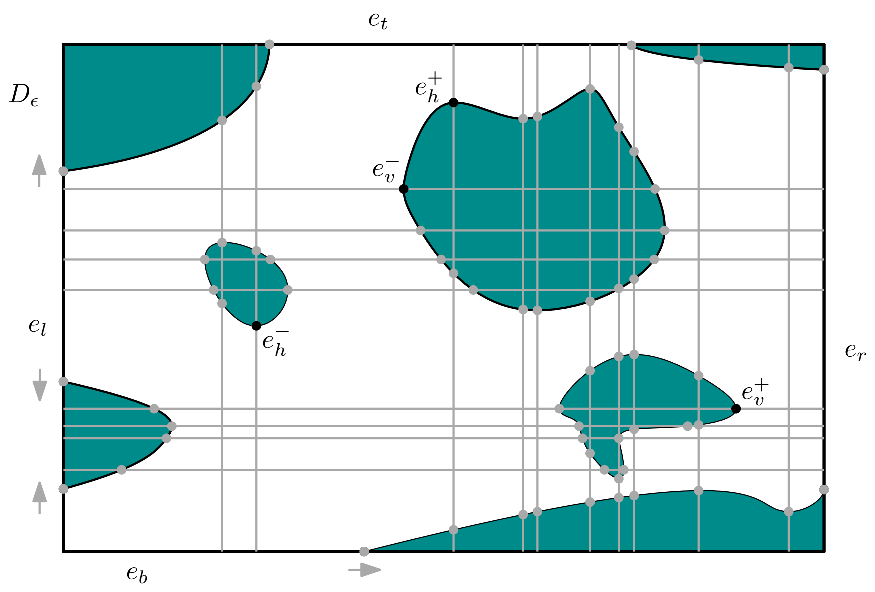


Figure 4 Refinement of a cell of the free space diagram into (sub)cells arising from the horizontal and vertical lines at extremities of the forbidden region in the coordinate directions.

considered set of algebraically bounded curves, so while the operations themselves become more complicated, the bound on the time complexity in Proposition 1 is independent of the dimension.

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 05(01n02):75–91, mar 1995.
2 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete and Computational Geometry*, 48(1):94–127, 2012.
3 Günter Rote. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry: Theory and Applications*, 37(3 SPEC. ISS.):162–174, aug 2007.

Subtrajectory Clustering for Human Motion Data*

Jacobus Conradi
Department of Computer Science, University of Bonn, Germany

Anne Driemel
Hausdorff Center for Mathematics, University of Bonn, Germany

Abstract

We extend the subtrajectory clustering approach by Brüning, Conradi and Driemel [3] to center curves consisting of more than a single edge, and provide evidence for its practical viability using full-body motion data.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Clustering, Set cover, Fréchet distance, Approximation algorithms

Supplementary Material Implementation can be found in [6].

1 Introduction

Advancements in motion tracking technology made it possible to track motion data from many different areas ranging from human motion data to animal migration to vehicle navigation. With vast amounts of data one is often interested to find suitable representatives of the given data reducing the size of the data considerably. We consider the subtrajectory clustering approach introduced by Akitaya, Brüning, Chambers and Driemel [2], extended upon by Brüning, Conradi and Driemel [3]. Similar approaches to subtrajectory clustering have been pursued by Buchin et al. [4], Agarwal et al. [1], Buchin et al. [5], Gudmundsson et al. [7] and most recently Schaefer et al. [10]. Here, we consider a given polygonal curve P parametrized over $[0,1]$ together with a distance threshold Δ and complexity bound l. The problem is to find a set of curves $\mathcal{C} = \{C_1, \dots, C_k\}$, each of complexity at most l, such that every point $P(p)$ for $p \in [0,1]$ is contained in a subcurve C_p of P and there is a curve $C_i \in \mathcal{C}$ that has continuous Fréchet distance $\mathrm{d}_{\mathcal{F}}(C_p, C_i) \leq \Delta$. The continuous ***Fréchet distance*** is defined on any two parametrized curves $P, Q : [0,1] \to \mathbb{R}^d$ as $\inf_{f,g} \max_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|$, where f and g—refered to as ***traversals***—range over all non-decreasing surjective functions from $[0,1]$ to $[0,1]$. We denote by $P[s,t]$ the ***subcurve*** of P starting at $P(s)$ and ending at $P(t)$ for $0 \leq s \leq t \leq 1$. A curve C that has Fréchet distance Δ to some such $P[s,t]$ covers every point $p \in [s,t]$. For any curve C in $\mathbb{R}^d$ we define

$$\mathrm{Cov}_P(C,\Delta) = \left(\bigcup_{0\leq s\leq t\leq 1,\, \mathrm{d}_{\mathcal{F}}(P[s,t],C)\leq\Delta} [s,t] \right) \subset [0,1]$$

as the Δ-***coverage*** of C. This subtrajectory clustering problem is naturally interpreted as a set-cover problem, where $[0,1]$ is the ground set and the sets are $\mathrm{Cov}_P(C,\Delta)$ for any curve C in $\mathbb{R}^d$ of complexity l. Brüning et al. [3] consider this problem as a bicriteria-approximation problem. Given an input of a curve P of complexity n and parameters $\Delta > 0$ and $l = 2$, they give an algorithm which—assuming there exists a set of line segments of size k^* such that the union of their Δ-coverages is $[0,1]$—produces a set of line segments of size $O(k^* \log k^*)$

* This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

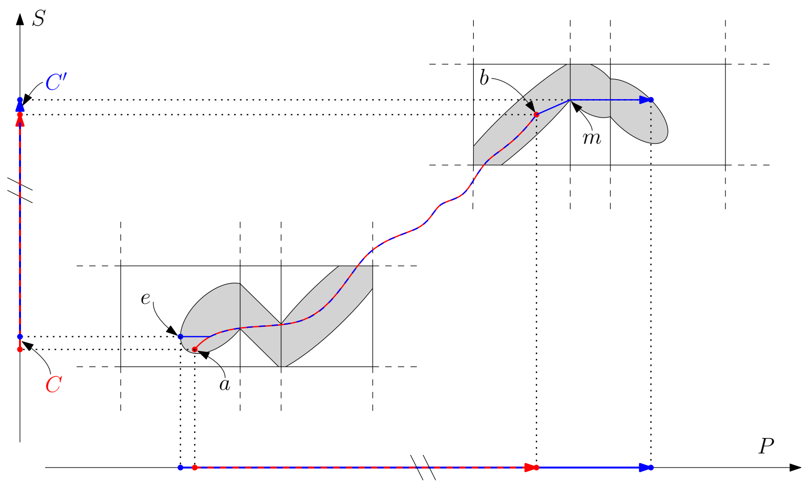


Figure 1 Example of how starting from a traversal—starting in a and ending in b—matching C to some piece of P (all in red), extending it to some curve C' matching an inclusion-wise larger piece of P (all in blue) induced by an extremal point e and a monotonicity point m.

such that the union of their (11Δ)-coverages is $[0, 1]$. They extend this result to arbitrary l, yielding a set of size $O(lk^* \log(lk^*))$. A drawback of this approach is that for any l it exclusively produces line segments as cluster centers, even if more complex centers are sought.

In Section 2 we present a set of candidate curves of size $O(n^3 l)$ containing a similarly good set of center curves, but using curves of higher complexity. This greatly improves the quality of the solution, as we are no longer restricted to line segments as center curves. Our approach heavily builds upon [3]. In particular, we also use so-called ***Δ-good*** simplifications of P. Omitted proofs can be found in the full version. In Section 3 we demonstrate the practical viability of our approach in an application to full-body motion trajectories.

2 Our Candidate Set

▶ **Lemma 1.** *Let a polygonal curve P and values $\Delta > 0$ and l be given. Let S be a Δ-good simplification of P. Let $\mathcal{C}$ be a set of curves of size k each with complexity at most l such that $\bigcup_{C\in\mathcal{C}} \mathrm{Cov}_P(C, \Delta) = [0, 1]$. Then there is a set $\mathcal{C}_S$ of size $3k$ of subcurves of S each of which has complexity at most l with $\bigcup_{C\in\mathcal{C}_S} \mathrm{Cov}_S(C, 8\Delta) = [0, 1]$.*

Brüning et al. [3] showed that under the special assumption $l = 2$ a set $\mathcal{C}_S$ resulting via Lemma 1 from any solution $\mathcal{C}$ for given polygonal curve P and $\Delta > 0$ can be transformed to a set of curves which each start and end in so called extremal or monotonicity points of the 8Δ-free space of S with itself. The ***Δ-free space*** of two curves P and Q is defined as the sublevel-set of pointwise distances less than Δ, i.e., $D_\Delta(P, Q) = \{(x, y) \mid \|P(x) - Q(y)\| \le \Delta\}$. Monotone paths (in x and y) in the Δ-free space correspond to subcurves of P and Q with Fréchet distance at most Δ. ***Extremal points*** are points in the Δ-free space that are local maxima in the x-direction. ***Monotonicity points*** are local minima in the y-direction.

▶ **Lemma 2.** *Let a polygonal curve P and $\Delta > 0$ together with l be given. Let further a Δ-good simplification S together with a subcurve C of S of complexity l be given. Then there are four subcurves $C_1, \dots, C_4$ of S, each of complexity l, starting and ending in extremal or monotonicity points of $D_{8\Delta}(S, S)$ such that $\mathrm{Cov}_S(C, 8\Delta) \subset \bigcup_{i=1}^{4} \mathrm{Cov}_S(C_i, 8\Delta)$.*

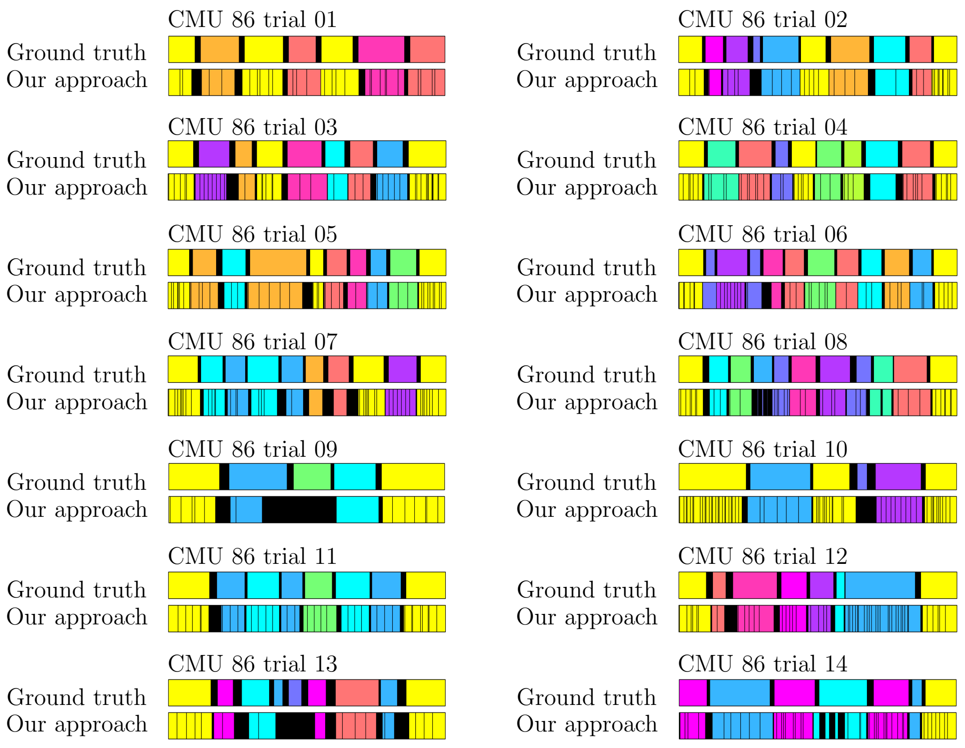


Figure 2 Segmentation result of subtrajectory clustering on CMU data, black indicates ambiguity.

Proof Sketch. We observe in the free-space diagram that a piece of S that is close to C can be extended by moving the start (resp. end-point) of their traversal to the left (resp. right). We consider four cases. In each case, we move the startpoint (resp. endpoint) of the traversal up or down, modifying C until we start in an extremal or monotonicity point (Figure 1). ◀

▶ **Theorem 3.** *For a given polygonal curve P of complexity n and values $\Delta > 0$ and l there is a set of curves B of complexity l with $|B| = O(n^3 l)$ such that for any set of curves $\mathcal{C}$ of complexity at most l with $\bigcup_{C\in\mathcal{C}} \mathrm{Cov}_P(C,\Delta) = [0,1]$ there is a set of curves $\mathcal{C}_B \subset B$ of size $12|\mathcal{C}|$ such that $\bigcup_{C\in\mathcal{C}_B} \mathrm{Cov}_P(C, 11\Delta) = [0,1]$.*

Proof. First simplify P resulting in a Δ-good simplification S of P (as in [3]). The complexity of S is bounded by n. The 8Δ-free space of S with itself has n rows. Each row has at most $O(n)$ extremal or monotonicity points acting as start points of candidates. The l rows above any row have a total of $O(nl)$ extremal or monotonicity points acting as possible endpoints defining the set B. Thus the size of B clearly is in $O(n^3 l)$. The correctness follows by Lemma 1 and Lemma 2 together with the fact that $\mathrm{d}_{\mathcal{F}}(P,S) \leq 3\Delta$. ◀

3 Human Motion Data

The candidate set of Theorem 3 induces a set cover instance $\mathcal{I}$ in the following way. Each candidate induces a subset of $[0,1]$ by its coverage, which is a set of $\mathcal{I}$. Applying the greedy set cover algorithm yields a bicriteral approximation to the subtrajectory clustering problem. We implemented this algorithm and applied it to the CMU data-set [9] (restricted to subject 86). Each trial is a 93-dimensional curve that tracks the positions of the joints (full-body

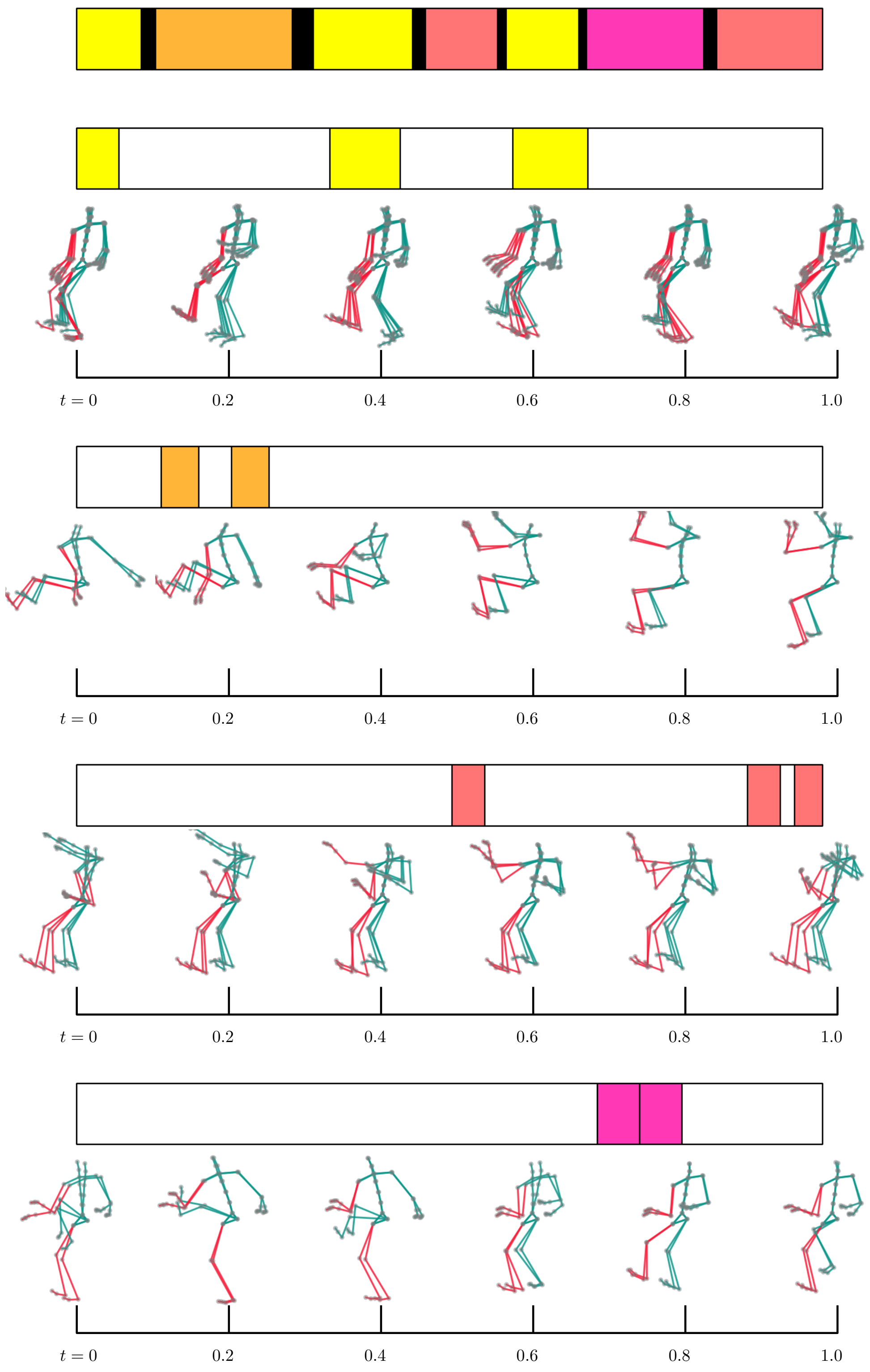


Figure 3 Motion data and labeled coverage of first four clusters of CMU 86 trial 01.

motion). The source code is publicly available at [6]. We compare the resulting segmentations to ground truth data from [8]. The results are depicted in Figure 2. Colors are chosen by matching to the ground truth. Overlaps between differently colored sets are shown in black. Figure 3 visualizes maximal subtrajectories contained in the first four sets chosen by the greedy set cover algorithm by superimposition of stills.

References

1 Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '18, page 75–87, 2018. `doi:10.1145/3196959.3196972`.

2 Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Subtrajectory clustering: Finding set covers for set systems of subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, Feb. 2023. `doi:10.57717/cgt.v2i1.7`.

3 Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ESA.2022.28`.

4 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(03):253–282, 2011.

5 Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020.

6 Jacobus Conradi. Github repository. `https://github.com/JacobusTheSecond/clustering/tree/YRF`, 2023.

7 Joachim Gudmundsson and Sampson Wong. Cubic upper and lower bounds for subtrajectory clustering under the continuous fréchet distance. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 173–189. SIAM, 2022. `doi:10.1137/1.9781611977073.9`.

8 Björn Krüger, Anna Vögele, Tobias Willig, Angela Yao, Reinhard Klein, and Andreas Weber. Efficient unsupervised temporal segmentation of motion data. *IEEE Transactions on Multimedia*, 19(4):797–812, 2017. `doi:10.1109/TMM.2016.2635030`.

9 C MoCap. Carnegie mellon university graphics lab motion capture database, 2007. URL: `http://mocap.cs.cmu.edu/`.

10 Peter Schaefer, Nils Rodrigues, Daniel Weiskopf, and Sabine Storandt. Group diagrams for simplified representation of scanpaths. In *Proceedings of the 15th International Symposium on Visual Information Communication and Interaction*, VINCI '22, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3554944.3554971`.

On the number of iterations of the DBA algorithm

Frederik Brüning
Department of Computer Science, University of Bonn, Germany

Anne Driemel
Hausdorff Center for Mathematics, University of Bonn, Germany

Alperen Ergür
The University of Texas at San Antonio, partially supported by NSF CCF 2110075

Heiko Röglin
Department of Computer Science, University of Bonn, Germany

Abstract

DTW Barycenter Averaging (DBA) is a widely used algorithm for estimating the mean of a given set of point sequences. We study DBA in the model of smoothed analysis, upper-bounding the expected number of iterations in the worst case under random perturbations of the input. We assume the algorithm is given n sequences of m points in $\mathbb{R}^d$ and a parameter k that specifies the length of the mean sequence. Our smoothed upper bound is polynomial in k, n and d, and for constant n, also polynomial in m (in contrast to our deterministic lower bound that is exponential in k for $n = 2$).

2012 ACM Subject Classification Theory of computation $\to$ Design and analysis of algorithms

Keywords and phrases Dynamic Time Warping, Smoothed Analysis, Time Series, Clustering

1 Introduction

The DTW Barycenter Averaging (DBA) algorithm [2] was introduced by Petitjean, Ketterlin and Gançarski in 2011 and has since been used in many applications for clustering time series data (e.g. [3, 4, 5]). The objective is to find a representative time series that optimally summarizes a given set of input time series. Here, the optimality is measured using the sum of dynamic time warping (DTW) distances to the input sequences. In this paper, we study the number of iterations that DBA performs until convergence. We follow a line of thought that has proved successful for the closely related k-means algorithm by Stuart Lloyd [1].

Preliminaries We denote with $\|.\|$ the standard Euclidean norm. For $m_1, m_2 \in \mathbb{N}$, each sequence $(1,1) = (i_1, j_1), (i_2, j_2), \dots, (i_M, j_M) = (m_1, m_2)$ such that $i_k - i_{k-1}$ and $j_k - j_{k-1}$ are either 0 or 1 for all k is a ***warping path*** from $(1,1)$ to (m_1, m_2). We denote with $\mathcal{W}_{m_1,m_2}$ the set of all warping paths from $(1,1)$ to (m_1, m_2). For any two point sequences $\gamma_1 = (\gamma_{1,1}, \dots, \gamma_{1,m_1}) \in (\mathbb{R}^d)^{m_1}$ and $\gamma_2 = (\gamma_{2,1}, \dots, \gamma_{2,m_2}) \in (\mathbb{R}^d)^{m_2}$, we also write $\mathcal{W}_{\gamma_1,\gamma_2} = \mathcal{W}_{m_1,m_2}$, and call elements of $\mathcal{W}_{\gamma_1,\gamma_2}$ warping paths between γ_1 and γ_2. The ***dynamic time warping distance*** between the point sequences γ_1 and γ_2 is defined as

$$d_{\mathrm{DTW}}(\gamma_1, \gamma_2) = \min_{w \in \mathcal{W}_{\gamma_1,\gamma_2}} \sum_{(i,j) \in w} \|\gamma_{1,i} - \gamma_{2,j}\|^2$$

A warping path that attains the above minimum is called an ***optimal warping path*** between γ_1 and γ_2. Let $X = \{\gamma_1, \dots, \gamma_n\} \subset (\mathbb{R}^d)^m$ be a set of n point sequences of length m and $C \in (\mathbb{R}^d)^k$ be a point sequence of length k. We call a sequence π of tuples (p, c) where p is an element of some $\gamma \in X$ and c is an element of C an ***assignment map*** between X and C.

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

1.1 The DBA Algorithm

Let X be a set of n point sequences $\gamma_1, \dots, \gamma_n \in (\mathbb{R}^d)^m$. Let $C \in (\mathbb{R}^d)^k$ be another point sequence and $w^{(1)}, \dots, w^{(n)} \in \mathcal{W}_{m,k}$ be chosen such that $w^{(i)}$ is an optimal warping path between γ_i and C. Then $w^{(1)}, \dots, w^{(n)}$ define an assignment map π between X and C. The assignment map π can be represented by sets $S_1(\pi), S_2(\pi), \dots, S_k(\pi)$, where

$$S_i(\pi) = \cup_{j=1}^{n} \{\gamma_{j,t} \mid (i,t) \in w^{(j)}\}.$$

By construction, π minimizes the DTW distances between $\gamma_1, \dots, \gamma_n$ and C. In the opposite direction, the following sequence C_π minimizes the sum of squared distances for fixed π.

$$C_\pi = (c_1(\pi), c_2(\pi), c_3(\pi), \dots, c_k(\pi)) \text{ where } c_i(\pi) := \frac{1}{|S_i(\pi)|} \sum_{p \in S_i(\pi)} p.$$

DBA alternately computes such assignment maps and average point sequences as follows.

1. Let π_0 be an initial assignment map (e.g. randomly drawn $w_0^{(1)}, \dots, w_0^{(n)}$). Let $j \leftarrow 0$.
2. Let $j \leftarrow j+1$. Compute the average point sequence $C_{\pi_{j-1}}$ based on π_{j-1}.
3. Compute optimal warping paths $w_j^{(i)}$ between γ_i and $C_{\pi_{j-1}}$ for all $1 \le i \le n$. The warping paths define an assignment map π_j between X and $C_{\pi_{j-1}}$.
4. If sum of DTW distances is lower for π_j than for π_{j-1}, go to Step 2. Otherwise, terminate.

2 Upper bound based on geometric assumptions on the input data

We can upper bound the number of iterations of DBA based on the following geometric assumptions on the input data.
Normalization Assumption: We assume the input data is normalized so that for any vector y in any input point sequence we have $\|y\|^2 \le B$.
Separation Assumption: For any two different assignment maps $\alpha, \beta \in \mathcal{A}_{n,m,k}$ we have

$$\|C_\alpha - C_\beta\|^2 := \sum_{i=1}^{k} (c_i(\alpha) - c_i(\beta))^2 \ge \varepsilon$$

The separation assumption ensures that any two different mean sequences are not too similar. We get the following upper bound based on a potential function argument that takes the DTW distance of the computed average sequence to the input sequences as the potential function and analyses the decrease of the potential in any iteration of DBA.

▶ **Theorem 1.** *If the input data satisfies the normalization assumption for B and separation assumption for ε, then the number of steps performed by DBA is at most $\frac{B(m+k)}{\varepsilon}$.*

3 Smoothed Analysis

Our randomness model is as follows: An adversary specifies an instance $X' \in ([0,1]^d)^{nm}$ of n point sequences $\gamma_1, \dots, \gamma_n$ of length m in $[0,1]^d$, where each sequence γ_i is given by its m points $\gamma_i = (\gamma_{i,1}, \dots, \gamma_{i,m})$. Then we add to each vertex of X' an independent d-dimensional random vector with independent Gaussian coordinates of mean 0 and standard deviation σ. The resulting vectors form the input point sequences. We assume without loss of generality that $\sigma \le 1$, since the case $\sigma > 1$ corresponds to a scaled down instance $X' \in ([0, \frac{1}{\sigma}]^d)^{nm}$

with additive d-dimensional Gaussian random vectors with mean 0 and standard deviation 1. We call this randomness model m-length sequences with $\mathcal{N}(0, \sigma)$ perturbation. For this randomness model, we obtain the following result.

▶ **Theorem 2.** *Suppose $d \geq 2$, then the expected number of iterations until DBA converges is at most*

$$O\left(\frac{n^2 m^{8\frac{n}{d}+6} d^4 k^6 \ln(nm)^4}{\sigma^2}\right).$$

To prove Theorem 2, we first bound the probability that the normalization assumption and the separation assumption hold for suitable parameters. Using these bounds and Theorem 1 in the calculation of the expected number of iterations then yields the theorem. The bound for the probability that the normalization assumption holds can be derived with standard Gaussian tail bounds. The bound for the probability that the separation assumption holds on the other hand uses the fact that each two different average sequences differ in the position of at least one center point. Note that the separation between the average sequences is at least as big as the distance of the two positions of that center point. So, we bound the probability that this distance is greater than some ε using the perturbation of the input. Then, we apply a union bound over all possible combinations of input points that determine different center points.

References

1 Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

2 François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

3 Skyler Seto, Wenyu Zhang, and Yichen Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *2015 IEEE symposium series on computational intelligence*, pages 1399–1406. IEEE, 2015.

4 Thanchanok Teeraratkul, Daniel O'Neill, and Sanjay Lall. Shape-based approach to household electric load curve clustering and prediction. *IEEE Transactions on Smart Grid*, 9(5):5196–5206, 2017.

5 Holger Teichgraeber and Adam R Brandt. Clustering methods to find representative periods for the optimization of energy systems: An initial framework and comparison. *Applied energy*, 239:1283–1293, 2019.

Tracking Evolving Labels using Cone based Oracles

Aditya Acharya
Department of Computer Science
University of Maryland, College Park MD, USA

David M. Mount
Department of Computer Science and Institute for Advanced Computer Studies
University of Maryland, College Park MD, USA

1 Introduction

The *evolving data framework* was first proposed by Anagnostopoulos *et al.* [2], where an *evolver* makes small changes to a structure behind the scenes. Instead of taking a single input and producing a single output, an algorithm judiciously *probes* the current state of the structure and attempts to continuously maintain a sketch of the structure that is as close as possible to its actual state. There have been a number of problems that have been studied in the evolving framework [3–6] including our own work on labeled trees [1].

We were motivated by the problem of maintaining a labeling in the plane, where updating the labels require physically moving them. Applications involve tracking evolving disease hot-spots via mobile testing units [7], and tracking unmanned aerial vehicles [9].

To be specific, we consider the problem of tracking labeled nodes in the plane, where an evolver continuously swaps labels of any two nearby nodes in the background unknown to us. We are tasked with maintaining a hypothesis, an approximate sketch of the locations of these labels, which we can only update by physically moving them over a sparse graph. We assume the existence of an *Oracle*, which when suitably probed, guides us in fixing our hypothesis.

2 Theta Graphs and Routing

Theta graph is a sparse geometric spanner with a spanning ratio: $t_\theta = 1/(1 - 2\sin(\theta_k/2))$ spanners, where $\theta_k = 2\pi/k$ is the angle induced by a set of cones around a point [8] (Figure 1(a)). Using a pre-computed theta graph, there is a simple routing scheme: let $C_{p,q}$ be the cone around p that contains q. We move from p to its neighbor in $C_{p,q}$, and repeat the process. (See Figure 1(b)). Note we only need information about the neighbors of p, and a rough location estimate for the target point q.

3 Problem Formulation

Let $P \in \mathbf{R}^2$, be a point set of size n. Each point is assigned a unique label from the set of labels $L = \{l_1, \ldots, l_n\}$, through a matching M. Every time step, the evolver swaps labels between points that are less than a unit distance apart.

Let G_θ be the theta graph on P, and Σ_θ be its embedding. Through $H : L \to \Sigma_\theta$, we maintain a mapping as close as possible to the original one. We update H by moving a single label over Σ_θ with a finite speed, at any particular moment of time. One of our goals is to find such a speed, which is sufficient to maintain a reasonable H.

We define the distance $\mathcal{D}_l$, for label l as the Euclidean distance between its hypothesized location, and its actual location. And the overall distance between the two mappings M,

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

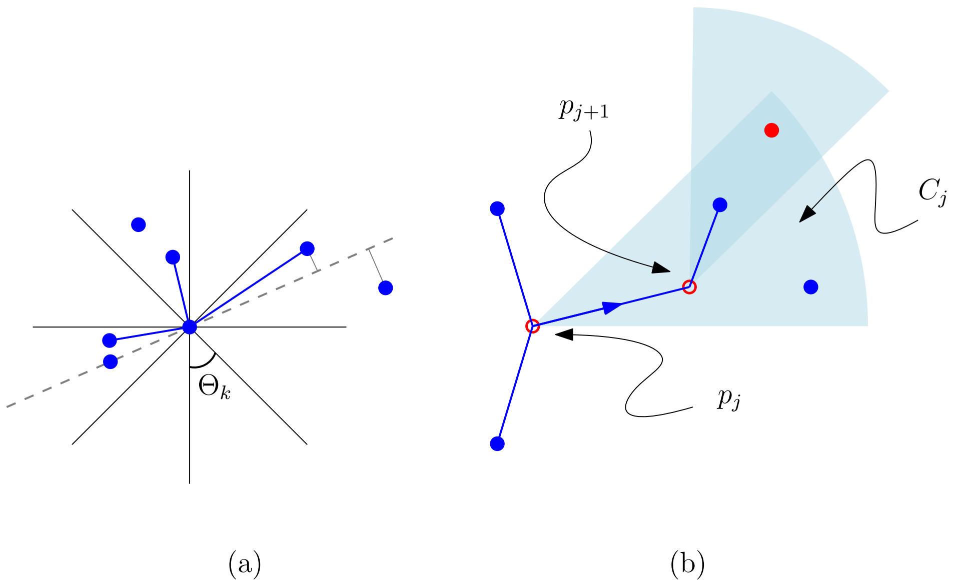


Figure 1 Theta Graph: Construction, and Routing. 1(a) shows the construction of theta graphs. For a point, consider the set of cones around it. For every cone add an edge between that point, and the point which is nearest to it when projected along the bisector of the cone. 1(b) shows how to route using theta graphs. If C_j is the cone at p_j containing the destination point (Red pt), then follow the edge contained inside it. Repeat the same at p_{j+1} until destination is reached

and H as the sum of all such distances: $\sum_{l \in L} \mathcal{D}_l = \mathcal{D}(M, H)$. It's easy to see that, this can get as large as $O(n^2)$ without a competing algorithm.

We assume we have access to an oracle, that we call a *Cone Oracle*: $\mathcal{O}$, which when queried on a label l, and its hypothesized location $H(l)$, returns the cone C_k around $H(l)$ which contains the true location of l.

We assume an additional thing here: for a label l, after an initial startup cost, subsequent queries on the oracle with the same label, takes $o(1)$ time. This is a valid assumption on massive point sets, when only a small region around the current point, in our case $H(l)$, is *cached* in memory. The expensive step is moving to a completely different, possibly uncached area of the data set. In other words switching from l_1 to l_2, on the oracle takes constant amount of time, what we call the *memory overhead*.

Our aim is to design an algorithm that physically moves the hypothesized labels over a sparse graph with a constant speed, and maintains a labeling of distance $O(n)$ at all times.

4 Algorithm

First we precompute the theta graph for our point set P. Then we run the randomized algorithm 1. (We assume the evolver does not have access to our random bit generator). In short, our algorithm selects a label at random every iteration, and tracks it down using the routing scheme on theta graphs (Fig 1(b)). We probe the oracle to find the cone around a point which contains the actual location of the label.

Algorithm 1 Randomized algorithm using cone oracle

```
Require: $\{H(l_1), H(l_2), \cdots, H(l_n)\}$        ▷ Initial hypothesized location of the labels
 1: while true do                                        ▷ Continuously run the algorithm
 2:     for $l$ chosen randomly uniformly from $\{l_1, l_2, \cdots, l_n\}$ do
 3:         while $\mathcal{O}(l, H(l)) \neq NULL$ do                ▷ Keep tracking $l$ until found
 4:             Let $C_k \leftarrow \mathcal{O}(l, H(l))$                ▷ Cone returned by the oracle
 5:             Let $e_{l,k} = (H(l), v)$ be the edge incident on $H(l)$ inside $C_k$
 6:             Move $l$ along $e_{l,k}$ with speed $ct_\theta$             ▷ $c$ is a constant
 7:             Update $H(l) \leftarrow v$
 8:         end while
 9:     end for
10: end while
```

5 Analysis

Under a cached memory model, with overhead M, lines 3, 4, 5, and 7 in Algorithm 1 take $o(1)$ time. Let $\mathcal{D}_i$ be the overall distance, and $\mathcal{D}_{l,i}$ be the distance for label l at the start of the i^{th} iteration. Since we are moving at the speed of ct_θ over a t_θ spanner, we spend $\mathcal{D}_{l,i}/c$ time in traversing Euclidean distance $\mathcal{D}_{l,i}$. In that time the evolver could have moved the label by at most another $\mathcal{D}_{l,i}/c$ distance. So, in the worst case, we spend at most $\mathcal{D}_{l,i}/c + \mathcal{D}_{l,i}/c^2 + \cdots = \mathcal{D}_{l,i}/(c-1)$ time tracking l. In that time, the evolver can increase $\mathcal{D}_i$ by at most $2\mathcal{D}_{l,i}/(c-1)$. Since we chose l at random, we have $\mathrm{E}[\mathcal{D}_{l,i}] = \mathcal{D}_i/n$. Therefore,

$$\begin{aligned}
\mathrm{E}\left[\mathcal{D}_{i+1} \mid \mathcal{D}_i\right] &\leq \mathcal{D}_i - \frac{\mathcal{D}_i}{n} + \frac{2}{c-1} \cdot \frac{\mathcal{D}_i}{n} + M \\
\implies \mathrm{E}\left[\mathcal{D}_{i+1}\right] &\leq \left(1 - \frac{c-3}{c-1} \cdot \frac{1}{n}\right) \mathrm{E}\left[\mathcal{D}_i\right] + M \\
&\leq \left(1 - \frac{z}{n}\right)^i \mathrm{E}\left[\mathcal{D}_1\right] + M \sum_{j=1}^{i} \left(1 - \frac{z}{n}\right)^i && \left[z = \frac{c-3}{c-1}\right] \\
&\leq \left(1 - \frac{z}{n}\right)^i O(n^2) + \frac{nM}{z} && [z > 0]
\end{aligned}$$

For $i = \ln n/(\ln n - \ln(n-z)) \sim n \ln n/z$, we have $\mathcal{D}_{i+1} \in O(n)$, provided $c > 3$. Therefore,

▶ **Theorem 1.** *There exists a randomized algorithm using a cone oracle, which moves labels at any speed greater than $3\,t_\theta$, and in expectation maintains a hypothesized labeling with distance: $O(n)$, in the presence of an adversarial evolver.*

References

1 Aditya Acharya and David M. Mount. Optimally tracking labels on an evolving tree. In Yeganeh Bahoo and Konstantinos Georgiou, editors, *Proceedings of the 34th Canadian Conference on Computational Geometry, CCCG 2022, Toronto Metropolitan University, Toronto, Ontario, Canada, August 25-27, 2022*, pages 1–8, 2022.

2 Aris Anagnostopoulos, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. Sorting and selection on dynamic data. *Theoretical Computer Science*, 412(24):2564–2576, 2011.

3 Aris Anagnostopoulos, Ravi Kumar, Mohammad Mahdian, Eli Upfal, and Fabio Vandin. Algorithms on evolving graphs. In *Proceedings of the 3rd Innovations in Theoretical Computer*

Science Conference, ITCS ’12, page 149–160, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2090236.2090249.

4 Bahman Bahmani, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. Pagerank on an evolving graph. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’12, page 24–32, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2339530.2339539.

5 Juan Jose Besa, William E. Devanny, David Eppstein, Michael T. Goodrich, and Timothy Johnson. Optimally Sorting Evolving Data. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81:1–81:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9085, doi:10.4230/LIPIcs.ICALP.2018.81.

6 Varun Kanade, Nikos Leonardos, and Frédéric Magniez. Stable matching with evolving preferences. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2016.

7 Kanglin Liu, Changchun Liu, Xi Xiang, and Zhili Tian. Testing facility location and dynamic capacity planning for pandemics with demand uncertainty. *European Journal of Operational Research*, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0377221721009796, doi:https://doi.org/10.1016/j.ejor.2021.11.028.

8 Jim Rupert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

9 Ugur Zengin and Atilla Dogan. Real-time target tracking for autonomous UAVs in adversarial environments: A gradient search algorithm. *IEEE Transactions on Robotics*, 23(2):294–307, 2007.

Reconfiguration of 3D Pivoting Modular Robots

Hugo A. Akitaya
University of Massachusetts Lowell, USA

Frederick Stock
University of Massachusetts Lowell, USA

Abstract

We study a new model of 3-dimensional modular self-reconfigurable robots Rhombic Dodecahedral (RD). By extending results on the 2D analog of this model we characterize the free space requirements for a pivoting move and investigate the *reconfiguration problem*, that is, given two configurations s and t is there a sequence of moves that transforms s into t? We show reconfiguration is PSPACE-hard for RD modules in a restricted pivoting model. In a more general model, we show that RD configurations are not universally reconfigurable despite the fact that their 2D analog is [Akitaya et al., SoCG 2021]. Additionally, we present a new class of RD configurations that we call *super-rigid*. Such a configuration remains rigid even as a subset of any larger configuration, which does not exist in the 2D setting.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases modular reconfigurable robotics, pivoting model, reconfiguration

Related Version A full version of the paper is available at `https://arxiv.org/abs/2304.09990`.

1 Introduction

Programmable matter refers to matter with the ability to change its physical properties, such as shape, on demand. A popular approach to implement such a system is through *modular self-reconfigurable robotic systems* (MSR) where small robotic units called *modules* can attach and detach from each other, communicate, and move relative to each other, changing the shape of the system. We require configurations to be *connected*, meaning the adjacency graph of modules is connected (known as the *single backbone condition* [3]). A *move* is an operation that transforms a configuration into another by changing the position of a single module. We focus on the *pivoting model* where the moving module rotates around a static module. This model is similar to many hardware implementations [5, 6] but it is challenging from an algorithmic perspective as a single pivoting move can collide with several modules.

If certain local configurations are forbidden, there are polynomial-time algorithms for square, hexagonal, and cube modules that compute a sequence of pivoting moves to transform a configuration into another [4, 7]. Akitaya et al. [1, 2] classified three sets of pivoting moves for square modules and two for hexagonal modules and described their required free space, Fig. 1. The *restricted hexagonal model* only uses the restricted move while the *monkey hexagonal model* uses both restricted and monkey moves. The general problem was shown to be PSPACE-hard for the restricted hexagonal model [1]. However, [1] also gave a universal reconfiguration algorithm for the monkey hexagonal model that produces move sequences of length $O(n^3)$, for configurations of n robots.

Not much is known about general reconfiguration in 3D models. A candidate to generalize hexagonal modules is the rhombic dodecahedron (RD). Implementations of RD-like modules exist [5], but free space constraints for pivoting moves of these modules have never been

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

described as in [1, 2]. We present such free space constraints to facilitate the description of 3D configurations and moves of such models. We then generalize the PSAPCE-hardness proofs from [1] to RD pivoting models. Finally, we show there are configurations of RD that are rigid even if they are a subset of a larger configuration, implying the configuration space of RD is disconnected.

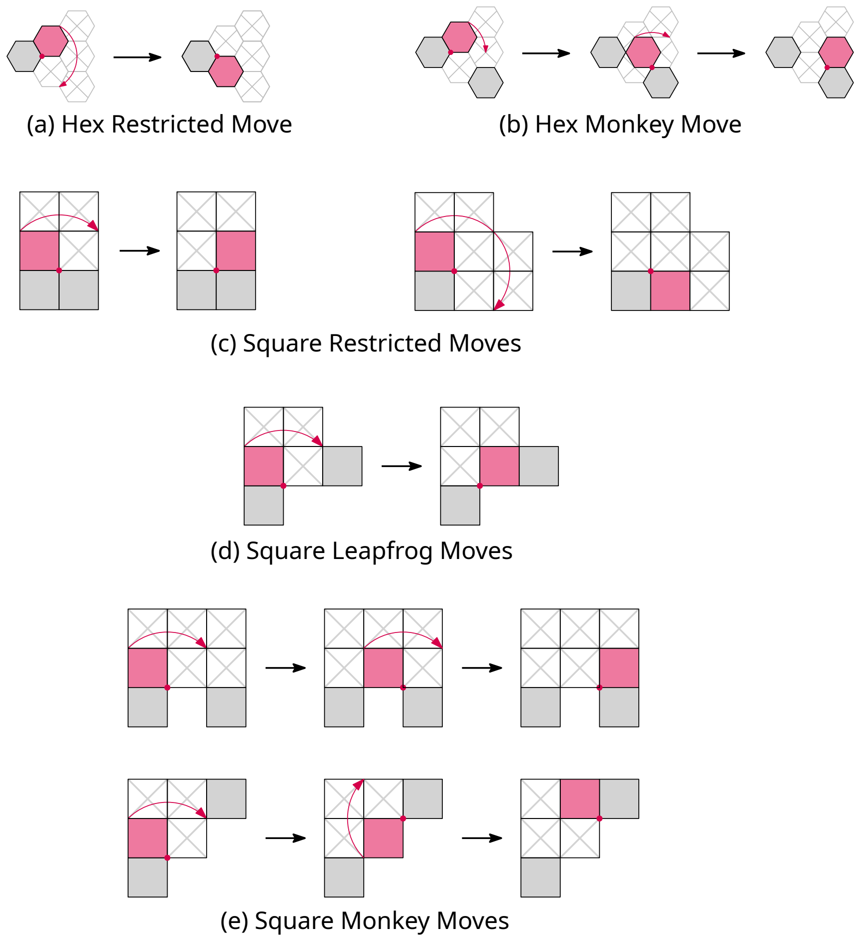


Figure 1 2-dimensional (a,c) restricted moves, (d) leapfrog, and (b, e) monkey moves.

2 Free-Space

A RD lattice can be represented as stacked layers of hexagons (Fig. 2), and RD are space-filling polyhedra and therefore are a logical 3D analog to hexagonal MSR. We define restricted and monkey moves as in [1]. During a pivoting move, a RD module might collide with modules that are in layers adjacent to its plane of movement, so three layers of modules are needed to describe their free-space requirements, presented in Fig. 3. A RD lattice is 3-cyclic so the bottom and top layer free-space requirements may flip based on a module's lattice position or direction of movement, hence Fig. 3 is presented w.l.o.g.

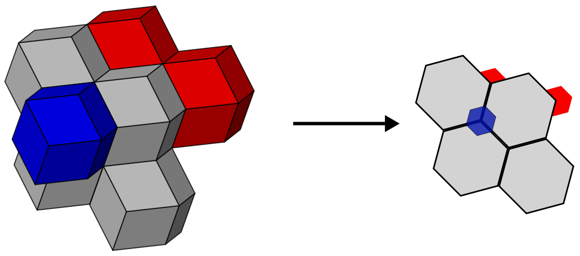


Figure 2 Equivalency between rhombic dodecahedra and hexagons. Red and blue modules are drawn small for legibility but are, in reality as large as the central grey modules.

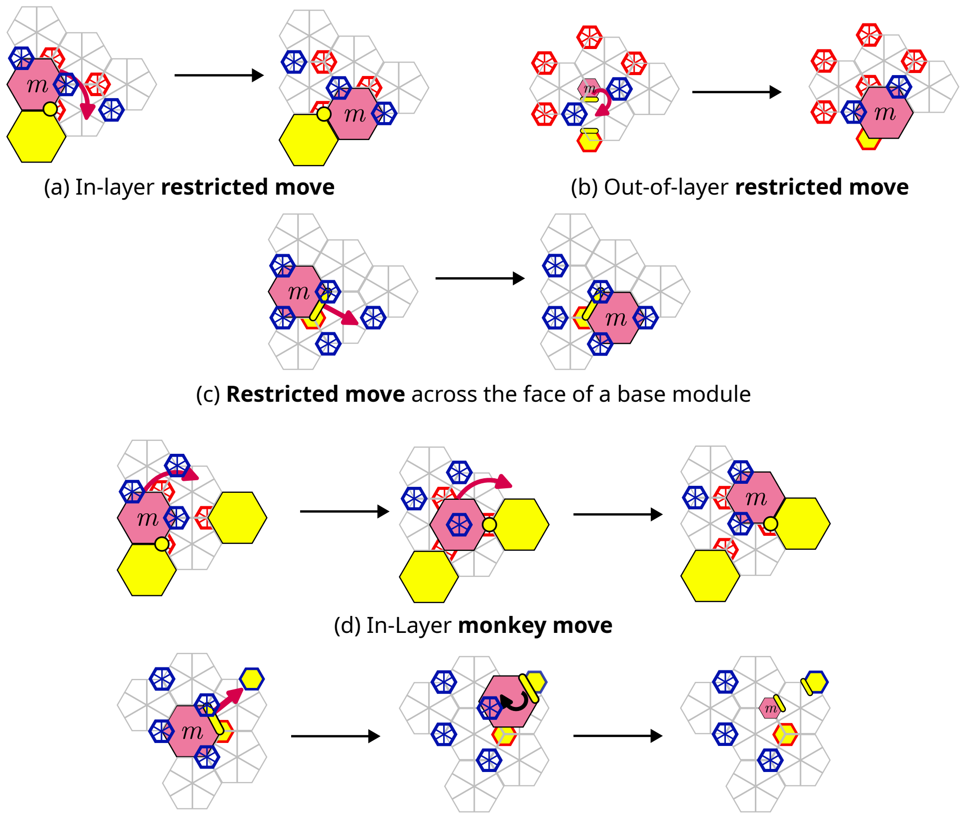

(a) In-layer **restricted move**
(b) Out-of-layer **restricted move**
(c) **Restricted move** across the face of a base module
(d) In-Layer **monkey move**
(e) Out-of-layer **monkey move**

Figure 3 3-dimensional RD Moves. The moving module m is pink, and the modules and the edges that m pivots around are highlighted in yellow. The required free lattice positions (resp. top, resp. bottom) are marked with a grey (blue, red) asterisk.

3 Reconfiguration is Hard

▶ **Theorem 1.** *General Reconfiguration of RD with Restricted moves is PSPACE-hard.*

We use the configurations in Fig. 4 to extend the gadgets used by Akitaya et al. [1] from 2D to 3D. A roof configuration is a gadget contained in a single layer formed by a cycle of modules, the *boundary*, and all positions that are enclosed by the boundary. A cap configuration takes a path of modules and makes one end rigid. We use the cap to make our roof rigid by attaching a path of modules to each module in the boundary of a roof and terminating it with a cap. By placing instances of the roof configurations two layers above and below the gadgets, no module can move up or down a layer, essentially restricting the gadgets to two dimensions. A similar approach can extend the square results from [1] to prove reconfiguration of cubes under restricted, leapfrog, and monkey moves is PSPACE-hard as well. Details will be in an upcoming full version.

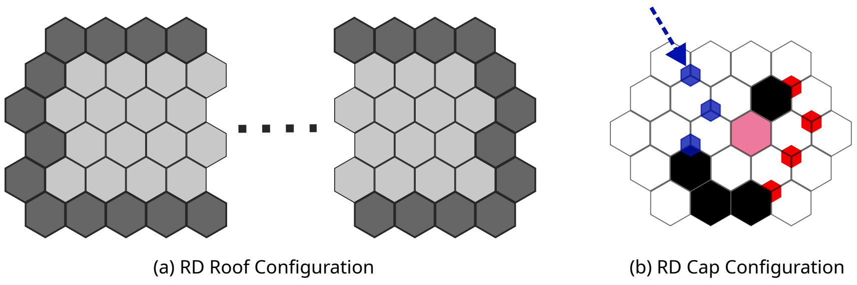


Figure 4 Roof and Cap configurations for rhombic dodecahedron.

4 Rhombic Dodechahedra Are Super Rigid

In this section, we define a new class of configuration we call *super-rigid*.

▶ **Definition 2.** *A configuration G is super rigid if given any configuration C where G is a subconfiguration, the modules in G cannot be moved.*

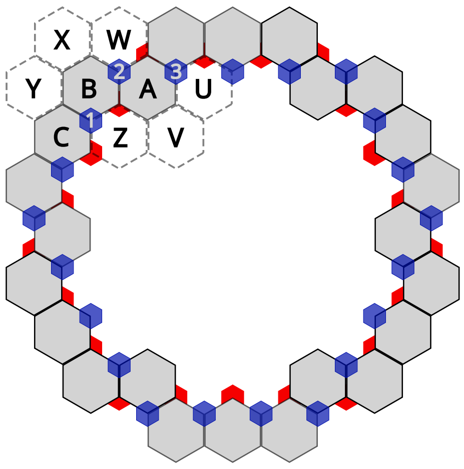
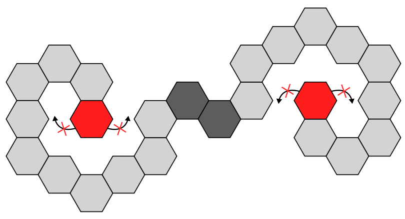


Figure 5 A super rigid configuration, positions U, V, W, X, Y, and Z are empty

Where a *subconfiguration* of a configuration C is any connected configuration that can be obtained by removing any number of modules from C. To our knowledge, super-rigid configurations are novel, and likely only possible in MSR models where moving modules can have collisions outside their plane of movement. Previously known rigid configurations (including all 2D rigid configurations) are like configuration (a) in Fig. 6, where adding just one new module (in blue) the configuration becomes mobile (b).

▶ **Theorem 3.** *The configuration in Fig. 5 is super rigid under restricted and monkey moves.*


(a) Rigid Configuration

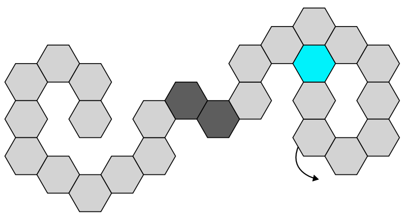

(b) Mobile Configuration

Figure 6 In (b) the addition of the blue module makes configuration (a) mobile

The proof uses the free-space requirements in Fig. 3 to verify no module in G is mobile and none can be made mobile by adding any modules to G. Refer to the full version of this paper for a detailed proof. A surprising consequence of this is a solid "cube" containing Fig. 5 cannot be reconfigured into a single row of modules in a line.

▶ **Corollary 4.** *The Rhombic Dodechadral MSR model is not universally reconfigurable under Restricted or Monkey moves.*

References

1 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing Universal Reconfigurability of Modular Pivoting Robots. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/13809`, `doi:10.4230/LIPIcs.SoCG.2021.10`.

2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The O(1) musketeers. *Algorithmica*, 83(5):1316–1351, 2021. `doi:10.1007/s00453-020-00784-6`.

3 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: Feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20(3):409–418, 2004.

4 Daniel Feshbach and Cynthia Sung. Reconfiguring non-convex holes in pivoting modular cube robots. *IEEE Robotics Autom. Lett.*, 6(4):6701–6708, 2021. `doi:10.1109/LRA.2021.3095030`.

5 Benoit Piranda and Julien Bourgeois. Designing a quasi-spherical module for a huge modular robot to create programmable matter. *Autonomous Robots*, 42:1619–1633, 2018.

6 John W Romanishin, Kyle Gilpin, and Daniela Rus. M-blocks: Momentum-driven, magnetic modular robots. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 4288–4295. IEEE, 2013.

7 Cynthia R. Sung, James M. Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 1933–1940. IEEE, 2015. `doi:10.1109/ICRA.2015.7139451`.

Convergence of Leray Cosheaves for Decorated Mapper Graphs

Justin Curry
University at Albany, State University of New York, USA

Washington Mio
Florida State University, Tallahassee, Florida

Tom Needham
Florida State University, Tallahassee, Florida

Osman Berat Okutan
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

Florian Russold[1]
Graz University of Technology, Austria

Abstract

We introduce decorated mapper graphs as a generalization of mapper graphs capable of capturing more topological information of a data set. A decorated mapper graph can be viewed as a discrete approximation of the cellular Leray cosheaf over the Reeb graph. We establish a theoretical foundation for this construction by showing that the cellular Leray cosheaf with respect to a sequence of covers converges to the actual Leray cosheaf as the resolution of the covers goes to zero.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Leray cosheaves, Reeb graphs, Mapper, convergence

Funding
Justin Curry: Supported by NSF CCF-1850052 and NASA 80GRC020C0016
Washington Mio: Supported by NSF grant DMS-1722995
Tom Needham: Supported by NSF DMS-2107808
Florian Russold: Supported by the Austrian Science Fund (FWF): W1230

1 Introduction

Reeb graphs and their discrete analogs—mapper graphs—are important tools in computational topology [2, 3, 8, 14]. They are used for data visualization [10, 12], for comparing scalar fields via distances between Reeb graphs [1, 6, 7], and data skeletonization [9], among other things. Given a continuous map $f\colon X \to \mathbb{R}$, the Reeb graph R_f summarizes the zero-dimensional connectedness of X. In practice, we deal with maps $f\colon P \to \mathbb{R}$ on discrete data sets P, where Reeb graphs are replaced by mapper graphs. A mapper graph is constructed by choosing a cover $\mathcal{U} = (U_i)_{i\in I}$ of $f(X)$ or $f(P)$, taking the components or clustering the data points of $f^{-1}(U_i)$, collapsing each of the obtained components to a single vertex and connecting two vertices if their corresponding components have common points. In [6, 11] it is shown that Reeb graphs can be viewed as cosheaves and that the cellular Reeb cosheaf w.r.t. a cover converges to the Reeb cosheaf if the resolution of the cover goes to zero, establishing

[1] Corresponding Author

[*] This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

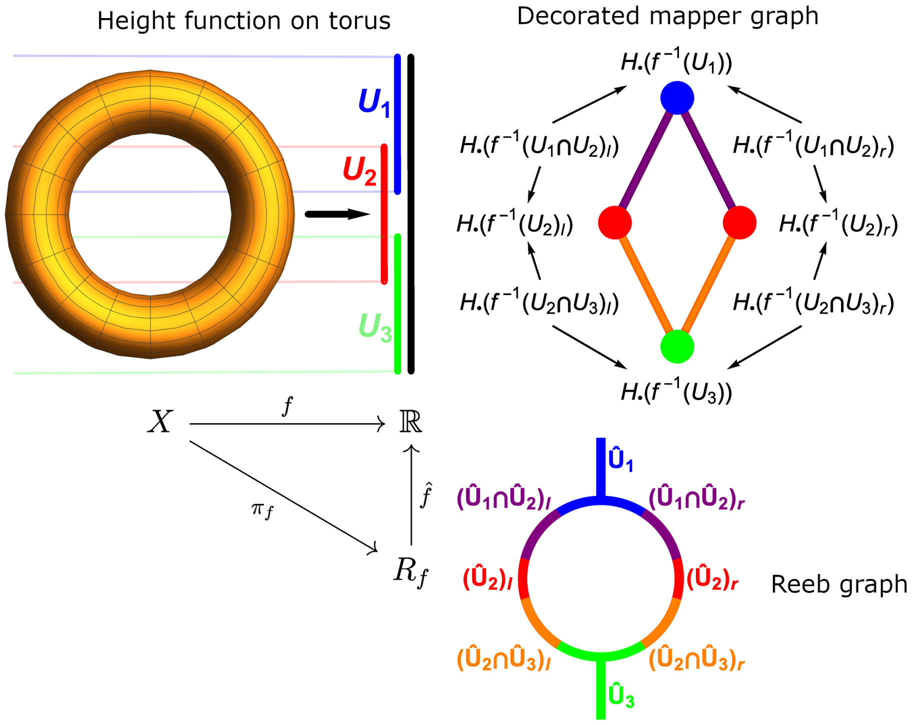


Figure 1 Pulling back the cover $\mathcal{U}$ of $f(X)$ along the induced map $\hat{f}$ on the Reeb graph and refining it into components yields a cover $\hat{\mathcal{U}}$ of R_f. The decorated mapper graph is the nerve complex of this cover decorated by the homology $H_\bullet\big(\pi_f^{-1}(-)\big)$ of the preimages of intersections in $\hat{\mathcal{U}}$ under the quotient map π_f. Since $f = \hat{f} \circ \pi_f$, this is analogous to using components of preimages under f.

a theoretical justification for working with finite covers. A limitation of this approach is that Reeb/mapper graphs fail to capture higher-dimensional topological features. To overcome these limitations we introduce decorated mapper graphs. A decorated mapper graph collects the homology (possibly inferred from a point sample) in all degrees $H_\bullet(-) := \bigoplus_{n\geq 0} H_n(-)$ of the components of $f^{-1}(U_i)$ and $f^{-1}(U_i \cap U_j)$ on the corresponding vertices and edges of the mapper graph; see Figure 1. It can be viewed as a discrete approximation of the Leray cosheaf over the Reeb graph. In this abstract we show that the cellular Leray cosheaf w.r.t. a cover converges to the actual Leray cosheaf for any finite 1D cover $\mathcal{U}$ as the resolution of the cover goes to zero.

2 Leray Cosheaves

The Leray (pre)cosheaf [4,5,15] parameterizes the homology of a space X when viewed along a continuous map $f\colon X \to Y$. It does this by recording for each pair of open subsets $V \subseteq W \subseteq Y$ their homology and the induced map $H_\bullet\big(f^{-1}(V)\big) \xrightarrow{H_\bullet(f^{-1}(V)\subseteq f^{-1}(W))} H_\bullet\big(f^{-1}(W)\big)$. We further assume that X is compact and $H_\bullet\big(f^{-1}(V)\big)$ is finite-dimensional for every $V \subseteq Y$.

▶ **Definition 1** (Leray (pre)cosheaf). *We define the graded Leray (pre)cosheaf $\mathcal{L}^f$ of a continuous map $f\colon X \to Y$ by the following assignments: For all $V \subseteq W \subseteq Y$ open*

$$\mathcal{L}^f(V) = \bigoplus_{n\in\mathbb{N}_0} \mathcal{L}^f_n(V) := \bigoplus_{n\in\mathbb{N}_0} H_n\big(f^{-1}(V)\big)$$
$$\mathcal{L}^f(V \subseteq W) = \bigoplus_{n\in\mathbb{N}_0} \mathcal{L}^f_n(V \subseteq W) := \bigoplus_{n\in\mathbb{N}_0} H_n\big(f^{-1}(V) \subseteq f^{-1}(W)\big)\,.$$

In practice, we can not access the whole Leray cosheaf. We can only get a cellular version [4, 4.1.6] given by its values on a finite cover of $f(X)$. An open cover $\mathcal{U} = (U_i)_{i\in I}$ of $f(X)$ defines a simplicial complex $\mathcal{N}_\mathcal{U} := \{\sigma = (i_0,\dots,i_k) |\ \mathbf{U}_\sigma := U_{i_0} \cap \ldots \cap U_{i_k} \neq \emptyset\}$, called the nerve complex of $\mathcal{U}$. We call $\mathcal{U}$ a **finite 1D cover** if it is finite and has a one-dimensional nerve complex and denote by $\leq$ the face-relation of $\mathcal{N}_\mathcal{U}$, i.e. $\sigma \leq \tau \iff \sigma$ is a face of τ .

▶ **Definition 2** (Cellular Leray cosheaf). *We define the cellular Leray cosheaf $D_\mathcal{U}\mathcal{L}^f$ w.r.t. a cover $\mathcal{U}$ as the cosheaf on $\mathcal{N}_\mathcal{U}$ given by the following assignments: For all $\sigma \leq \tau \in \mathcal{N}_\mathcal{U}$*

$$D_\mathcal{U}\mathcal{L}^f(\sigma) := \mathcal{L}^f(U_\sigma)$$
$$D_\mathcal{U}\mathcal{L}^f(\sigma \leq \tau) := \mathcal{L}^f(U_\tau \subseteq U_\sigma)\,.$$

If $\pi_f\colon X \to R_f$ is the quotient map from X to the Reeb graph and $\hat{\mathcal{U}}$ a cover of R_f, then we define a decorated mapper graph as $D_{\hat{\mathcal{U}}}\mathcal{L}^{\pi_f}$; see Figure 1.

3 Convergence

It is obvious that this process of discretization can lose information. This raises the question: How well is $\mathcal{L}^f$ represented by $D_\mathcal{U}\mathcal{L}^f$? To compare $\mathcal{L}^f$ and $D_\mathcal{U}\mathcal{L}^f$, we define a process of transforming $D_\mathcal{U}\mathcal{L}^f$ into a (pre)cosheaf on Y. We want to approximate the value of $\mathcal{L}^f$ on any open set $V \subseteq Y$ given only the information in $D_\mathcal{U}\mathcal{L}^f$. To this end, we define the subcomplex $K_V := \{\sigma \in \mathcal{N}_\mathcal{U} \mid U_\sigma \cap V \neq \emptyset\} \leq \mathcal{N}_\mathcal{U}$, which can be viewed as a simplicial approximation of $V \subseteq \bigcup_{\sigma\in K_V} U_\sigma$; see Figure 2. Moreover, if $V \subseteq W$, we obtain an inclusion $\iota\colon K_V \hookrightarrow K_W$.

▶ **Definition 3** (Continuous extension). *Let $\mathcal{U}$ be a finite 1D cover of Y. We define the continuous extension $C_\mathcal{U}D_\mathcal{U}\mathcal{L}^f$ of $D_\mathcal{U}\mathcal{L}^f$ as a (pre)cosheaf on Y via the following assignments: For all $V \subseteq W \subseteq Y$ open*

$$C_\mathcal{U}D_\mathcal{U}\mathcal{L}^f(V) := \bigoplus_{n\in\mathbb{N}_0} \Big(H_0\big(K_V; D_\mathcal{U}\mathcal{L}^f_n|_{K_V}\big) \oplus H_1\big(K_V; D_\mathcal{U}\mathcal{L}^f_{n-1}|_{K_V}\big)\Big)$$
$$C_\mathcal{U}D_\mathcal{U}\mathcal{L}^f(V \subseteq W) := \bigoplus_{n\in\mathbb{N}_0} \Big(H_0\big(\iota\big)_n \oplus H_1\big(\iota\big)_{n-1}\Big)$$

where $H_i\big(K_V; D_\mathcal{U}\mathcal{L}^f_n|_{K_V}\big)$ is the degree i homology [4, 6.2.2] of the cosheaf $D_\mathcal{U}\mathcal{L}^f_n$ restricted to K_V and $H_i(\iota)_n$ is the map induced on cosheaf homology by $\iota\colon K_V \hookrightarrow K_W$ cf. [13, A.17].

As shown in [4, 5, 15], if $\mathcal{U}$ is a finite 1D cover of $f(X)$ the continuous extension yields $C_\mathcal{U}D_\mathcal{U}\mathcal{L}^f(V) \cong \bigoplus_{n\in\mathbb{N}_0} H_n\Big(f^{-1}\big(\bigcup_{\sigma\in K_V} U_\sigma\big)\Big)$ approximating $H_\bullet\big(f^{-1}(V)\big)$; see Figure 2. The following proposition shows that $C_\mathcal{U}D_\mathcal{U}\mathcal{L}^f(V \subseteq W)$ also gives us the correct induced map.

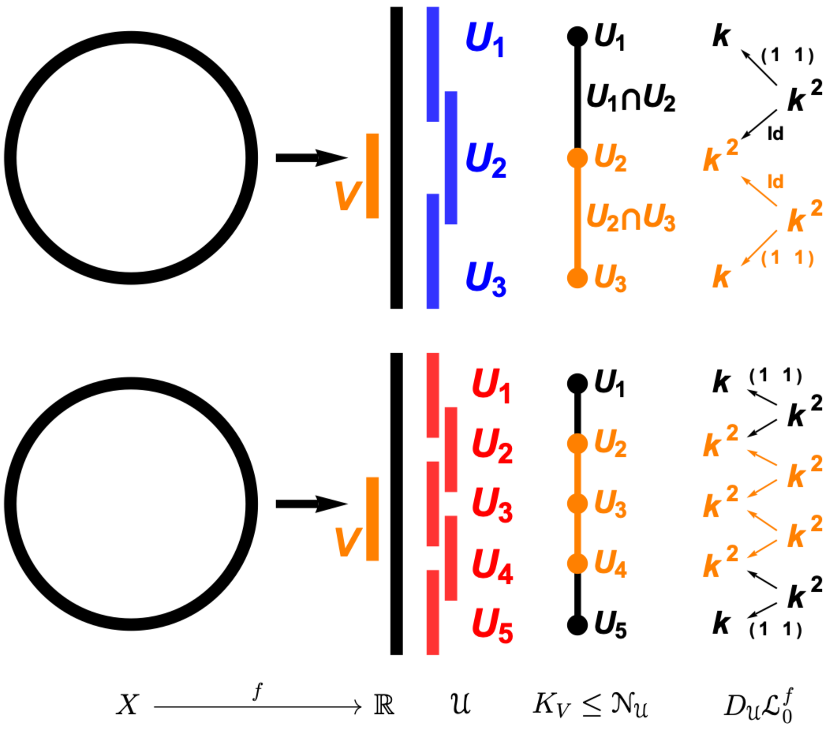


Figure 2 The figure shows $D_{\mathcal{U}}\mathcal{L}_0^f \cong D_{\mathcal{U}}\mathcal{L}^f$ on $\mathcal{N}_{\mathcal{U}}$ w.r.t. two covers with different resolution as well as the restriction $D_{\mathcal{U}}\mathcal{L}_0^f|_{K_V}$ to K_V (in orange). Homology is taken with coefficients in a field k and unlabeled arrows represent identity maps. For the coarse cover we get $C_{\mathcal{U}}D_{\mathcal{U}}\mathcal{L}^f(V) \cong k \ncong H_0\big(f^{-1}(V)\big)$ but for the finer one we get $C_{\mathcal{U}}D_{\mathcal{U}}\mathcal{L}^f(V) \cong k^2 \cong H_0\big(f^{-1}(V)\big)$.

▶ **Proposition 4.** *Let $\mathcal{U}$ be a finite 1D cover of $f(X)$. Then, for all $V \subseteq W \subseteq Y$ open, the following diagram commutes and the horizontal arrows are isomorphisms:*

$$\begin{tikzcd}
H_0\big(K_V; D_{\mathcal{U}}\mathcal{L}_n^f|_{K_V}\big) \oplus H_1\big(K_V; D_{\mathcal{U}}\mathcal{L}_{n-1}^f|_{K_V}\big) \arrow[r, "\cong"] \arrow[d, "H_0(\iota)\oplus H_1(\iota)"'] & H_n\Big(f^{-1}\big(\bigcup_{\sigma\in K_V} U_\sigma\big)\Big) \arrow[d, "H_n\left(f^{-1}\left(\bigcup_{\sigma\in K_V} U_\sigma\right)\subseteq f^{-1}\left(\bigcup_{\sigma\in K_W} U_\sigma\right)\right)"'] \\
H_0\big(K_W; D_{\mathcal{U}}\mathcal{L}_n^f|_{K_W}\big) \oplus H_1\big(K_W; D_{\mathcal{U}}\mathcal{L}_{n-1}^f|_{K_W}\big) \arrow[r, "\cong"'] & H_n\Big(f^{-1}\big(\bigcup_{\sigma\in K_W} U_\sigma\big)\Big)
\end{tikzcd}.$$

To talk about convergence, we have to define a distance on (pre)cosheaves. Assume Y is a metric space and that all the involved (pre)cosheaves are constructible [4, 11.0.10]. For every $\epsilon > 0$ define $V^\epsilon := \{y \in Y \mid d(y,V) < \epsilon\}$. To a (pre)cosheaf F on Y we now associate $F^\bullet := \{(F^\epsilon)_{\epsilon\geq 0}\,,\ (F_\epsilon^\delta\colon F^\epsilon \to F^\delta)_{\epsilon\leq\delta}\}$, a parameterized family of (pre)cosheaves on Y, by setting $F^\epsilon(V) := F(V^\epsilon)$. This allows us to define the distance of two (pre)cosheaves F and G as the interleaving distance of $F^\bullet$ and $G^\bullet$, i.e. $d(F,G) := d_I(F^\bullet, G^\bullet)$. Denote by $\operatorname{res}(\mathcal{U}) := \sup_{U\in\mathcal{U}} \sup_{x,y\in U} d(x,y)$ the resolution of an open cover $\mathcal{U}$. We are now able to show that if the resolution of the cover $\mathcal{U}$ goes to zero, $C_{\mathcal{U}}D_{\mathcal{U}}\mathcal{L}^f$ converges to $\mathcal{L}^f$.

▶ **Theorem 5.** *If $f\colon X \to Y$ is continuous and $\mathcal{U}$ is a finite 1D cover of $f(X)$, then*

$$d\big(C_{\mathcal{U}}D_{\mathcal{U}}\mathcal{L}^f, \mathcal{L}^f\big) \leq res(\mathcal{U})\,.$$

If we only consider $\mathcal{L}_0^f$, the degree-zero part of $\mathcal{L}^f$, the construction in Definition 3 and Theorem 5 specializes to an abelianization of the convergence result in [11].

4 Future Work

Theorem 5 guarantees that, if we choose a cover of resolution $\leq \delta$, the cellular Leray cosheaf is a δ-approximation of the continuous one. This result establishes a theoretical justification for working with finite covers. Since we have to deal with maps from finite point sets in practice, in future work we plan to investigate under what conditions we can infer the cellular Leray cosheaf of a map $f\colon X \to Y$ from a finite sample of X; cf. [2].

References

1 Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance between reeb graphs. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, page 464–473, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2582112.2582169.

2 Adam Brown, Omer Bobrowski, Elizabeth Munch, and Bei Wang. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology*, 5(1):99–140, 2021. doi:10.1007/s41468-020-00063-x.

3 Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003. Special Issue on the Fourth CGC Workshop on Computational Geometry. doi:https://doi.org/10.1016/S0925-7721(02)00093-7.

4 Justin Curry. *Sheaves, Cosheaves and Applications.* PhD thesis, University of Pennsylvania, 2014. arXiv:1303.3255.

5 Justin Curry, Robert Ghrist, and Vidit Nanda. Discrete morse theory for computing cellular sheaf cohomology. *Foundations of Computational Mathematics*, 16, 2013. doi:10.1007/s10208-015-9266-8.

6 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, 2016. doi:10.1007/s00454-016-9763-9.

7 Barbara Di Fabio and Claudia Landi. The edit distance for reeb graphs of surfaces. *Discrete & Computational Geometry*, 55(2):423–461, 2016. doi:10.1007/s00454-016-9758-6.

8 Herbert Edelsbrunner, John Harer, and Amit K. Patel. Reeb spaces of piecewise linear mappings. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry*, SCG '08, page 242–250, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1377676.1377720.

9 Xiaoyin Ge, Issam Safa, Mikhail Belkin, and Yusu Wang. Data skeletonization via reeb graphs. *Advances in Neural Information Processing Systems*, 24, 2011.

10 P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1):1236, 2013. doi:10.1038/srep01236.

11 Elizabeth Munch and Bei Wang. Convergence between Categorical Representations of Reeb Space and Mapper. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SoCG.2016.53.

12 Monica Nicolau, Arnold Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences of the United States of America*, 108:7265–70, 04 2011. `doi:10.1073/pnas.1102826108`.

13 Florian Russold. Persistent sheaf cohomology, 2022. `arXiv:2204.13446`.

14 Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Eurographics Symposium on Point-Based Graphics*. The Eurographics Association, 2007. `doi:10.2312/SPBG/SPBG07/091-100`.

15 Hee Rhang Yoon and Robert Ghrist. Persistence by parts: Multiscale feature detection via distributed persistent homology, 2020. `arXiv:2001.01623`.

Adaptive Approximation of Persistent Homology

Maria Herick
Department of Computer Science, University of Münster, Germany

Michael Joachim
Mathematical Institute, University of Münster, Germany

Jan Vahrenhold
Department of Computer Science, University of Münster, Germany

Abstract

In this paper, we propose an algorithm that approximates the persistent homology of the Čech complex of a given point set by creating an adaptive subsample and calculating the persistent homology of specific filtrations on that sample. We assess the quality of our approach empirically and theoretically: we give topological guarantees for the approximation quality on a global scale as well as locally, keeping a high precision in areas of the initial point sample where it is needed.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Design and analysis of algorithms

Keywords and phrases Topological Data Analysis, Persistent Homology, Local Feature Size

1 Introduction and Related Work

Persistent homology (PH) is an important tool in Topological Data Analysis to infer the topology of an object using a discrete point sample. In this paper, we propose an algorithm that approximates *persistence diagrams* which capture the PH of a filtered space. In particular, we reduce the input sample and build two exemplary filtrations on the resulting set whose PH can be calculated using state-of-the-art software [3, 17, 4]. As a motivating example, we consider point samples of manifolds: We aim at achieving high precision in areas of the manifold with high curvature; this motivates the use of the *local feature size (lfs)*. We contribute a definition of a new dissimilarity measure on adaptive point samples and a localized analysis of persistence modules built on filtrations using this measure. Reducing the sample size has been studied [2, 7, 9, 10, 11, 12, 15, 16, 18, 20], but we explicitly consider local properties and give adaptive guarantees. Dey et al. [13] use a similar approach, but impose stronger limitations on the input to exactly infer the manifold's topology.

2 Algorithm and Theoretical Guarantees

Let X be the input point sample. We initially motivated our approach using the *lfs*, which is a point's distance to the medial axis of the manifold. However, our descriptions and analyses do not rely on specifics of the *lfs*, thus we will model it using a generic function $f : X \to \mathbb{R}$ instead. We coarsen X iteratively by discarding, for each $x \in X$, all points in the ball $B(x, f(x))$ of radius $f(x)$ centered at x and adding only x to the resulting set Y. This is similar to [14] with the exception that we process the points in decreasing f-order and keep track of the set L_x of points that were removed while adding x to Y. To construct a filtration we look at Čech complexes which can be illustrated by centering growing balls on all points and adding edges whenever two balls meet, or – equivalently – when the balls' radii

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

equal some value α, in this case half the points' distance. We improve this notion in our case by defining a dissimilarity measure that considers the discarded points: instead of centering the balls only on the points in Y, we place them on all points of X and add an edge between points $x, y \in Y$ whenever a path $x \to x' \to y' \to y$ with $x' \in L_x, y' \in L_y$ exists, see Figure 1.

▶ **Definition 1.** *Let $d_Y : Y \times Y \to \mathbb{R}$ be defined as*

$$d_Y(x, y) := \min_{x' \in L_x, y' \in L_y} \max \{\|x' - y'\|_2, \|x - x'\|_2, \|y - y'\|_2\}.$$

We define a Čech-like filtration S and a Vietoris-Rips-like filtration T whose index sets are formed by the points in Y. Let $\tilde{L}_y(\alpha) := \{y' \in L_y \mid \|y - y'\|_2 \leq 2\alpha\}$ for $\alpha \in \mathbb{R}^{\geq 0}$.

▶ **Definition 2.** *Let S be a filtration with simplicial complex $S(\alpha)$ for $\alpha \geq 0$ defined as*

$$S(\alpha) := \left\{ \sigma \subseteq Y \,\middle|\, \left(\bigcap_{y \in \sigma} \bigcup_{x \in \tilde{L}_y(\alpha)} B(x, \alpha) \right) \neq \emptyset \right\}.$$

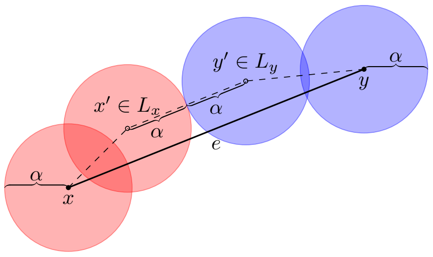
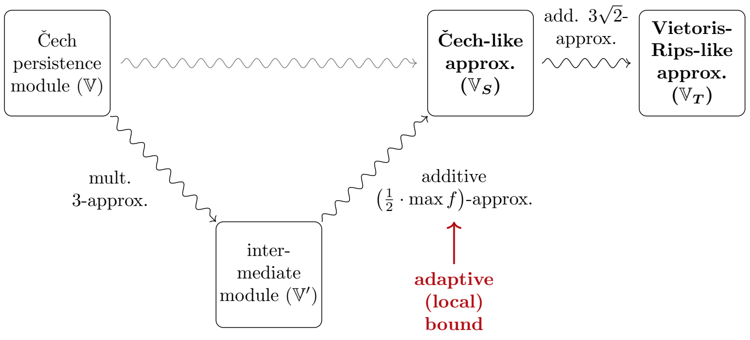


Figure 1 Here, $\alpha := \frac{1}{2} d_Y(x, y)$ is the value at which e appears between x and y in the filtration S using d_Y: this models the existence of a path between x and y in the Čech filtration of X (dashed).

▶ **Definition 3.** *Let T be a filtration with simplicial complex $T(\alpha)$ for $\alpha \geq 0$ defined as*

$$T(\alpha) := \{\{v_0, ..., v_n\} \subseteq Y \mid n \in \mathbb{N}, d_Y(v_i, v_j) \leq 2\alpha \textit{ for all } 0 \leq i, j \leq n\}.$$

We can prove overall and adaptive bounds on the bottleneck distance between the persistence diagram of the Čech module of X and the persistence modules of S and T for each dimension as seen in Figure 2: Let $\mathbb{V}$ be the Čech persistence module on X. First,



Čech persistence module ($\mathbb{V}$)
Čech-like approx. ($\mathbb{V}_S$)
add. $3\sqrt{2}$-approx.
Vietoris-Rips-like approx. ($\mathbb{V}_T$)
mult. 3-approx.
additive ($\frac{1}{2} \cdot \max f$)-approx.
inter-mediate module ($\mathbb{V}'$)
adaptive (local) bound

Figure 2 Visual representation of the bounds on approximation quality.

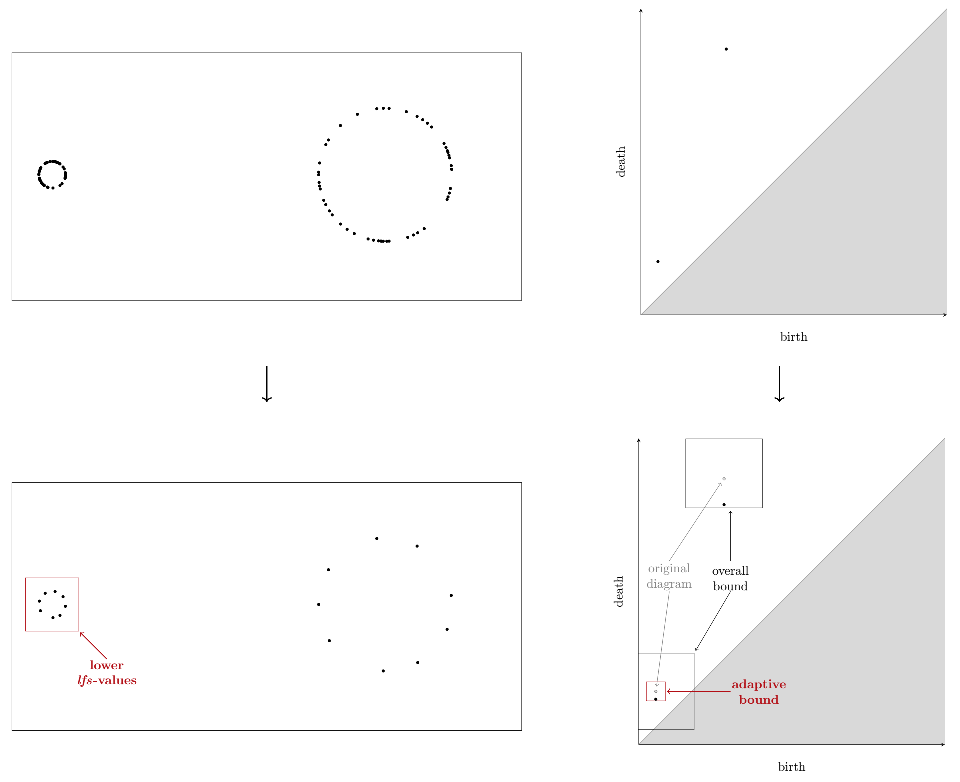


Figure 3 Point sample/adaptive sample (left) and (approximated) persistence diagrams (right).

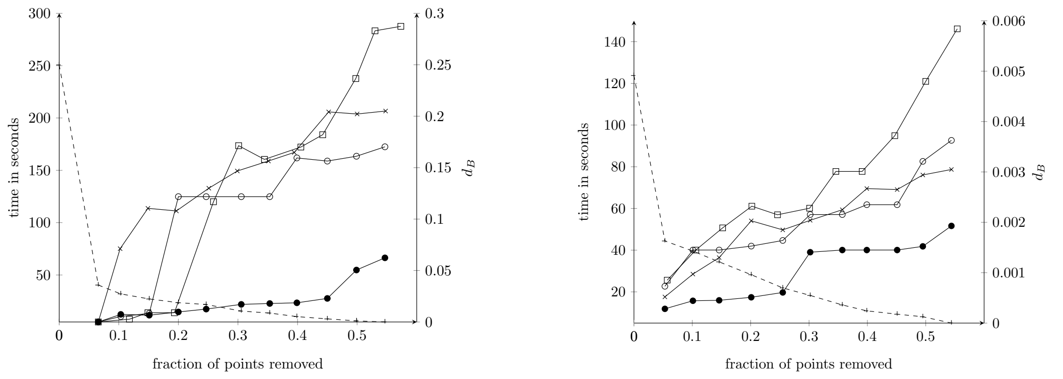
we define an intermediate persistence module $\mathbb{V}'$, similar to [6]. $\mathbb{V}'$ can be shown to be a multiplicative 3-approximation of $\mathbb{V}$. Using [8] we show that $\mathbb{V}_S$, the persistence module built on S, is an additive $\left(\frac{1}{2} \cdot \max f\right)$-approximation of $\mathbb{V}'$ and that the persistence module on T, $\mathbb{V}_T$, $3\sqrt{2}$-approximates $\mathbb{V}_S$. Unfortunately, as seen in Figure 3, a bound on the bottleneck distance that depends on the highest *lfs* might lead to an approximation in which smaller features do not exist. Instead, our construction will yield an adaptive bound (shown in red in Figures 2 and 3): For each feature in $\mathbb{V}'$ we can choose a subset of Y, whose highest f-value determines this feature's approximation quality. This is shown using *induced matchings* [5].

3 Experimental results

For simplicity of computation we calculated $\mathbb{V}_T$. We implemented our algorithm in C++ and compared our approximations to the Vietoris-Rips persistence module on X. We report the results of running Ripser [3] on a Dell XPS 13 (16 GB RAM, Ubuntu 20.04) on different samples – see the table in Figure 4.

Stanford Dragon Some results for the Stanford Dragon [21, 19] are shown in Figure 5. Figure 4 shows that our approach outperforms its competitors on all subsamples. However, there is a limit to how many points we can lose while keeping a low bottleneck distance.

Klein Bottle The second data set is sampled from an immersion of the Klein Bottle in $\mathbb{R}^3$ [19]. Since our method works for general functions f, we can apply it to point sets sampled from other objects than smooth manifolds. The function we used here is similar to the *lfs* approximation by [14]. Figure 6 shows that our approach results in persistence diagrams close to the ground truth even on smaller samples. Again, our approach outperforms its



Approach	Mark	Sampling	Filtration	
(a)	$\times$	uniform	Vietoris Rips	
(b)	$\circ$	(approx.) local feature size [1]	Vietoris Rips	
(c)	$\bullet$	(approx.) local feature size [1]	**our method** (T)	**[this paper]**
(d)	$\square$	*lean feature size* [13]	**our method** (T)	

Figure 4 Bottleneck distance to ground truth for 1-dim. PH (top left: Klein Bottle, $d_B \in [0, 4]$; top right: Stanford Dragon, $d_B \in [0, 0.004]$). The runtime of Ripser (dashed, shown for calibration) depends on the size of the subsample and internal heuristics and thus is similar for all methods.

competitors. In all cases, the running time for Ripser dominates the overall runtime.

Using the *lean feature size* [13] in our algorithm deteriorates the results, likely because the benchmark data set did not conform to Dey et al.'s strong requirements. We conclude that our approach stably outperforms its competitors with respect to the bottleneck distance.

References

1 Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Comput. Geom.*, 19(2-3):127–153, 2001.

2 Naheed Anjum Arafat, Debabrota Basu, and Stéphane Bressan. Topological data analysis with ε-net induced lazy witness complex. In *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part II*, volume 11707 of *Lecture Notes in Computer Science*, pages 376–392. Springer, 2019.

3 Ulrich Bauer. Ripser: efficient computation of vietoris-rips persistence barcodes. *Journal of Applied and Computational Topology*, 2021. doi:10.1007/s41468-021-00071-5.

4 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat - persistent homology algorithms toolbox. *J. Symb. Comput.*, 78:76–90, 2017.

5 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *J. Comput. Geom.*, 6(2):162–191, 2015.

6 Magnus Bakke Botnan and Gard Spreemann. Approximating persistent homology in euclidean space through collapses. *Appl. Algebra Eng. Commun. Comput.*, 26(1-2):73–101, 2015.

7 Yueqi Cao and Anthea Monod. Approximating persistent homology for large datasets. *CoRR*, abs/2204.09155, 2022.

8 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 237–246. ACM, 2009.

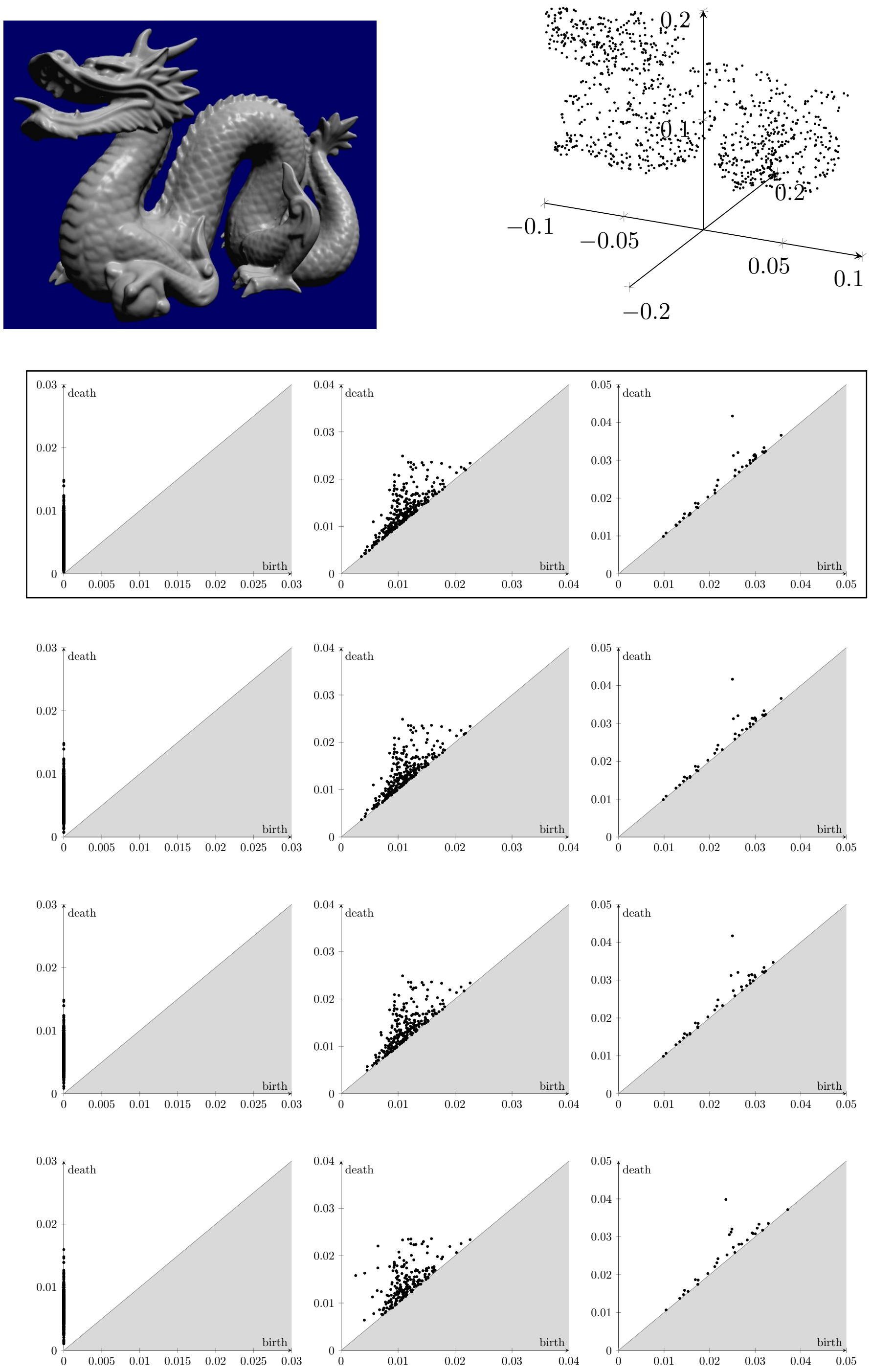


Figure 5 Stanford Dragon [19] with persistence diagrams for 0-, 1-, and 2-dim. PH: Ground truth (framed, 1000 points), our approach (799, 667, and 489 points).

9 Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, and Larry A. Wasserman. Subsampling methods for persistent homology. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2143–2151. JMLR.org, 2015.

10 Frédéric Chazal, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry A. Wasserman. Stochastic convergence of persistence landscapes and silhouettes. *J. Comput. Geom.*, 6(2):140–161, 2015.

11 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra. Improved approximate rips filtrations with shifted integer lattices and cubical complexes. *J. Appl. Comput. Topol.*, 5(3):425–458, 2021.

12 Vin de Silva and Gunnar E. Carlsson. Topological estimation using witness complexes. In Markus H. Gross, Hanspeter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors, *1st Symposium on Point Based Graphics, PBG 2004, Zurich, Switzerland, June 2-4, 2004*, pages 157–166. Eurographics Association, 2004.

13 Tamal K. Dey, Zhe Dong, and Yusu Wang. Parameter-free topology inference and sparsification for data on manifolds. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2733–2747. SIAM, 2017.

14 Stefan Funke and Edgar A. Ramos. Smooth-surface reconstruction in near-linear time. In David Eppstein, editor, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 781–790. ACM/SIAM, 2002.

15 Leonidas J. Guibas and Steve Oudot. Reconstruction using witness complexes. *Discret. Comput. Geom.*, 40(3):325–356, 2008. `doi:10.1007/s00454-008-9094-6`.

16 Michael Kerber and R. Sharathkumar. Approximate cech complexes in low and high dimensions. *CoRR*, abs/1307.3272, 2013.

17 Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, volume 8592 of *Lecture Notes in Computer Science*, pages 167–174. Springer, 2014.

18 Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discret. Comput. Geom.*, 39(1-3):419–441, 2008.

19 Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Sci.*, 6(1):17, 2017.

20 Donald R. Sheehy. Linear-size approximations to the vietoris-rips filtration. *Discret. Comput. Geom.*, 49(4):778–796, 2013.

21 Stanford University Computer Graphics Laboratory. The stanford 3d scanning repository. URL: `https://graphics.stanford.edu/data/3Dscanrep/`.

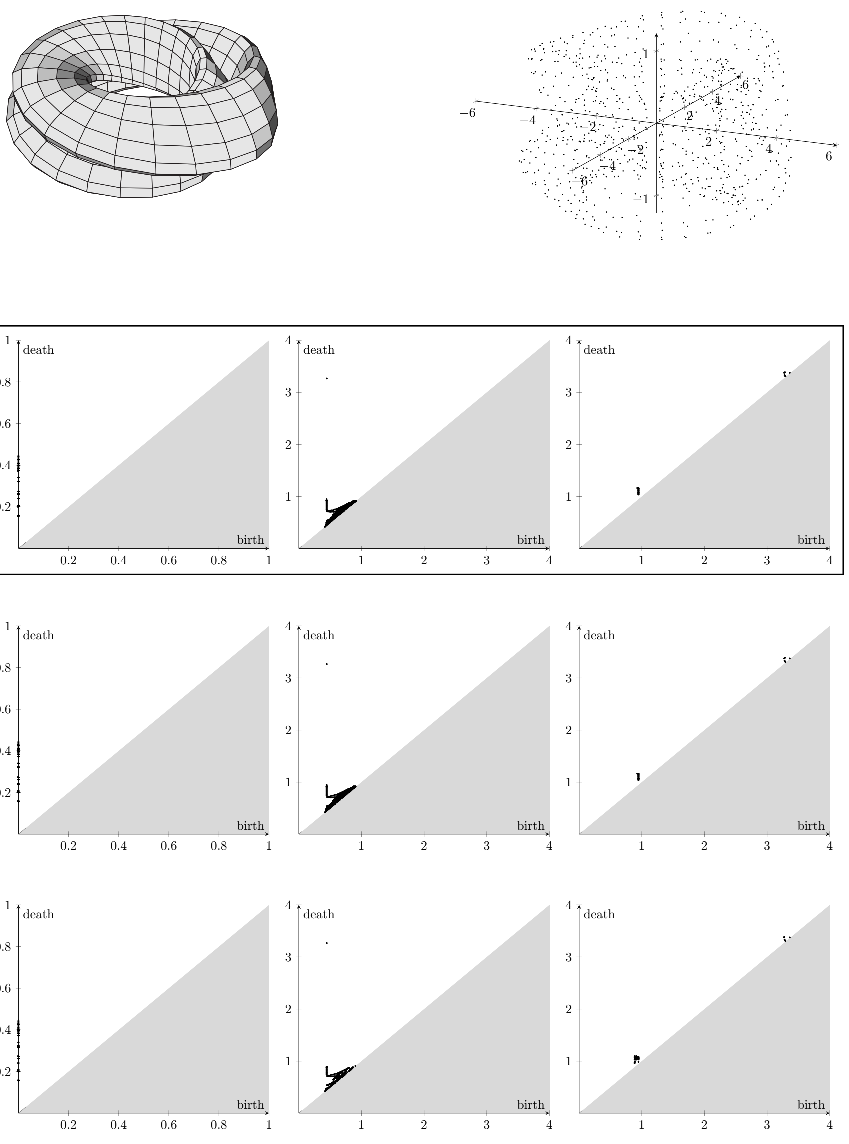


Figure 6 Klein Bottle [19] with persistence diagrams for 0-, 1-, and 2-dim. PH: Ground truth (framed, 900 points), our approach (second row: 743 points; third row: 408 points).

Using Sub-barcodes for Topological Inference[0]

Oliver A. Chubet
North Carolina State University, United States

Kirk P. Gardner
North Carolina State University, United States

Donald R. Sheehy
North Carolina State University, United States

Abstract

Persistent homology is a method in topological data analysis that computes topological invariants with respect to a function $f : X \to \mathbb{R}$. The output is encoded by a set of intervals called the barcode. This work introduces methods of factorization that produce portions of the barcode, which we call a sub-barcode. We show that when only a sample of the data is provided we can obtain a sub-barcode of a c-Lipschitz function by bounding the unknown function by the Lipschitz extensions of the sample. Remarkably, the sub-barcodes computed from these factorization methods are not approximate, they are guaranteed to be contained within the barcode of the unknown function.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases sub-barcodes, factorizations, Lipschitz extensions, persistence modules, persistent homology, TDA

1 Introduction and Background

We introduce sub-barcodes as a new tool for topological inference. The crux of topological data analysis is to characterize a dataset by topological invariants, which are represented as vector spaces. The methods in this work analyze unknown scalar fields. That is, functions $X \xrightarrow{f} \mathbb{R}$, where X is an unknown space. The sublevel sets are the sets $f^{-1}(-\infty, \alpha]$ for $\alpha \in \mathbb{R}$. The nested sequence of sublevel sets for increasing α is referred to as a filtration. We characterize f topologically by its persistence module, which is a parameterization of the topological invariants of the filtration. A persistence module is stored as a barcode, which is a collection of intervals that encode the persistence module. In some works the intervals of the barcode are referred to as topological features.

Denoting the collection of all finite dimensional vector spaces by **vec**, more explicitly, a persistence module is a function $M : \mathbb{R} \to \textbf{vec}$ sending each $t \in \mathbb{R}$ to a space $M_t \in \textbf{vec}$, paired with linear maps $M_{s\leq t} : M_s \to M_t$ for each $s \leq t$ such that $M_{s\leq t} \circ M_{r\leq s} = M_{r\leq t}$. The barcode of a persistence module M is denoted $\mathsf{bar}(M)$. A persistence module homomorphism, $A \xrightarrow{\varphi} B$, is a collection of maps $(\varphi_t : A_t \to B_t)$ such that $\varphi_t \circ A_{s\leq t} = B_{s\leq t} \circ \varphi_s$ for all $s \leq t$.

The goal then, traditionally, given a sample of $S \subset X$ and values of f on S, is to approximate the barcode of the filtration of f. Existing techniques allow one to approximate this barcode such that the quality of the approximation is guaranteed by the quality of the sample [10, 1, 5, 15, 8]. We only require function values on a subsample $P \subset S$. The methods that we introduce do not aim to approximate, but rather extract the portions of the barcode that are known for certain. By constructing a Lipschitz extension of the input sample we induce a factorization of persistence modules, which we show guarantees a sub-barcode.

[0] This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

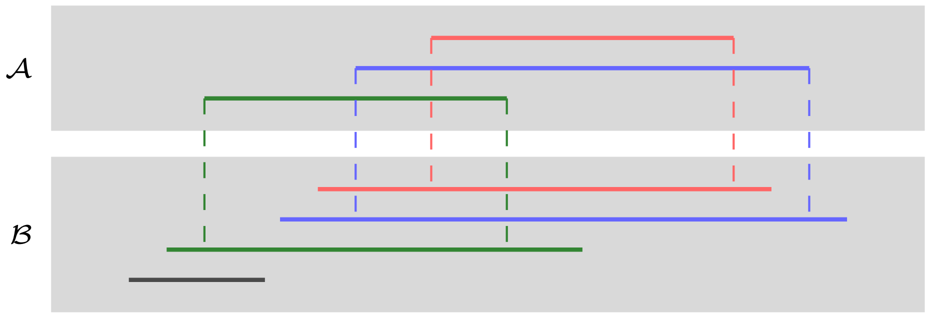


Figure 1 The barcode $\mathcal{A}$ is a sub-barcode of $\mathcal{B}$. The dashed lines indicate an injective mapping from the bars of $\mathcal{A}$ to the bars of $\mathcal{B}$.

2 Sub-barcodes

Given persistence modules A and B, $\mathsf{bar}(A)$ is a sub-barcode of $\mathsf{bar}(B)$ if there exists an injective mapping $\mathsf{bar}(A) \xrightarrow{m} \mathsf{bar}(B)$ such that $\alpha \subseteq m(\alpha)$ for all $\alpha \in \mathsf{bar}(A)$. We denote this relation as $A \sqsubseteq B$. Note that in general m does not correspond to a persistence module homomorphism, however, sub-barcodes do appear naturally in the form of factorizations. A map f factors through g if there exist maps ϕ_1 and ϕ_2 such that $f = \phi_2 \circ g \circ \phi_1$.

▶ **Theorem 1.** *If f and g are persistence module homomorphisms such that f factors through g then $\mathsf{im} f \sqsubseteq \mathsf{im} g$. In particular when g is the identity on a persistence module M, we have $\mathsf{im} f \sqsubseteq M$ [21, Theorem 3.7].*

As a corollary, if we have a map $A \xrightarrow{f} B$, then $\mathsf{im} f \sqsubseteq A$ and $\mathsf{im} f \sqsubseteq B$. This result is powerful in the sense that any factorization of a homomorphism with non-zero image guarantees a portion of the sub-barcode for the images as well as the spaces within the factorization.

3 Sub-barcodes in TDA

The persistent homology of a function $f : X \to \mathbb{R}$ is the persistence module of the filtration of f that uses homology as the topological invariant, denoted $\mathsf{PH}(f)$. Thus $\mathsf{PH}(f)$ is a persistence module, $\mathsf{PH}(f) : \mathbb{R} \to \mathbf{vec}$. Let there be functions $l, g : X \to \mathbb{R}$ such that g is pointwise greater than l, denoted $g \geq l$. Then there is an inclusion of the α-sublevel sets $g^{-1}(-\infty, \alpha] \subseteq l^{-1}(-\infty, \alpha]$ for each α. This induces a persistence module homomorphism $\mathsf{PH}(g) \to \mathsf{PH}(l)$. We abbreviate this map to $\mathsf{PH}(g \hookrightarrow l)$.

▶ **Theorem 2.** *If $g, f, l : X \to \mathbb{R}$ such that $g \geq f \geq l$, then $\mathsf{imPH}(g \hookrightarrow l) \sqsubseteq \mathsf{PH}(f)$ [21, Corollary 4.2].*

Thus if one obtains lower and upper bounds to an unknown function one may infer part of the barcode using the bounding functions. Although, if the bounding functions are not sufficiently close, it is possible that $\mathsf{imPH}(g \hookrightarrow l)$ may be empty. Notably, if l and g are sufficiently close on part of the domain yet arbitrarily far in another part, the portion of the sub-barcode associated to the relevant features will be captured. When the input is a sample we can apply this result by using Lipschitz extensions.

Let $f : X \to \mathbb{R}$ be an unknown c-Lipschitz function, and $P \subseteq X$ a finite sample of a metric space $(X, \mathbf{d})$. The minimum Lipschitz extension is $\hat{f}_P := \max_{p \in P}\{f(p) - c\mathbf{d}(x, p)\}$.

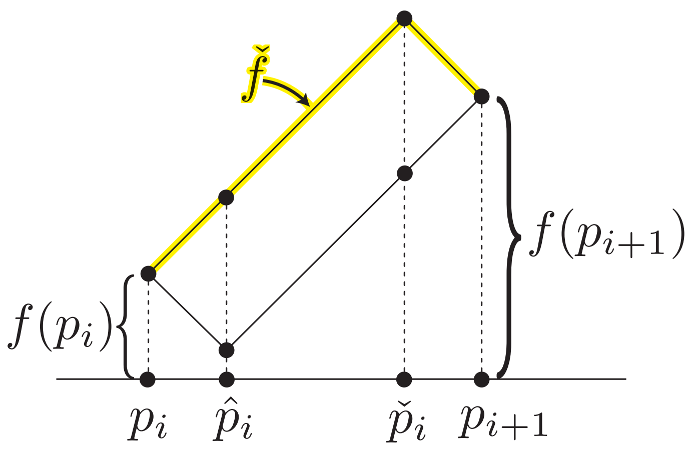


Figure 2 Lipschitz extensions on the line.

The maximum Lipschitz extension is $\check{f}_P := \min_{p\in P}\{f(p) + c\mathbf{d}(x,p)\}$. We conclude with the following corollary.

▶ **Corollary 3.** *Given $f : X \to \mathbb{R}$ and $P \subseteq X$ is finite, then* $\mathsf{imPH}(\hat{f}_P \hookrightarrow \check{f}_P) \sqsubseteq \mathsf{PH}(f)$.

The minimum and maximum Lipschitz extensions bound the unknown function, so there is a factorization of persistence modules induced by the inclusions of their filtrations. Factorizations guarantee sub-barcodes, which allow us to uncover intervals that are guaranteed to be contained within the unknown barcode.

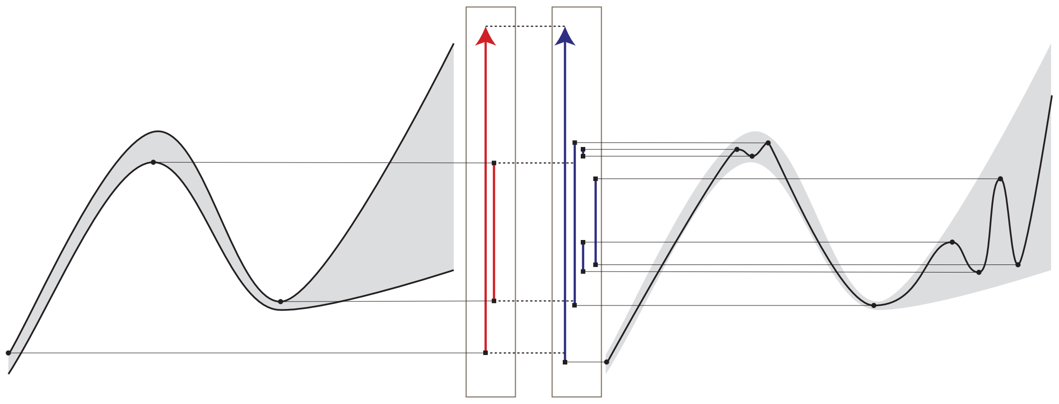


Figure 3 On the left, two functions are depicted, one is an upper bound and the other is a lower bound on an unknown function f. There is a corresponding barcode associated with the pair that matches minima in the upper bound to maxima in the lower bound. On the right is a candidate function f that lies between the upper and lower bounds and its barcode $\mathbb{B}_f$. The barcode of the inclusion of the upper and lower bounds is a sub-barcode of $\mathbb{B}_f$.

References

1 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, page Vol. 6 No. 2 (2015): Special issue of

Selected Papers from SoCG 2014, 2015. URL: `https://jocg.org/index.php/jocg/article/view/2983`, `doi:10.20382/JOCG.V6I2A9`.

2 Ulrich Bauer and Michael Lesnick. Persistence diagrams as diagrams: A categorification of the stability theorem. In *Topological Data Analysis*, pages 67–96. Springer, 2020.

3 Ulrich Bauer and Maximilian Schmahl. Efficient computation of image persistence. *arXiv preprint arXiv:2201.04170*, 2022.

4 Peter Bubenik, Vin De Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015.

5 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 237–246, New York, NY, USA, 2009. ACM. URL: `http://doi.acm.org/10.1145/1542362.1542407`, `doi:10.1145/1542362.1542407`.

6 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Scalar field analysis over point cloud data. *Discrete & Computational Geometry*, 46(4):743–775, 2011.

7 Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in euclidean spaces. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, pages 232–241, New York, NY, USA, 2008. ACM. URL: `http://doi.acm.org/10.1145/1377676.1377719`, `doi:10.1145/1377676.1377719`.

8 Frederic Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. The structure and stability of persistence modules, 2013. `arXiv:1207.3674`.

9 Oliver A. Chubet, Kirk P. Gardner, and Donald R. Sheehy. A theory of sub-barcodes, 2022. URL: `https://arxiv.org/abs/2206.10504`, `doi:10.48550/ARXIV.2206.10504`.

10 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, Jan 2007. `doi:10.1007/s00454-006-1276-5`.

11 David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1011–1020. SIAM, 2009.

12 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.

13 Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002. `doi:10.1007/s00454-002-2885-2`.

14 Herbert Edelsbrunner, Grzegorz Jabłoński, and Marian Mrozek. The persistent homology of a self-map. *Foundations of Computational Mathematics*, 15(5):1213–1244, 2015.

15 Shaun Harker, Miroslav Kramar, Rachel Levanger, and Konstantin Mischaikow. A comparison framework for interleaved persistence modules, 2018. `arXiv:1801.06725`.

16 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

17 Benoît Hudson, Gary L Miller, Steve Y Oudot, and Donald R Sheehy. Topological inference via meshing. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 277–286, 2010.

18 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015.

19 Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

20 Donald R Sheehy. Linear-size approximations to the vietoris–rips filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013.

21 Oliver A. Chubet, Kirk P. Gardner, and Donald R. Sheehy. A theory of sub-barcodes, 2022. URL: `https://arxiv.org/abs/2206.10504`, `doi:10.48550/ARXIV.2206.10504`.

Persistence of complements of spanning trees

Fritz Grimpen
Institute for Algebra, Geometry, Topology and their Applications (ALTA), Department of Mathematics, University of Bremen, 28334 Bremen, Germany

Anastasios Stefanou
Institute for Algebra, Geometry, Topology and their Applications (ALTA), Department of Mathematics, University of Bremen, 28334 Bremen, Germany

Abstract

Given an $\mathbb{N}^n$-filtration $X^\bullet = (X^q)_{q \in Q}$ of simplicial complexes, we consider the problem of explicitly constructing generators of the persistent homology group $H_1(X^\bullet)$ with integer coefficients. We propose the consideration of spanning trees as a tool to identify such generators by introducing a condition for persistent spanning trees, which is accompanied by an existence result for a global, canonical choice of spanning trees.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Topological data analysis, multiparameter persistence, spanning trees, upper set decompositions

1 Introduction

A problem that has recently emerged in topological data analysis (TDA) is the efficient computation of multiparameter persistent homology modules [3, 6, 11]. Recall that a *simplicial complex* X is a set of finite sets such that $\sigma \in X$ implies $\tau \in X$ for every subset $\tau \subseteq \sigma$. Denote by X_n the set of n-simplices of X. For a partially ordered set $(Q, \leq)$, a *(multi-)filtration* $X^\bullet$ of a complex $\mathbb{X}$ over Q is a family $(X^q)_{q \in Q}$ of subcomplexes of $\mathbb{X}$ satisfying $X^q \subseteq X^{q'}$ whenever $q \leq q' \in Q$.

Given such filtration $X^\bullet = (X^q)_{q \in Q}$ of simplicial complexes, one typically seeks a description of the persistent homology module $H_n(X^\bullet) = (H_n(X^q))_{q \in Q}$, $n \in \mathbb{N}$, in terms of a free resolution or, equivalently, generators and relations, which are represented as syzygies [2, 4, 5, 8, 9]. For practical reasons, both representations should be minimized.

In our work, we propose to determine the generators for the multiparameter persistent homology group $H_1(X^\bullet) = H_1(X^\bullet; \mathbb{Z})$ by using *spanning trees*. Spanning trees are known to provide minimal generators for $Z_1(X) = \operatorname{Ker}(\partial_1 \colon C_1(X) \to C_0(X))$ in the static case, as the following theorem, which we shall extend to the persistent setting, asserts.

▶ **Theorem 1** ([7, Theorem 2.33]). *Recall that the relative chain group $C_1(X, T)$ is the quotient $C_1(X)/C_1(T)$. If X is a simplicial complex and $T \subseteq X$ a corresponding spanning tree of X, then $j \colon Z_1(X) \to C_1(X, T)$, $z \mapsto z + C_1(T)$ is an isomorphism.*

2 Persistence of spanning trees

Spanning trees are subcomplexes of a complex that reflect the connectivity properties of the original complex while retaining acyclicity in higher dimensions. We begin with a generalized notion of spanning trees, also known as *spanning forests* or *maximal spanning forests* [1, 7].

This is an abstract of a presentation given at CG:YRF 2023. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.

▶ **Definition 2.** *Let X be a simplicial complex. A 1-dimensional simplicial subcomplex $T \subseteq X$ is a* spanning tree *of X if $T_0 = X_0$ and if it satisfies the* maximal acyclicity condition*: The simplicial complex T is acyclic, and whenever e is an edge in the complement of T, then the simplicial complex $T \cup \{e\}$ is not acyclic.*

An important consequence of Theorem 1 is the possibility to explicitly construct a minimal set of generators of $Z_1(X)$ from a spanning tree T of X. A basis B of $C_1(X, T)$ is given by a choice of (oriented) simplices in X that are not in T [10]. Thus, a basis of $Z_1(X)$ is given by $j^{-1}(B)$, where $j \colon Z_1(X) \to C_1(X, T)$ is the isomorphism from Theorem 1, which consists of *fundamental cycles* in X.

One might hope that given a filtration $(X^q)_{q\in Q}$ there exists a subfiltration $(T^q)_{q\in Q}$ of degreewise spanning trees, i.e. $T^q \subseteq X^q$ and $T^q \subseteq T^{q'}$ for all $q \le q' \in Q$. In this case, Theorem 1 extends to persistence.

▶ **Proposition 3.** *Given a filtration $(T^q)_{q\in Q}$ of degreewise spanning trees of a filtration $(X^q)_{q\in Q}$, there is an isomorphism of persistence modules $(Z_1(X^q))_{q\in Q} \cong (C_1(X^q, T^q))_{q\in Q}$, which is degreewisely given by $j^q \colon Z_1(X^q) \to C_1(X^q, T^q)$, $z \mapsto z + C_1(T^q)$.*

However, existence of a subfiltration $(T^q)_{q\in Q}$ fails spectacularly if Q is not totally ordered, and simple counterexamples exist for $Q = \mathbb{N}^2$ [4, Example 5.1]. Therefore, we loosen the condition $(T^q)_{q\in Q}$ being a filtration to something weaker.

To do so, we impose a well-ordering $\preceq$ on $\mathbb{X}$ and restrict it to well-orderings $\preceq^q$ on each X^q, for $q \in Q$. Then in each degree $q \in Q$ consider the family $\mathcal{T}^q$ of spanning trees of X^q. If X^q is finite, the set $\mathcal{T}^q$ is well-ordered by the lexicographic order $\preceq^q_{\text{lex}}$ induced by $\preceq^q$, given by $T \preceq^q_{\text{lex}} T'$ whenever the minimum of the symmetric difference of T_1 and T'_1 lies in A, and we may choose the minimal $T^q_* \in \mathcal{T}^q$, called the *order-minimal spanning tree*.

If every X^q is finite, the family $(T^q_*)_{q\in Q}$ of degreewise order-minimal spanning trees of a filtration $(X^q)_{q\in Q}$ satisfies the following property and weakened condition as required.

▶ **Theorem 4.** *If $(T^q_*)_{q\in Q}$ is the family of degreewise order-minimal spanning trees of $(X^q)_{q\in Q}$, then it holds $X^q \setminus T^q_* \subseteq X^{q'} \setminus T^{q'}_*$ for all $q \le q' \in Q$.*

3 Generators of persistent homology in dimension one

Given a family $(T^q_*)_{q\in Q}$ of degreewise order-minimal spanning trees of $(X^q)_{q\in Q}$ and an accompanying family $(j^q \colon C_1(X^q, T^q_*) \to Z_1(X^q) \mid q \in Q)$ of isomorphisms defined as in Theorem 1, we can construct a generating morphism of the persistent homology module $H_1(X^\bullet) = (H_1(X^q))_{q\in Q}$ as follows.

First note that there is the epimorphism $(Z_1(X^q))_{q\in Q} =: Z_1(X^\bullet) \to (H_1(X^\bullet)$, defined in each $q \in Q$ by $z \mapsto z + B_1(X^q)$. Thus, for a calculation of generators of $H_1(X^\bullet)$, it suffices to construct them for the persistent cycle module $Z_1(X^\bullet)$.

For every $q \in Q$ choose a basis B^q of $C_1(X^q, T^q_*)$ and consider the filtration of sets $(\bar{B}^q)_{q\in Q}$, defined for each $q \in Q$ as $\bar{B}^q := \bigcup\{j^{q'}(B^{q'}) \mid q' \in Q,\ q' \le q\}$. Taking the degreewise $\mathbb{Z}$-span of $(\bar{B}^q)_{q\in Q}$ and the inclusion maps $\bar{B}^q \hookrightarrow \bar{B}^{q'}$ induces a persistence module $U^\bullet$ together with an epimorphism of persistence modules $f \colon U^\bullet \to Z_1(X^\bullet)$.

▶ **Theorem 5.** *Denote the indicator persistence module of an upper set $U \subseteq Q$ by $\mathbb{Z}[U]$. The persistence module $U^\bullet$ as constructed above is upper set decomposable, i.e. there is a family (U_i) of upper sets of Q such that $U^\bullet = \bigoplus_i \mathbb{Z}[U_i]$. In particular, if $Q = \mathbb{N}^n$, then $U^\bullet$ is finitely generated whenever it decomposes into a finite upper set decomposition.*

4 Conclusion

It is important to note that an arbitrary choice of well-ordering $\preceq$ on $\mathbb{X}$ does not guarantee the minimality of generators for $Z_1(X^\bullet)$. However, for $Q = \mathbb{N}$ there is a natural choice of $\preceq$ such that the constructed generators for $Z_1(X^\bullet)$ from Section 3 are minimal. Further, the described method to construct generators does not depend on the boundaries $B_1(X^\bullet)$, and thus it cannot provide minimal generators for $H_1(X^\bullet)$.

We further conjecture that the computation of the degreewise spanning trees T^q_* can be reduced to the computation of minimum spanning trees of X^q with respect to an appropriate edge weight. This would result in an upper bound for complexity of computing T^q_*.

References

1 J. Adrian Bondy and U. S. R. Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer Berlin, 2008.

2 Magnus Bakke Botnan and William Crawley-Boevey. Decomposition of persistence modules. *Proceedings of the American Mathematical Society*, 148(11):4581–4596, 2020. doi:10.1090/proc/14790.

3 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:10.1007/s00454-009-9176-0.

4 Wojciech Chachólski, Martina Scolamiero, and Francesco Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221(5):1055–1075, 2017. doi:10.1016/j.jpaa.2016.09.001.

5 William Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(5):8, 2015. doi:10.1142/S0219498815500668.

6 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002. doi:10.1007/s00454-002-2885-2.

7 Dmitry N. Kozlov. *Organized collapse. An introduction to discrete Morse theory*, volume 207 of *Graduate Studies in Mathematics*. American Mathematical Society (AMS), 2020. doi:10.1090/gsm/207.

8 Roberto La Scala and Michael Stillman. Strategies for computing minimal free resolutions. *Journal of Symbolic Computation*, 26(4):409–431, 1998. doi:10.1006/jsco.1998.0221.

9 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded betti numbers of 2-parameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 6(2):267–298, 2022. doi:10.1137/20M1388425.

10 Edwin H. Spanier. *Algebraic topology*. Springer, 1982.

11 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005. doi:10.1007/s00454-004-1146-y.