Computing α-Shapes for Temporal Range Queries on Point Sets

Annika Bonerath[1], Jan-Henrik Haunert[1], and Benjamin Niedermann[1]

1 **Institute of Geodesy and Geoinformation, University of Bonn**
`(bonerath,haunert,niedermann)@igg.uni-bonn.de`

Abstract

The interactive exploration of data requires data structures that can be repeatedly queried to obtain simple visualizations of parts of the data. In this paper we consider the scenario that the data is a set of points each associated with a time stamp and that the result of each query is visualized by an α-shape, which generalizes the concept of convex hulls. Instead of computing each shape independently, we suggest and analyze a simple data structure that aggregates the α-shapes of all possible queries. Once the data structure is built, it particularly allows us to query single α-shapes without retrieving the actual (possibly large) point set and thus to rapidly produce small previews of the queried data.

1 Introduction

In scientific projects that deal with large amounts of spatio- and temporal data, the data management is essential. As an example take a project dealing with a database of storm events of the United States; see Figure 1. Each storm event is a data point with a geo-location and a time stamp. Assuming a collection of storm events over several decades the amount of data becomes enormous. On the other hand, for certain scientific questions the user may not be interested in all data, but only in a subset in a pre-defined temporal range. Hence, before

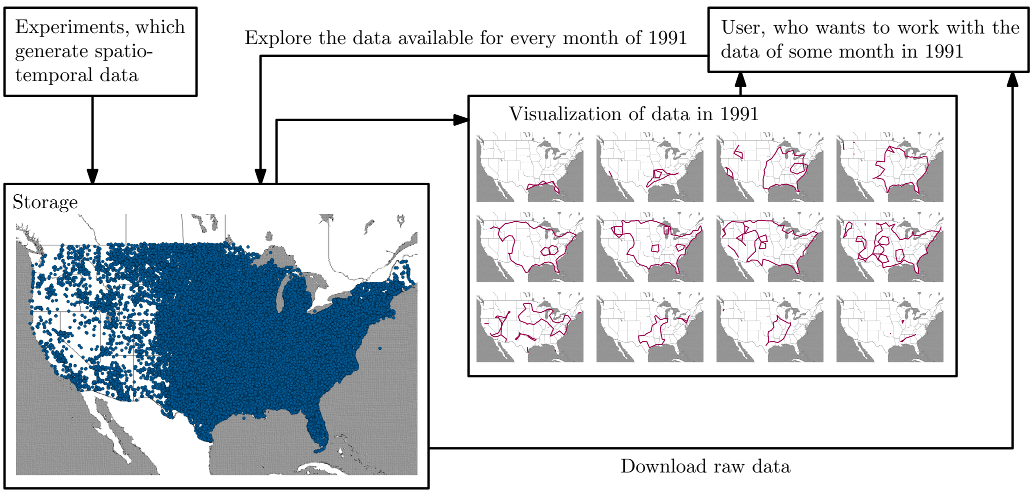

Figure 1 Scenario for the case that the user queries simplified visualizations for all storm events in the year 1991 broken down to months. The α-shapes (lilac) were generated with our approach. Data retrieved from Data.gov. Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 eventually in more final form at a conference with formal proceedings and/or in a journal.

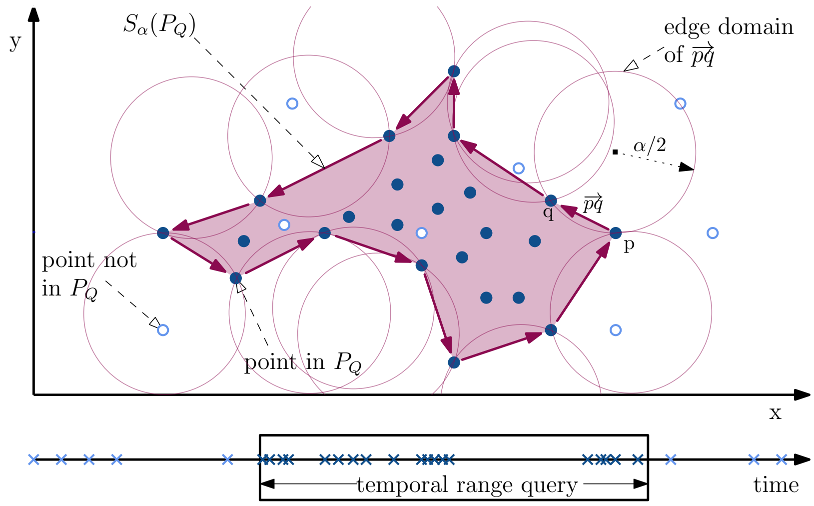

Figure 2 The α-shape (lilac arcs) for a point set (filled blue disks) of a temporal range query.

downloading the actual data for a thorough analysis, the user may be interested in exploring the data by querying simplified visualizations of the data within temporal ranges.

One approach to create a simplified visualization is to sketch the outline of the queried data set providing the user with the possibility of roughly assessing the spatial distribution of the data. For example, the convex hull is a simple polygonal representation for that purpose. However, for most data sets this representation is not adequate, because the convex hull may easily cover large areas that do not contain any points of the data set. A wide range of more sophisticated polygonal representations exists; some of these are based on Delaunay-triangulations and shortest-path graphs [5, 6, 9] while others use spatial grids to define the representation [1, 3, 10, 13]. In this paper we use α-shapes [8, 9] for representing point sets, which are a generalization of convex hulls and strongly related to Delaunay-triangulations. Among others, this technique finds its application in digital shape sampling and processing [2], in pattern recognition [14, 15] and micro-biology [7, 11].

An α-shape of a set $P \subset \mathbb{R}^2$ of n points in the plane is defined as follows. Let $\alpha > 0$. The *edge domain* of a directed edge $pq \in P \times P$ with $|q - p| \leq \alpha$ is the open disk D_{pq} with radius $\frac{\alpha}{2}$ whose center lies to the right of pq and whose boundary contains the points p and q. The set $S_\alpha(P) \subseteq P \times P$ of all edges that are shorter than α and do not contain any point of P in their edge domain is called *α-shape*; see Figure 2. It can be computed in $O(n \log n)$ time [9].

In our use-case each point $p \in P$ additionally is associated with a *time stamp* $t_p \in \mathbb{R}$; we assume that all points in P have pairwise distinct spatial and temporal coordinates. As described in the running example the point set P is queried frequently. Such a *query* Q is a temporal range $[t_Q^{\text{start}}, t_Q^{\text{end}}]$ and its result is the subset $P_Q = \left\{p \in P \mid t_p \in \left[t_Q^{\text{start}}, t_Q^{\text{end}}\right]\right\}$. We are then interested in visualizing P_Q by its α-shape. A straight-forward approach for a query Q first queries the set P obtaining P_Q and then computes the α-shape $S_\alpha(P_Q)$. Utilizing a balanced binary search-tree, finding P_Q takes $O(\log n + |P_Q|)$ time. Additionally computing the α-shape we obtain $O(\log n + |P_Q| \log |P_Q|)$ running time in total. For our use-case of frequently providing α-shapes for visualizing the query results, we aim at a better running time per query. In particular, for creating previews of the data, we only want to retrieve the α-shape of P_Q but not the entire set P_Q. In a pre-processing phase we compute a data structure that aggregates the α-shapes of all possible queries; we call it the *α-structure* of P. We use this data structure in the query phase to obtain the α-shapes of the incoming queries.

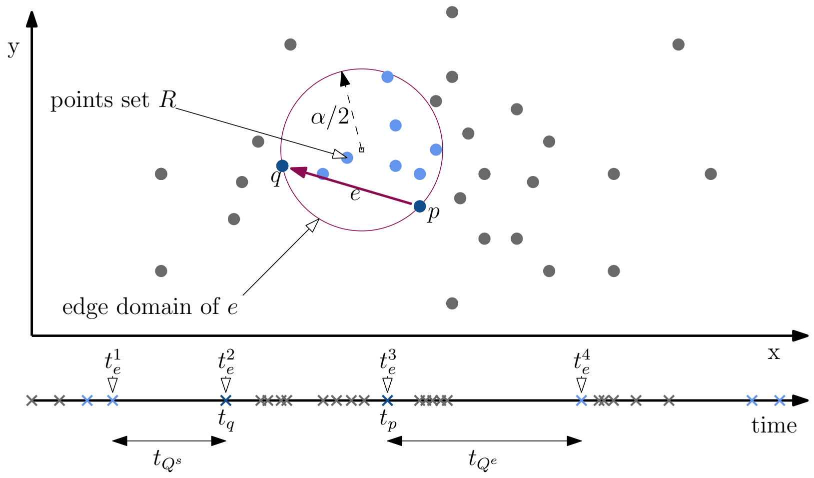

Figure 3 The edge domain with its contained point set R and the time attributes of an edge pq.

As we show in Section 3 the α-structure leads to quadratic memory consumption in the worst case. However, a detailed analysis for points sets whose spatial distribution is uniform and uncorrelated to their temporal distribution shows that the size of the α-structure is more complaisant. In Section 4 we present an algorithm that computes an α-structure in $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$ time utilizing linear and rotational sweeps, where m_{R} denotes the maximum number of points $p \in P$ in a square of width 2α. For the query phase we use a data structure for filtering search to answer a query in $O(\log n + k)$ time, where k is the size of the returned α-shape [4]. In Section 5 we present our initial experiments on real-world data showing that the α-structure is applicable in our concrete use case.

2 On α-Structures

In the following we define the α-structure of P. We say that an edge $pq \in P \times P$ is *active* for a temporal query Q if the α-shape $S_\alpha(P_Q)$ contains pq. We observe that an edge pq can be active for an infinite set of temporal queries, but it can only be active for $O(n^2)$ different subsets of P. To characterize this set, we introduce the following notation; see Figure 3.

Let $e = pq \in P \times P$ with $t_p < t_q$, and let $R \subseteq P$ be the set of points contained in the edge domain of pq. Further, let t_r with $r \in R$ be the largest time stamp that is smaller than t_p; if r does not exist, we set $t_r = -\infty$. The *minimal query start time* is $t_e^1 := t_r$ and the *maximal query start time* is $t_e^2 := t_p$. Similarly, let t_s with $s \in R$ be the smallest time stamp that is greater than t_q; if s does not exist, we set $t_s = \infty$. The *minimal query end time* is $t_e^3 := t_q$ and the *maximal query end time* is $t_e^4 := t_s$. We call t_e^1, t_e^2, t_e^3, t_e^4 the *time attributes* of pq. The next lemma characterizes for which queries a particular edge is active.

▶ **Lemma 2.1.** *The edge $e = pq$ is active for a query Q if and only if:*

1. *The distance between p and q is smaller than α, and*
2. *$\forall r \in R : t_r \notin [t_p, t_q]$, and*
3. *$t_Q^{\text{start}} \in \left[t_e^1, t_e^2\right]$ and $t_Q^{\text{end}} \in \left[t_e^3, t_e^4\right]$.*

Proof. Assume that e is active for Q. This is equivalent to the following three conditions; (i) p and q are contained in P_Q, which is equivalent to $t_p, t_q \in \left[t_Q^{\text{start}}, t_Q^{\text{end}}\right]$, (ii) e is shorter than

α (equivalent to Condition (1) of Lemma 2.1) and (iii) no point $r \in P_Q$ is contained in R, which is equivalent to $\forall r \in R : t_r \notin \left[t_Q^{\text{start}}, t_Q^{\text{end}}\right]$. Applying the definition of t_e^1, t_e^2, t_e^3, and t_e^4 the Conditions (i) and (iii) are equivalent to Condition (2) and (3) of Lemma 2.1. ◀

The α-*structure* $\mathcal{S}_\alpha(P) \subseteq P \times P$ of P is the set of all active edges over all possible temporal queries. We show that Condition (1) and (2) of Lemma 2.1 are necessary and sufficient for an edge pq to be contained in $\mathcal{S}_\alpha$.

▶ **Lemma 2.2.** *The edge $e = pq \in P \times P$ is contained in $\mathcal{S}_\alpha(P)$ if and only if:*

1. *The distance between p and q is smaller than α.*
2. $\forall r \in R : t_r \notin [t_p, t_q]$

Proof. Let $e \in \mathcal{S}_\alpha(P)$, and let Q be a temporal range query for which $e \in \alpha(P_Q)$. Then e fulfills the conditions of Lemma 2.1 and therefore the conditions of Lemma 2.2. Conversely, let e be shorter than α and all points $r \in R$ be temporally not in $[t_p, t_q]$. Then the α-shape of the query Q with $t_Q^{\text{start}} = t_p$ and $t_Q^{\text{end}} = t_q$ contains the edge e. ◀

3 Memory Consumption

For our use-case of a database the memory consumption of our approach is decisive for being deployed in practice. We first observe that $O(n^2)$ is an upper bound for the size of an α-structure. The following theorem shows that this is also a lower bound in the worst case.

▶ **Theorem 3.1.** *For a set P of n points the α-structure has size $\Omega(n^2)$ in the worst case.*

Proof. Let $P = \{p_1, p_2, \dots, p_n\}$ be a point set with time stamps $t_1 < t_2 < \dots < t_n$ such that the points lie on a circle C of radius $r < \frac{1}{2}\alpha$ ordered clockwise according to their time stamps; see Figure 4. Let $p_i, p_j \in P$ be two points with $t_i < t_j$. We show that $p_i p_j$ is contained in the α-structure $\mathcal{S}_\alpha(P)$ by proving the two conditions of Lemma 2.2. Due to $r < \frac{1}{2}\alpha$ the points p_i, p_j have distance smaller than α. Hence, Condition (1) of Lemma 2.2 is satisfied.

For the second condition let R_{ij} be the set of points contained in edge domain D_{ij} of $p_i p_j$. We observe that D_{ij} and C intersect in p_i and p_j. Since the radius of C is smaller than

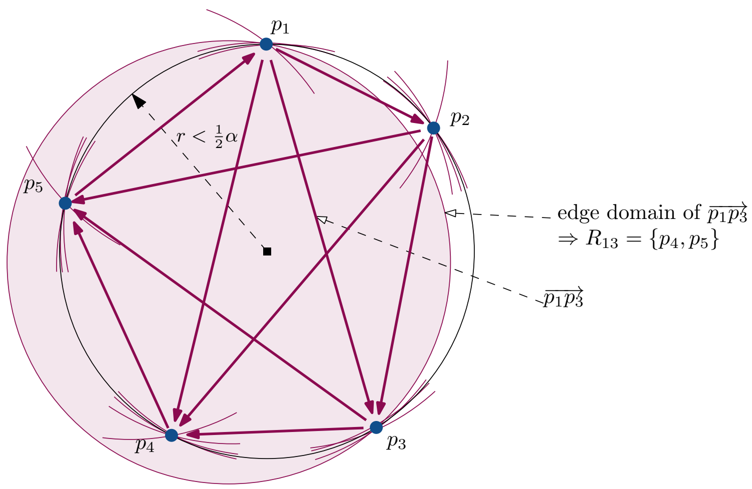

Figure 4 Worst-case example for the size of the α-structure as described in Theorem 3.1.

the radius of D_{ij}, the boundary of D_{ij} partitions C into two parts. One part is contained in D_{ij} and the other lies outside of D_{ij}. Since the points $p_1, p_2, \dots, p_n$ appear in clockwise order on C, and since the center of D_{ij} lies to the right of p_ip_j by definition, we obtain $R_{ij} = \{p_1, \dots, p_{i-1}, p_{j+1}, \dots, p_{n-1}\}$. Consequently, Condition (2) is satisfied. ◀

Hence, the database may exceed a size that is applicable in practice. However, the example is rather unlikely to occur in practice. The next theorem indicates that the data structure is more complaisant than the worst case example suggests. Following our use case we assume that the points are contained in a rectangle $\mathbb{B}$ of width and height at least 2α.

▶ **Theorem 3.2.** *For a finite set $P \subseteq \mathbb{B}$ of n points for which the spatial distribution is uniform in $\mathbb{B}$ and the spatial distance is uncorrelated to the temporal distance the α-structure has expected size $O(n)$.*

To prove Theorem 3.2 we show that the expected size of $\mathcal{S}_\alpha$ is in $O(nm/\kappa)$, where m is the expected number of points in the CPN of a point over all points and κ is the expected number of points in an edge domain over all possible edge domains. Since the area covered by a CPN and an edge domain has a fixed ratio together with a uniform density distribution we can show that m/κ is in $O(1)$. In our experiments on real-world data we also observe a linear relation between the number of points and the size of the α-structure; see Section 5.

4 Constructing and Querying α-Structures

We introduce an algorithm that computes an α-structure of a point set P in $O(n(\log n + m_\mathrm{R} \log m_\mathrm{R}))$ time and describe how to query this data structure. The construction algorithm applies two steps for each point $p \in P$; see Algorithm 1. The first step, which we call *CPN-Search*, computes all points $T_p \subseteq P$ that fulfill Condition (1) of Lemma 2.2, i.e., all points that lie in a circle with center at p and radius α. We call this circle the *circle of potential neighbors (CPN)* of p. We use the sweep line approach by Peng and Wolff [12] to find T_p in $O(\log n + m_\mathrm{R})$ time. The second step, which we call *CPN-Check*, checks for each point $q \in T_p$ whether the edge pq fulfills Condition (2) of Lemma 2.2. If this is the case it computes the time attributes of pq. To implement this efficiently, we use a rotational sweep. More precisely, we use a circle C of radius $\frac{\alpha}{2}$ which sweeps around p such that the center of C moves along the circle with center p and radius $\frac{\alpha}{2}$; see Figure 5. We call C the *sweep circle* of p. Let $\mathcal{R}$ be the points contained in C; we represent $\mathcal{R}$ using a binary search tree ordered by the time stamps of the points. The sweep circle C stops its rotation whenever its boundary intersects with a point $q \in T_p$. Two kind of events are possible; either the point q enters C, or it leaves C. Whenever a point q enters C, the sweep circle equals the edge domain of pq. Utilizing the properties of the binary search tree $\mathcal{R}$ Condition (2) of Lemma 2.2 can be checked in $O(\log m_\mathrm{R})$ time. If this is the case the time attributes of pq can be computed

---

Algorithm 1: Computation of the α-structure

Input: Point set P, parameter α
Output: α-structure $\mathcal{S}_\alpha(P)$
foreach $p \in P$ **do**
  CPN-Search: Find all points $T_p \subseteq P$ in the CPN of p
  CPN-Check: Check for each edge pq with $q \in T_p$ whether it fulfils Condition (2) of Lemma 2.2, possibly compute the time attributes and add to $\mathcal{S}_\alpha(P)$

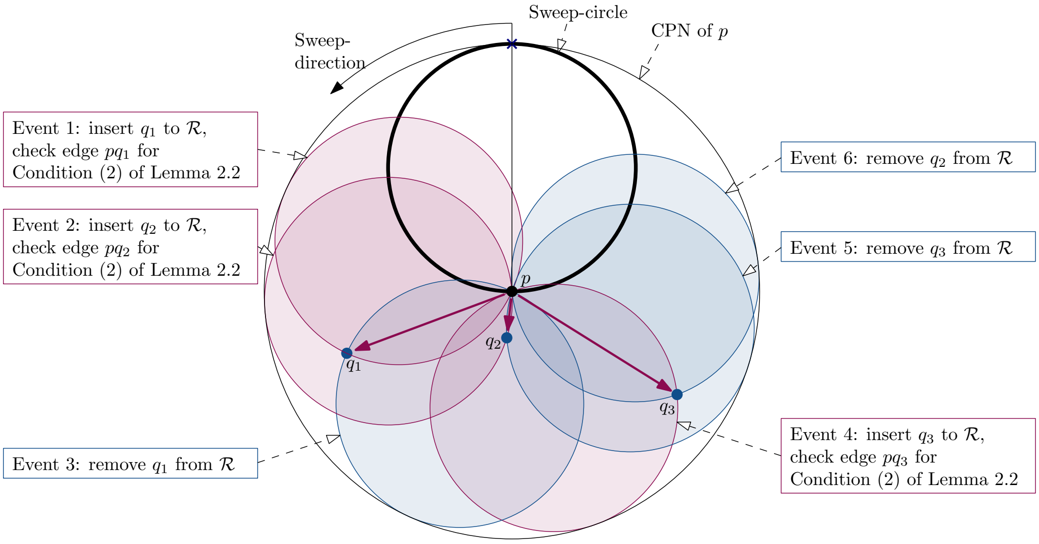

Figure 5 The rotational sweep CPN-Check method for point p and CPN points $T_p = \{q_1, q_2, q_3\}$.

using the temporal order of $\mathcal{R}$ in $O(\log m_{\mathrm{R}})$ time. This rotational sweep can be done in $O(m_{\mathrm{R}} \log m_{\mathrm{R}})$ time. Overall Algorithm 1 has running time $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$.

▶ **Theorem 4.1.** *For a set P of n points the α-structure can be computed in $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$ time, where m_{R} is the maximum number of points in a square with width 2α.*

For the query phase we represent each edge e with time attributes t_e^1, t_e^2, t_e^3 and t_e^4 of the α-structure by a rectangle $[t_e^1, t_e^2] \times [t_e^3, t_e^4]$. A query $[t_Q^{\mathrm{start}}, t_Q^{\mathrm{end}}]$ corresponds to finding all rectangles containing the point $(t_Q^{\mathrm{start}}, t_Q^{\mathrm{end}})$. Using a data structure for filtering search, we can solve this problem in $O(\log n + k)$ time per query, where k is the size of the α-shape [4].

5 Experimental Evaluation

We analyze the performance of α-structures using a data set of storm events in the United States in the years 1991–2000 obtained from Data.gov; see Figure 6 for the year 1991. The experiments[1] indicate that the memory consumption is linear in n; see Figure 7. The construction time for a point set of size $n = 70\,000$ varies between several seconds and hours depending on the value of α; see Figure 7. We assume this to be acceptable, since it is a pre-processing step. Applying the α-structure for temporal range queries the experiments indicate the query time to be nearly constant 200 [ms]; see Figure 8. In contrast an implemented straight forward approach yields results that indicate a dependency to the subset size.

6 Conclusion and Outlook

Overall we presented the design and construction of a data structure that provides the edges of α-shapes for temporal range queries on point sets. For future work we plan to consider other aggregated representations of geographic objects. Further, to reduce memory

[1] Implementation in Java, performed on a 4-core Intel Core i7-7700T CPU with 16 GiB RAM.

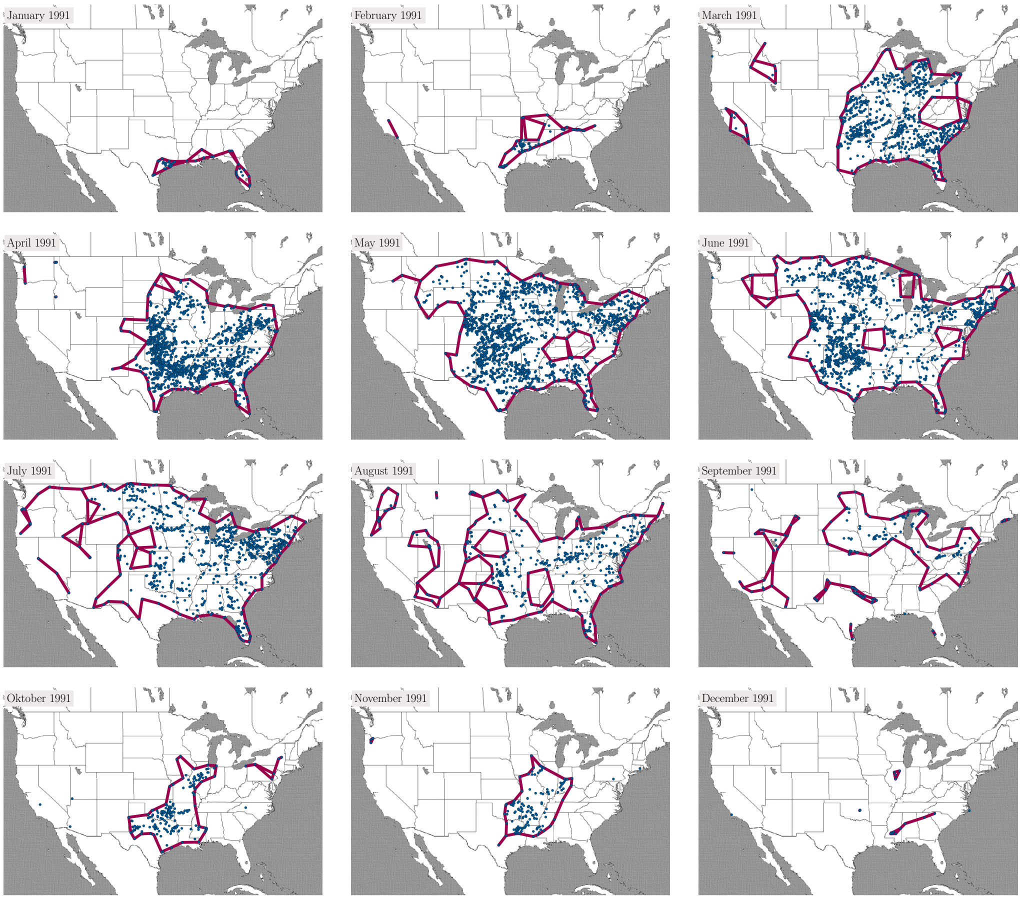

Figure 6 Storm events for the months of 1991 represented by α-shapes (lilac). The actual point set (blue) is drawn for illustration. Data retrieved from Data.gov. Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

consumption we plan to work on an extension that only considers temporally long-lasting α-shape edges.

References

1 Dominique Attali. r-regular shape reconstruction from unorganized points. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 248–253. ACM, 1997.

2 Fausto Bernardini and Chandrajit L. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. Technical Report 97-013, Purdue University, Department of Computer Science, 1997.

3 A. Ray Chaudhuri, Bidyut Baran Chaudhuri, and Swapan K. Parui. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68(3):257–275, 1997.

4 Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal on Computing*, 15(3):703–724, 1986.

5 M.T. de Berg, W. Meulemans, and B. Speckmann. Delineating imprecise regions via shortest-path graphs. In *19th ACM SIGSPATIAL International Symposium on Advances*

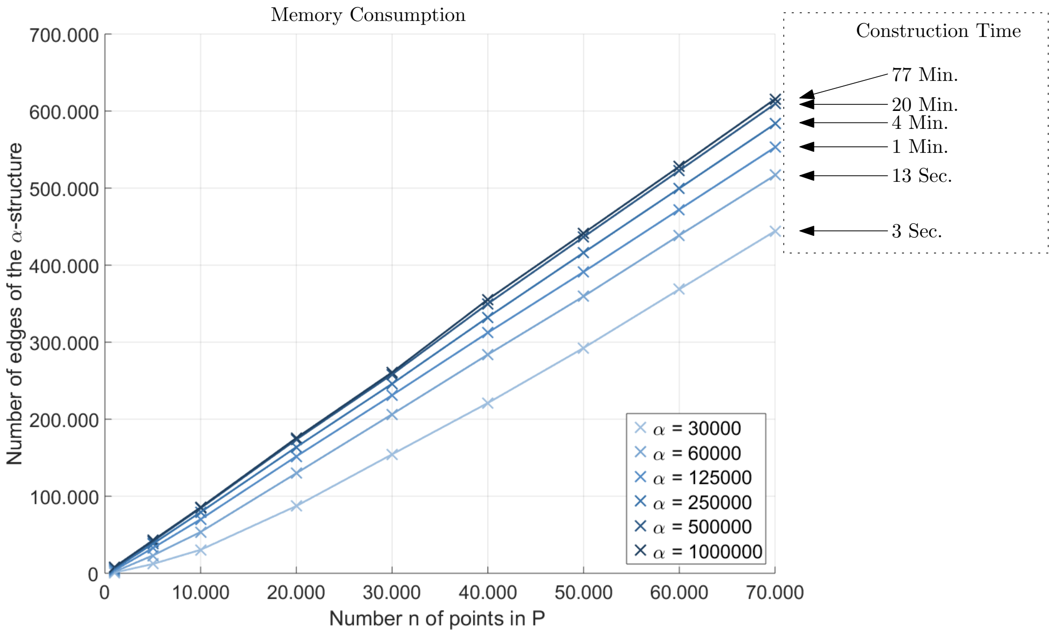

Figure 7 Memory consumption and construction time of an α-structure.

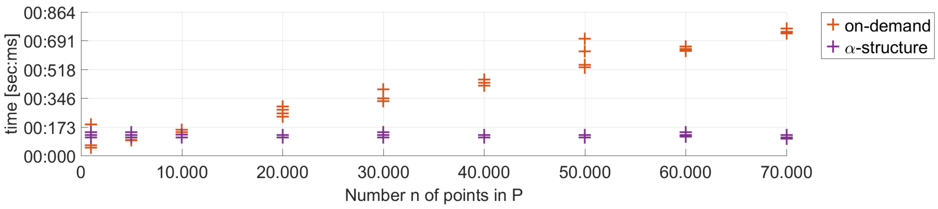

Figure 8 Query phase time of the α-structure compared to a straight forward implementation.

in Geographic Information Systems (ACM GIS), pages 271–280, United States, 2011. Association for Computing Machinery, Inc. `doi:10.1145/2093973.2094010`.

6 Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition*, 41(10):3224–3236, 2008.

7 Herbert Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

8 Herbert Edelsbrunner. Alpha shapes - a survey. *Tessellations in the Sciences*, 27:1–25, 2010.

9 Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983.

10 Christopher B. Jones, Ross S. Purves, Paul D. Clough, and Hideo Joho. Modelling vague places with knowledge from the web. *International Journal of Geographical Information Science*, 22(10):1045–1065, 2008.

11 Jie Liang, Herbert Edelsbrunner, Ping Fu, Pamidighantam Sudhakar, and Shankar Subramaniam. Analytical shape computation of macromolecules: I. molecular area and volume through alpha shape. *Proteins: Structure, Function, and Bioinformatics*, 33(1):1–17, 1998.

12 Dongliang Peng and Alexander Wolff. Watch your data structures. In *Proc. 22th Annu. Geograph. Inform. Sci. Research Conf. UK (GISRUK'14)*, Glasgow, 2014.

13 Ross Purves, Paul Clough, and Hideo Joho. Identifying imprecise regions for geographic information retrieval using the web. In *Proceedings of the 13th Annual GIS Research UK Conference*, pages 313–18, 2005.

14 Jari Vauhkonen, Ilkka Korpela, Matti Maltamo, and Timo Tokola. Imputation of single-tree attributes using airborne laser scanning-based height, intensity, and alpha shape metrics. *Remote Sensing of Environment*, 114(6):1263–1276, 2010.

15 Jari Vauhkonen, Timo Tokola, Petteri Packalén, and Matti Maltamo. Identification of scandinavian commercial species of individual trees from airborne laser scanning data using alpha shape metrics. *Forest Science*, 55(1):37–47, 2009.