Auto-completion of Contours in Sketches, Maps and Sparse 2D Images Based on Topological Persistence

Vitaliy Kurlin
vitaliy.kurlin@gmail.com
Department of Mathematical Sciences
Durham University, Durham, UK, http://kurlin.org

***Abstract*—We design a new fast algorithm to automatically complete closed contours in a finite point cloud on the plane.**

The only input can be a scanned map with almost closed curves, a hand-drawn artistic sketch or any sparse dotted image in 2D without any extra parameters. The output is a hierarchy of closed contours that have a long enough life span (persistence) in a sequence of nested neighborhoods of the input points.

We prove theoretical guarantees when, for a given noisy sample of a graph in the plane, the output contours geometrically approximate the original contours in the unknown graph.

I. Introduction: auto-completion of contours

A. Problem: from a Point Cloud to Closed Contours

Recognizing closed contours is a primary phenomenon according to the Gestalt laws of perception. Humans can easily form closed loops from incomplete contours with gaps. Closed contours bound semantic regions whose extraction is needed for higher level image understanding. So an automatic completion of contours is important for low level vision.

The problem is most similar to Saund [11], who studied how to find perceptually closed paths using weights or preferences for quality criteria. We propose a mathematical definition of persistent contours and suggest a hierarchical solution.

The input is a cloud C, which is a finite set of points with real coordinates in the plane $\mathbb{R}^2$, see Fig. 1. Such a cloud can be a noisy sample or a scan of a hand-drawn sketch or an artistic drawing. In a simple case, C can be a binary image. However C might be sparse without any pixel connectivity.

The output is a hierarchy of most likely closed contours that we can extract from a given finite sample of an original sketch. A *contour* is a non-self-intersecting cycle of straight edges connecting points in the given cloud. So a single output is a union of several contours that bound different regions in the plane. For easier visualization, we present each output by coloring all bounded regions using random colors in Fig. 1.

The algorithm was a bit faster than humans in our experiments for counting most likely closed contours in big clouds with numerous gaps as in Fig. 1. All these closed contours were actually found without using any input parameters like a threshold distance for neighboring points, see section V.

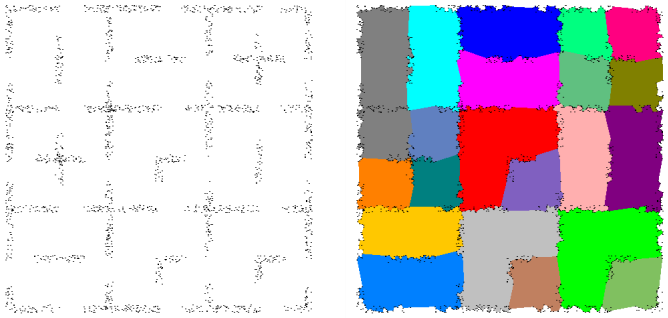

Fig. 1. Input: only a noisy cloud C randomly sampled near a graph $G \subset \mathbb{R}^2$. Output: regions bounded by most persistent contours (best viewed in color).

B. Applications: Maps, Closed Objects, Auto-colorization

In addition to the motivations from Saund [11] we suggest four more applications for the auto-completion of contours.

Map reading. Conventional paper maps contain many level set curves. These curves are often split into disjoint arcs by labels showing the actual height at the points in the curve above the sea level. Hence auto-completion of contours can help digitize paper maps for real-time robot navigation.

Object recognition. Objects in real-life images are often detected by using saliency feature points such as SIFT or SURF. The input can be a noisy and incomplete contour of a person in a video, while the output should be a closed silhouette produced in real-time. So it is important to quickly reconstruct a full boundary from a sparse cloud of features.

Automatic colorization. Artistic drawings are line sketches that often contain gaps, but may give an impression of closed contours. We may enhance these black-and-white drawings by automatically coloring regions that appear visually closed if they are bounded by almost complete contours.

Auto-closure of polygons. A typical difficulty for users of graphics software is to accurately match endpoints of a polygonal line for further painting a resulting bounded region. A quick auto-completion of contours will allow us to suggest an automatic matching of final endpoints on-the-fly.

C. Related Work in Image Segmentation and Graphics

We review only most closely related approaches below.

Pixel-based segmentation is a traditional approach to find boundary contours in an image based on a regular grid of pixels. Such a segmentation usually minimizes an energy function that contains a cost of assigning a single label and a cost of assigning two different labels to neighboring pixels. The resulting minimization problem is often NP-hard. Including higher-order potential between more than two pixels has even larger computational costs and still encodes only local properties. Chen et al. [4] suggested the first algorithm for a binary segmentation with global topological constraints, namely a foreground object is connected and has no holes.

Persistence-based clustering is an excellent method of Skraba et al. [12] using the 0-dimensional persistence. Their problem is to segment a cloud into bases of attraction of a function. Then given points essentially *form* final regions and *do not bound* them as in our problem for finding closed contours. So the inputs in their problem and ours are complementary, which is formalized as a duality between persistence in dimensions 0 and 1 in Lemma 14. Hence a direct comparison of their output with our approach seems impossible.

Input scale parameters are essentially needed for many algorithms including the image segmentation using topological persistence by Letscher and Fritts [10]. Similarly to our method, for a given point cloud C, a Delaunay triangulation is built on C. Then small triangles are merged into persistent regions using two given threshold parameters: α for the radius of disks centered at the points of C, and p for the desired level of persistence. We extend this method and avoid the input parameters by using stability of 1-dimensional persistence.

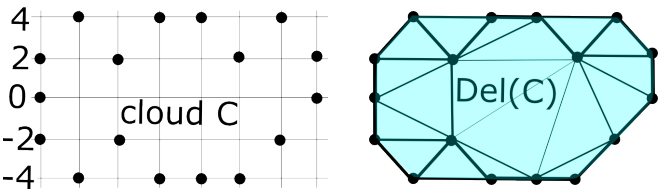

Fig. 2. A cloud C and its (non-unique) Delaunay triangulation $\mathrm{Del}(C)$

Fast 1-dimensional persistence for point clouds in the plane can be computed by the standard algorithm of Attali et al. [1]. This approach was applied by Kurlin [9] for counting topologically persistent holes in noisy clouds. However, the 1-dimensional persistence diagram contains only unstructured data. A segmentation of a cloud requires more information about neighboring relations of persistent regions. So we extend this fast algorithm from [1] by adding a more complicated data structure $\mathrm{Map}(\alpha)$, which allows us to merge all regions into most persistent ones, see section V and Theorem 15.

Homology inference conditions were obtained in many cases to guarantee a correct topological reconstruction from a sample. Our Theorem 20 says that the true number of contours in an unknown graph G can be found from a noisy sample C without using the bound ε of noise, hence doesn't follow from [3, section 5.2]. Indeed, [3, Theorem 4.7] requires counting homology classes that live between given bounds ε and 3ε. Our earlier algorithm [5] for a graph reconstruction from a noisy sample in the plane also used the noise parameter ε.

D. Contributions: Parameterless Algorithm and Guarantees

The key differences between our new approach and the known segmentation methods above are the following.

• We solve the harder problem of completing contours or segmenting a cloud of points with *real coordinates* in $\mathbb{R}^2$.

• The only input is a point cloud *without extra parameters*. The algorithm is unsupervised and requires no training data.

• Quality of contours is measured by topological persistence. Hence all contours form a natural *data-driven hierarchy*.

Arbitrary clouds of points are taken as the input. So our method works for any feature points that can be placed between usual pixels on a regular grid. A hierarchy of closed contours will be produced for any input.

No scale parameters are needed, because we analyze a given cloud at all values of the radius α. The 1st output consists of those contours whose persistence is above a 1st widest gap in a persistence diagram, see Definition 7. The 2nd output has contours with a persistence above a 2nd widest gap etc.

Data-driven measurements are used for quantifying persistence of contours. If a point cloud C lives in a metric space, then we have only a distance function for studying a shape of C. So a natural representation of such a shape is the *offset* C^α, which is a neighborhood of C whose width is a scale parameter $\alpha > 0$. This complicated offset C^α continuously deforms to the simpler α-complex $C(\alpha)$, see Fig. 3, Definition 3, Lemma 4.

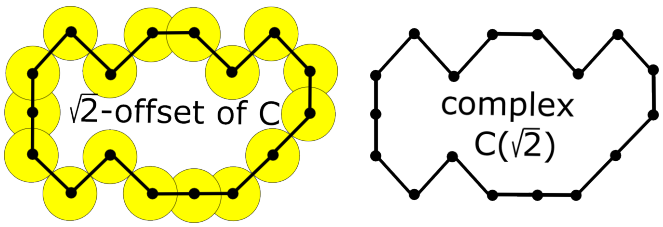

Fig. 3. The α-offset C^α deforms to the α-complex $C(\alpha)$ at $\alpha = \sqrt{2}$.

The persistence of a contour is its life span in the ascending filtration of offsets C^α. When α is increasing, a contour can be *born* at $\alpha = \mathrm{birth}$ and can die in a larger offset at $\alpha = \mathrm{death}$. So the persistence is $\mathrm{death} - \mathrm{birth}$, see Definition 6. For instance one long contour is born in $C^{\sqrt{2}}$, see Fig. 3. Another short contour is born in C^2, see Fig. 4.

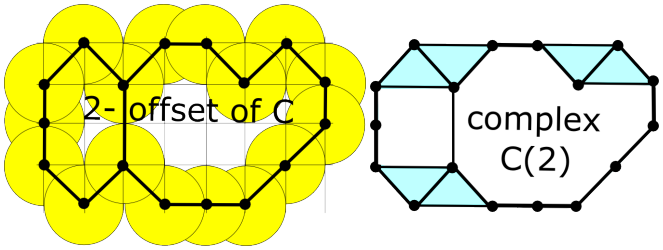

Fig. 4. The α-offset C^α deforms to the α-complex $C(\alpha)$ at $\alpha = 2$.

Here is a high-level description of our key contributions.

• **Fast time**: for any n points in $\mathbb{R}^2$, a hierarchy of closed contours is computed in time $O(n \log n)$, see Theorem 15.

• **Guarantees**: if a cloud C is densely sampled from a good graph $G \subset \mathbb{R}^2$, all contours of G can be geometrically approximated by using only the cloud C, see Theorem 20.

II. Graphs, triangulations and α-complexes

Definition 1 (a plane graph, its cycles and contours): A *plane graph* is a subset $G \subset \mathbb{R}^2$ consisting of finitely many vertices and non-intersecting arcs joining vertices. A *cycle* of G is a subset $L \subset G$ consisting of edges that connect neighboring vertices: p_1 to p_2, p_2 to p_3 and so on until p_k to p_1. A cycle is a closed loop, but may have self-intersections.

A cycle will be called a *contour* if it encloses a connected region in the complement $\mathbb{R}^2 - G$. The boundary of the external region in $\mathbb{R}^2 - G$ is the *boundary contour* of G.

If every bounded region in the complement $\mathbb{R}^2 - G$ of a plane graph is a triangle, then the graph G defines a triangulation on its vertices. The following Delaunay triangulation $\mathrm{Del}(C)$ is a fast and small structure on a cloud $C \subset \mathbb{R}^2$.

Definition 2 (Delaunay triangulation): For a cloud $C = \{p_1, \dots, p_n\} \subset \mathbb{R}^2$ of n points, a *Delaunay triangulation* $\mathrm{Del}(C)$ has all triangles with vertices $p_i, p_j, p_k \in C$ whose circumcircle doesn't enclose any other points of C, see Fig. 2.

A Delaunay triangulation is not unique if C contains 4 points on the same circle. The boundary edges of $\mathrm{Del}(C)$ form the convex $\mathrm{hull}(C)$ of C. The complement $\mathbb{R}^2 - \mathrm{hull}(C)$ will be called the external region. If $\mathrm{Del}(C)$ has k triangles and b boundary edges, then counting E edges over k triangles gives $3k + b = 2E$. By the Euler formula $n - E + (k + 1) = 2$ in the plane, we conclude that $k = 2n - b - 2$, $E = 3n - b - 3$, so $\mathrm{Del}(C)$ has $O(n)$ edges and triangles. Also $\mathrm{Del}(C)$ can be found in time $O(n \log n)$ with $O(n)$ space [2, section 9.1].

A Delaunay triangulation $\mathrm{Del}(C)$ is an example of a general 2-dimensional *complex* consisting of vertices, edges and triangles in $\mathbb{R}^2$. To study the shape of a cloud C at different scales, we shall define subcomplexes that contain the elements of $\mathrm{Del}(C)$ whose sizes are bounded above by a fixed radius α. It will be convenient to re-define a Delaunay triangulation $\mathrm{Del}(C)$ in terms of Voronoi cells, which are neighborhoods of points $p \in C$ and will be used for building the α-complex.

For a point $p_i \in C$, the *Voronoi cell* consists of all points $q \in \mathbb{R}^2$ that are closer to p_i than to all other points of C, so $V(p_i) = \{q \in \mathbb{R}^2 : d(p_i, q) \le d(p_j, q)\ \forall j \ne i\}$. Then a *Delaunay triangulation* $\mathrm{Del}(C)$ consists of all triangles with vertices $p, q, r \in C$ such that $V(p) \cap V(q) \cap V(r)$ is not empty. If the Voronoi cells are restricted to a scale $\alpha > 0$, we get the α-complexes $C(\alpha)$. For any $p \in \mathbb{R}^2$ and $\alpha > 0$, denote by $B(p; \alpha)$ the closed disk with the center p and radius α.

Definition 3 (α-complexes): For a finite cloud $C \subset \mathbb{R}^2$, the *α-complex* $C(\alpha) \subset \mathbb{R}^2$ contains all edges between points $p, q \in C$ such that $V(p) \cap B(p; \alpha)$ meets $V(q) \cap B(q; \alpha)$, see [8, section III.4]. Similarly, the α-complex $C(\alpha)$ contains all triangles with vertices p, q, r such that the full intersection $V(p) \cap B(p; \alpha) \cap V(q) \cap B(q; \alpha) \cap V(r) \cap B(r; \alpha) \ne \emptyset$.

A *hole* of $C(\alpha)$ is a connected region in $\mathbb{R}^2 - C(\alpha)$. The boundaries of all holes are *boundary contours* of $C(\alpha)$.

If $\alpha > 0$ is small, $C(\alpha)$ consists of all isolated points of C. For any large enough α, the complex $C(\alpha)$ is $\mathrm{Del}(C)$. So all α-complexes form a sequence of nested complexes, called a filtration $C = C(0) \subset \dots \subset C(\alpha) \subset \dots \subset C(+\infty) = \mathrm{Del}(C)$. So $\mathrm{Del}(C)$ is built on isolated points of C by adding edges and triangles at the following critical values:

• an edge between points p_i, p_j is added at $\alpha = \frac{1}{2} d(p_i, p_j)$;

• an *acute* triangle (that has all angles less than $\frac{\pi}{2}$) is added at the critical value α equal to the circumradius of the triangle;

• a non-acute triangle is added to $C(\alpha)$ at the scale α that is equal to the half-length of the largest side in the triangle.

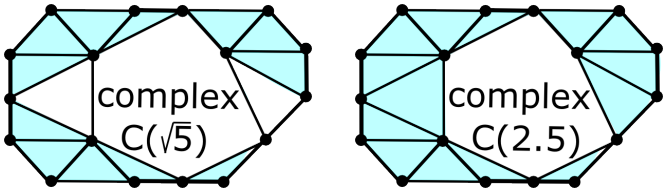

Fig. 5. The α-complexes $C(\alpha)$ of the cloud C in Fig. 2 for $\alpha = \sqrt{5}, 2.5$.

The α-complex $C(\sqrt{5}) \subset \mathbb{R}^2$ in Fig. 5 has 3 holes bounded by 3 independent cycles that generate the group $H_1 = \mathbb{Z}_2^3$. The rightmost hole is the triangle with sides $2\sqrt{2}, 2\sqrt{5}, 2\sqrt{5}$, which are in $C(\alpha)$ at $\alpha = \sqrt{5}$. This acute triangle has the circumradius $R_3 = \frac{5}{3}\sqrt{2} \approx 2.357$ and enters $C(\alpha)$ at $\alpha = R_3$. So the rightmost hole persists only over $\sqrt{5} \le \alpha < R_3$.

The leftmost hole in the complex $C(\sqrt{5}) \subset \mathbb{R}^2$ from Fig. 5 is the triangle with sides $4, 2\sqrt{5}, 2\sqrt{5}$. This hole appears in $C(\alpha)$ earlier at $\alpha = 2$ as a square in Fig. 4. The triangle has the circumradius $R_2 = 2.5$ and enters $C(\alpha)$ at $\alpha = 2.5$. So the leftmost hole persists over the interval $2 \le \alpha < 2.5$.

The following lemma actually motivates the concept of the α-complex, which is a simpler object than the α-offset C^α.

Lemma 4: [7] For any scale α, the α-offset C^α of a cloud $C \subset \mathbb{R}^2$ continuously deforms to the α-complex $C(\alpha) \subset \mathbb{R}^2$.

III. Persistent Homology and its Stability

The persistent homology is a flagship method of topological data analysis. The key idea is to study the evolution of topological invariants in a filtration of complexes $\{S(\alpha)\}$ when a scale α is increasing. For our purposes, the convenient invariant of a complex S is the homology group $H_1(S)$.

Definition 5 (homology group H_1): Cycles of a complex S can be algebraically written as linear combinations of edges with coefficients 0 or 1 in the group $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z} = \{0, 1\}$. The vector space C_1 consists of all these linear combinations. The boundaries of all triangles in S (as cycles of 3 edges) generate the subspace $B_1 \subset C_1$. The quotient C_1/B_1 is the *1-dimensional homology* group $H_1(S)$ with the coefficients $\mathbb{Z}_2$.

For coefficients $\mathbb{Z}_2$, the group $H_1(S)$ is a vector space whose dimension is the first Betti number $\beta_1(S)$ equal to the number of independent loops in S, so $H_1(S) = \mathbb{Z}_2^{\beta_1(S)}$.

The homology group H_1 can be defined for topological spaces that are more general than a complex S in Definition 5. For instance, any α-offset C^α continuously deforms to the α-complex $C(\alpha)$ and has the same homology group H_1.

For simplicity we consider an ascending filtration $\{S_\alpha\}$ of complexes instead of general spaces. Here the scale α is increasing through finitely many critical values $\alpha_1, \dots, \alpha_m$ when $H_1(S(\alpha))$ changes. The inclusions $S(\alpha_1) \subset \dots \subset S(\alpha_m)$ induce the linear maps $H_1(S(\alpha_1)) \to \dots \to H_1(S(\alpha_m))$.

Definition 6 (births and deaths): In a filtration $\{S(\alpha)\}$ of complexes a homology class $\gamma \in H_1(S(\alpha_i))$ is *born* at a time $\alpha_i = \text{birth}(\gamma)$ if γ is not in the image of the map $H_1(S(\alpha)) \to H_1(S(\alpha_i))$ for any $\alpha < \alpha_i$. The class γ *dies* at $\alpha_j = \text{death}(\gamma) \geq \alpha_i$ when the image of γ under $H_1(S(\alpha_i)) \to H_1(S(\alpha_j))$ merges into the image of $H_1(S(\alpha)) \to H_1(S(\alpha_i))$ for some smaller scale $\alpha < \alpha_i$.

The filtration $\{C(\alpha)\}$ of α-complexes for a cloud C in Fig. 2, has 3 pairs $(\text{birth}, \text{death})$ corresponding to 3 life intervals of holes (or their bounding contours). The largest hole is born at $\alpha = \sqrt{2}$ in Fig. 3 and dies only at $\alpha = R_1 = \frac{5}{7}\sqrt{26} \approx 3.642$, which is the circumradius of the largest central triangle in $\text{Del}(C)$, see Fig. 2. The persistence of this hole is $R_1 - \sqrt{2} \approx 2.23$. The two smaller holes in Fig. 5 have lower persistences $R_2 - 2 = 0.5$ and $R_3 - \sqrt{5} \approx 0.12$, see Fig. 6.

All pairs $(\text{birth}, \text{death})$ are usually visualized as points in the persistence diagram on the plane, which is defined below.

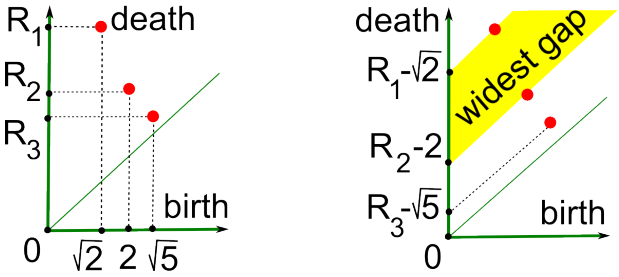

Fig. 6. The persistence diagram of $\{C(\alpha)\}$ for the cloud C in Fig. 2.

Definition 7 (persistence diagram PD*):* For a filtration $\{S(\alpha)\}$ of complexes, let $\alpha_1, \dots, \alpha_k$ be all values of the scale α when the group $H_1(S(\alpha))$ changes. Let μ_{ij} be the number of independent classes in $H_1(S(\alpha))$ that are born at α_i and die at α_j. The *persistence diagram* $\text{PD}\{S(\alpha)\}$ is the multi-set consisting of all points (α_i, α_j) with the multiplicity μ_{ij} and all diagonal points (x, x) with the infinite multiplicity.

Points near the diagonal have a low persistence $\text{death} - \text{birth}$ and are considered as noise. A persistence diagram includes all diagonal so that we can compare diagrams with different numbers of off-diagonal points in Definition 10.

Definition 8 (widest gap): For a cloud $C \subset \mathbb{R}^2$, a *widest* gap in the persistence diagram $\text{PD}\{C(\alpha)\}$ of the filtration of α-complexes is a diagonal strip $\{a < y - x < b\}$ with a largest vertical width $b - a$, see Fig. 6. If there are several widest gaps with the same width, we choose the highest gap.

The widest gap separates pairs $(\text{birth}, \text{death})$ with a high persistence $\text{death} - \text{birth}$ from pairs with a low persistence.

Definition 9 (persistent regions and contours): For a finite cloud $C \subset \mathbb{R}^2$, all m pairs $(\text{birth}, \text{death})$ above the widest gap in $\text{PD}\{C(\alpha)\}$ are called *persistent*. Each persistent pair $(\text{birth}, \text{death})$ corresponds to a region bounded by the new contour in $C(\alpha)$ at $\alpha = \text{birth}$. For each triangle T outside these m regions in $\text{Del}(C)$, we merge T with its neighbor along a longest edge of T. After all mergers the resulting m regions and their boundary contours are called *persistent*.

There is only one pair $(\sqrt{2}, R_1)$ above the widest gap in the persistence diagram in Fig. 6. The corresponding contour coincides with $C(\sqrt{2})$ in Fig. 3. The remaining 3 right-angled triangles outside this contour merge with the external region. So the only contour in $C(\sqrt{2})$ is persistent by Definition 9.

If we consider pairs $(\text{birth}, \text{death})$ above the k-th widest gap for $k > 1$ in Definition 9, we shall get the k-th collection of persistent contours. So the hierarchy of persistent contours corresponds to diagonal gaps in a persistence diagram.

Definition 10 (the bottleneck distance): Let $||p - q||_\infty = \max\{|x_1 - x_2|, |y_1 - y_2|\}$ be the distance between $p = (x_1, y_1)$, $q = (x_2, y_2)$ in $\mathbb{R}^2$. The *bottleneck distance* between persistence diagrams PD, PD' is $d_B = \inf_\psi \sup_{q \in \text{PD}} ||q - \psi(q)||_\infty$ over all bijections $\psi : \text{PD} \to \text{PD}'$ of multi-sets PD and PD′.

Definition 11 (ε-sample): For any $\varepsilon > 0$, a finite cloud C is an ε-sample of a plane graph G if $C \subset G^\varepsilon$ and $G \subset C^\alpha$.

Stability Theorem 12 below is a key foundation of topological data analysis saying that the persistence diagram is stable under perturbations of original data. We quote only a simple version of the Stability Theorem for filtrations of offsets.

Theorem 12 (stability of persistence under noise): [6] If a finite cloud $C \subset \mathbb{R}^2$ of points is an ε-sample of a plane graph $G \subset \mathbb{R}^2$, then we have $d_B(\text{PD}\{G^\alpha\}, \text{PD}\{C^\alpha\}) \leq \varepsilon$.

By Lemma 4 any offset C^α has the same homology group H_1 as $C(\alpha)$. So we may replace the filtration $\{C^\alpha\}$ of offsets by the filtration$\{C(\alpha)\}$ of simpler complexes in Theorem 12.

IV. Duality between α-complexes and α-graphs

We shall analyze the evolution of contours in the 2-dimensional α-complexes $C(\alpha)$ using the simpler filtration of 1-dimensional α-graphs $C^*(\alpha)$ that are dual to $C(\alpha)$.

Let us associate a node v_i to every triangle in $\text{Del}(C)$. We shall call the external region of $\text{Del}(C)$ also a 'triangle' and represent it by an extra node v_0. So v_i are abstract nodes, not geometric centers of Delaunay triangles, though it is convenient to show them as small red centers in Fig. 7.

Definition 13 (α-graphs): The metric graph C^* dual to $\text{Del}(C)$ has the nodes $v_0, v_1, \dots, v_k$ and edges of a length d_{ij} connecting nodes v_i, v_j such that the corresponding triangles in $\text{Del}(C)$ share a common (longest) side of length d_{ij}. Then C^* is filtered by the α-graphs $C^*(\alpha)$ that have only the edges of a length $d_{ij} > \alpha$. Any isolated node v (except v_0) is removed from the graph $C^*(\alpha)$ if the corresponding triangle T_v in $\text{Del}(C)$ is not acute or has a small *circumradius* $\text{rad}(v) \leq \alpha$.

The smallest graph $C^*(+\infty)$ is the isolated vertex v_0 corresponding to the external region in $\text{Del}(C)$. When α drops from 2.5 to $\sqrt{5}$ in Fig. 7, two isolated nodes v_2 and v_3 enter $C^*(\alpha)$,

because $\mathrm{rad}(v_2) = R_2 = 2.5$, $\mathrm{rad}(v_3) = R_3 = \frac{5}{3}\sqrt{2} > \sqrt{5}$. However, these nodes remain isolated in $C^*(\sqrt{5})$ because all sides of their triangles have half-lengths not more than $\sqrt{5}$.

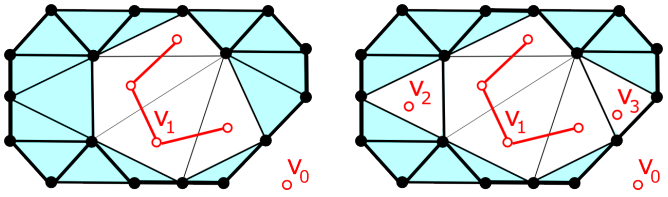

Fig. 7. The complex $C(\alpha)$ and graph $C^*(\alpha)$. Left: $\alpha = 2.5$. Right: $\alpha = \sqrt{5}$.

The ascending filtration of a Delaunay triangulation $\mathrm{Del}(C)$ by α-complexes gives rise to the descending filtration of α-graphs $C^* = C^*(0) \supset \cdots \supset C^*(\alpha) \supset \cdots \supset C^*(+\infty) = \{v_0\}$. Each connected component of $C^*(\alpha)$ has the corresponding region enclosed by a boundary contour of the complex $C(\alpha)$. For instance, if $C(\alpha)$ contains a triangular cycle, but not the enclosed triangle T_v, then the circumradius $\mathrm{rad}(v) > \alpha$ and the corresponding node v belongs to $C^*(\alpha)$.

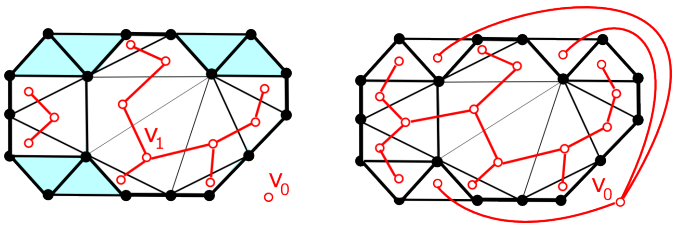

Fig. 8. The complex $C(\alpha)$ and graph $C^*(\alpha)$. Left: $\alpha = 2$. Right: $\alpha = \sqrt{2}$.

So there is a 1-1 correspondence between all isolated nodes (except v_0) of the graph $C^*(\alpha)$ and all triangular boundary contours of the complex $C(\alpha)$. We extend this duality to all components of $C^*(\alpha)$ and all boundary contours of $C(\alpha)$.

Lemma 14 (duality between C and C^*): For any $\alpha > 0$, all components of the α-graph $C^*(\alpha)$ are in a 1-1 correspondence with all boundary contours of the α-complex $C(\alpha)$.

Proof: When the scale α is decreasing, the birth of a boundary contour in the α-complex $C(\alpha)$ means that a small hole appears around the center (of the circumcircle) of an acute triangle T_v in the Delaunay triangulation $\mathrm{Del}(C)$. By Definition 13 at the same time the isolated node v corresponding to the triangle T_v enters the graph $C^*(\alpha)$ as a new component.

The death of a boundary contour in $C(\alpha)$ means that the contour is torn at its longest edge e for $\alpha =$ half-length of e. If the edge e is shared by triangles T_u, T_v, the corresponding nodes u, v are linked, their components merge in $C^*(\alpha)$.

So there is a 1-1 correspondence between births of contours in $C(\alpha)$ and components in $C^*(\alpha)$, and similarly between their deaths. In general, for any fixed value of α, each component of $C^*(\alpha)$ consisting of nodes $v_1, \dots, v_k$ is dual to the contour going along the boundary of the union $T_1 \cup \cdots \cup T_k \subset \mathrm{Del}(C)$ of the triangles represented by the nodes $v_1, \dots, v_k$. ■

By Duality Lemma 14, when α is decreasing from $+\infty$ to 0, the α-complex $C(\alpha) \subset \mathbb{R}^2$ is shrinking, while the α-graph $C^*(\alpha)$ is growing. Initially, $C(+\infty) = \mathrm{Del}(C)$ and we show all triangles of $\mathrm{Del}(C)$ in blue. If a triangle of $\mathrm{Del}(C)$ disappears from $C(\alpha)$ at a critical value α, we show this triangle in white, so eventually all blue triangles become white.

V. Data Structures and our Algorithm

In addition to the data structure $\mathrm{Forest}(\alpha)$ from [9], we introduce $\mathrm{Map}(\alpha)$ that captures all neighboring relations between regions. These relations are not contained in the unstructured persistence diagram. Hence the nice algorithm of Attali et al. [1] for a fast 1-dimensional persistence requires the essential extension to work for a segmentation of clouds.

Following [9], we consider the array $\mathrm{Forest}(\alpha)$ of nodes of the α-graph $C^*(\alpha)$ whose nodes are in a 1-1 correspondence $v \leftrightarrow T_v$ with the triangles of a Delaunay triangulation $\mathrm{Del}(C)$, where the external region of $\mathrm{Del}(C)$ is also called a 'triangle' for convenience. We shall call a node $u \in \mathrm{Forest}(\alpha)$ blue or white according to its corresponding triangle $T_u \subset \mathrm{Del}(C)$. So initially all nodes are isolated and blue. When α is decreasing, the nodes start turning *white* (are born) and join each other to form *white* connected components of $\mathrm{Forest}(\alpha)$.

Merging two components. When two nodes are linked and their white components merge, a younger component dies. Since α is decreasing a component is younger if its first white node was born at a smaller value of α than for the older component. The older component survives by the standard *elder rule* [8, p. 150] that maximizes persistence. The new larger white component contains the dead nodes from the younger component and the *live* nodes (or corresponding *live* triangles) from the older component. Any younger component dies at a smaller value of α than its birth. So the persistence of a white component in $C^*(\alpha)$ and the corresponding boundary contour in $C(\alpha)$ is $\mathrm{pers} = \mathrm{birth} - \mathrm{death} > 0$.

$\mathrm{Forest}(\alpha)$ **of nodes.** Any node v has these attributes:

- $\mathrm{birth}(v) = \sup\{\alpha \mid v \in C^*(\alpha)\}$ when T_v enters $C(\alpha)$;
- $\mathrm{uplink}(v)$ points to a unique parent of the node v;
- $\mathrm{height}(v)$ is the height of $\mathrm{Tree}(v)$ going down from v;
- $\mathrm{live}(v)$ is the list of triangles that are alive in $\mathrm{Tree}(v)$;
- $\mathrm{bar}(v) =$ index of the region in $\mathrm{Map}(\alpha)$ containing T_v.

For each acute triangle T_v in $\mathrm{Del}(C)$, the corresponding node v has $\mathrm{birth}(v) =$ circumradius $\mathrm{rad}(v)$ of T_v. For any non-acute triangle T_v, the node v is linked to an existing component when $\alpha =$ half of the longest edge of T_v, so the triangle T_v merges with its neighbor. Starting from any node v, we come to $\mathrm{root}(v)$ by going up along uplinks until the root node $\mathrm{root}(v)$ points to itself.

Then u, v belong to the same component of $\mathrm{Forest}(\alpha)$ if and only if $\mathrm{root}(u) = \mathrm{root}(v)$ as in a union-find structure. When we need to join two nodes u, v from different components, we actually join their roots by adding a smaller tree to a taller tree. Keeping the height minimal, we guarantee that the root can be reached in $O(\log n)$ steps in a tree of size n.

Any root keeps most important information about its tree. For instance, the birth time for any node v is extracted as $\mathrm{birth}(\mathrm{root}(v))$. To justify the notation bar, we mention that any pair $(\mathrm{birth}, \mathrm{death}) \in \mathbb{R}^2$ can also be considered as the time interval $[\mathrm{birth}, \mathrm{death}] \subset \mathbb{R}$. These intervals or bars form a *bar code*, which is equivalent to the persistence diagram PD. A value $\mathrm{bar}(v) = k$ means that the triangle T_v belongs to the region whose boundary contour was the k-th to die.

$\mathrm{Map}(\alpha)$ **of persistent regions.** Any entry of $\mathrm{Map}(\alpha)$ represents a component c of $C^*(\alpha)$ and has the attributes:

- $\mathrm{ind}(c)$ is the index of c when it was added to $\mathrm{Map}(\alpha)$;
- $\mathrm{birth}(c) =$ scale α when a first node enters c;
- $\mathrm{death}(c) = \alpha$ when c is absorbed by an older component;
- $\mathrm{core}(c)$ points to nodes in c that lived just before c died;
- $\mathrm{heir}(c)$ points to the root of a component that absorbed c;
- $\mathrm{supr}(c)$ is the index of the component rooted at $\mathrm{heir}(c)$.

The input is a set of n points given by real coordinates $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^2$. We start by finding the Delaunay triangulation in time $O(n \log n)$ with $O(n)$ space.

Initialization. We set $\mathrm{birth}(v_0) = +\infty$ for the external node $v_0 \in \mathrm{Forest}(\alpha)$. After finding $\mathrm{Del}(C)$, we go through each triangle T_v of $\mathrm{Del}(C)$ and set the birth of the corresponding node $v \in \mathrm{Forest}(\alpha)$ as the circumradius $\mathrm{rad}(v)$ for acute T_v and $\mathrm{birth}(T_v) = 0$ for non-acute T_v. All $\mathrm{bar}(v)$ have the initial value 0 meaning that the bar indices are undefined. All arrays $\mathrm{live}(v)$ and $\mathrm{Map}(\alpha)$ are empty. We put all edges of the triangles in the decreasing order starting from the longest edge of a length d. We start from the initial largest value $\alpha = \frac{1}{2}d$.

The 'while' loop goes through each edge e of $\mathrm{Del}(C)$ in the decreasing order of length until $\mathrm{Forest}(\alpha)$ becomes connected. Let the edge e is shared by two neighboring triangles T_u, T_v. If α is going down through the critical value equal to the half-length of e, we add the new edge e between the corresponding nodes u, v to the α-graph $C^*(\alpha)$. This addition doesn't affect $\mathrm{Forest}(\alpha)$ if the nodes u, v were already connected, namely they have a common root as in Case 1 below.

It is possible that one of the nodes, say u, was not included in $\mathrm{Forest}(\alpha)$ at the initialization stage, because the corresponding triangle T_u is not acute, so u was blue. In this Case 2 we link the node u to the white component of v. However, it is impossible that both nodes u, v are blue. Indeed, otherwise the triangles T_u, T_v are not acute and share their longest side e. Hence T_v is enclosed by the circumcircle of T_u, which is forbidden in a Delaunay triangulation.

In remaining cases 3 and 4 the nodes u, v are in different components of $\mathrm{Forest}(\alpha)$ that we should merge by linking their roots. At the end of the 'while' loop, $\mathrm{Map}(\alpha)$ contains the full 1-dimensional persistence diagram of the filtration $\{C(\alpha)\}$ and also all neighboring relations of persistent regions.

Case 1: the edge e has the same region on both sides, namely the neighboring triangles T_u, T_v that share the edge v have the same root $\mathrm{root}(u) = \mathrm{root}(v)$. This value of α is not critical, because both u, v are already connected in $\mathrm{Forest}(\alpha)$.

Case 2: e is the longest edge of a non-acute triangle T_u and another triangle T_v whose node v was in $\mathrm{Forest}(\alpha)$. We find $\mathrm{root}(v)$ and add the new node u to $\mathrm{Forest}(\alpha)$ setting $\mathrm{uplink}(u) = \mathrm{root}(v)$, $\mathrm{birth}(u) = \mathrm{birth}(\mathrm{root}(v))$. We increase $\mathrm{height}(\mathrm{root}(v))$ by 1 only if it was 1. If $\mathrm{bar}(v)$ is defined, then v belongs to a dead component. Hence u joins this dead component and we set $\mathrm{bar}(u) = \mathrm{bar}(v)$. If $\mathrm{bar}(v)$ is undefined, T_v is a live triangle, we add u to $\mathrm{live}(\mathrm{root}(v))$.

Creating a new entry in $\mathrm{Map}(\alpha)$**.** In Cases 3-4 we merge two components of $\mathrm{Forest}(\alpha)$ containing the nodes u, v. First we compare the births of $\mathrm{root}(u), \mathrm{root}(v)$ to decide which component is younger, hence dies. If $\mathrm{birth}(\mathrm{root}(u)) \le \mathrm{birth}(\mathrm{root}(v))$, we call the component of u younger. So we can create a new entry c in $\mathrm{Map}(\alpha)$, say with an index i. Namely, we set $\mathrm{ind}(c) = i$, $\mathrm{birth}(c) = \mathrm{birth}(\mathrm{root}(u))$, $\mathrm{death}(c) = \alpha$ (the current value equals the half-length of e). Each node $w \in \mathrm{live}(\mathrm{root}(u))$ dies, we set $\mathrm{bar}(w) = i$.

Then we copy the pointer $\mathrm{live}(\mathrm{root}(u))$ to $\mathrm{core}(c)$ so that the component c knows all its nodes that were alive just before c died. We don't know yet, which entry of $\mathrm{Map}(\alpha)$ will correspond to the *heir* component that is absorbing c now and will die later. Let $\mathrm{heir}(c)$ point to the first triangle from $\mathrm{live}(\mathrm{root}(v))$ in the surviving component, not $\mathrm{live}(\mathrm{root}(u))$, which has just died. After the 'while' loop is finished we can find the index of this heir component as $\mathrm{bar}(\mathrm{heir}(c))$.

Case 3: $\mathrm{height}(\mathrm{root}(u)) \le \mathrm{height}(\mathrm{root}(v))$. Then we link $\mathrm{root}(u)$ of the shorter tree to $\mathrm{root}(v)$ of the taller tree to keep to maximum height of all trees in $\mathrm{Forest}(\alpha)$ minimal, so $\mathrm{root}(v)$ becomes $\mathrm{uplink}(\mathrm{root}(u))$. If the heights were equal, then $\mathrm{height}(\mathrm{root}(v))$ jumps up by 1. All live triangles in the new larger tree at $\mathrm{root}(v)$ came from $\mathrm{live}(\mathrm{root}(v))$.

Case 4: $\mathrm{height}(\mathrm{root}(u)) > \mathrm{height}(\mathrm{root}(v))$. Then we link $\mathrm{root}(v)$ of the shorter tree to $\mathrm{root}(u)$ of the taller tree. Hence $\mathrm{root}(u)$ becomes $\mathrm{uplink}(\mathrm{root}(v))$, but $\mathrm{height}(\mathrm{root}(u))$ remains the same. We should keep all triangles from $\mathrm{live}(\mathrm{root}(v))$ at the root of the new tree, so we replace $\mathrm{live}(\mathrm{root}(u))$ by $\mathrm{live}(\mathrm{root}(v))$. That is why it was important to save $\mathrm{live}(\mathrm{root}(u))$ in $\mathrm{core}(c)$ as we did above.

Final component in $\mathrm{Map}(\alpha)$**.** When $\mathrm{Forest}(\alpha)$ becomes connected, in $\mathrm{Map}(\alpha)$ it remains to add only the final entry c corresponding to the external region. The final root v has the list $\mathrm{live}(v)$ containing the node v_0 and possibly other nodes that didn't die earlier, because they were linked directly to v_0. This list $\mathrm{live}(v)$ is copied to $\mathrm{core}(c)$ as before Cases 3 and 4. We similarly set $\mathrm{birth}, \mathrm{death}$ and $\mathrm{bar}(w)$ equal to the final index of c for any $w \in \mathrm{live}(v)$, but there is no $\mathrm{heir}(c)$.

Initial segmentation. Each triangle from $\mathrm{Del}(C)$ contributes to a single entry of $\mathrm{Map}(\alpha)$, namely all lists $\mathrm{core}(c)$ are disjoint. Hence $\mathrm{Map}(\alpha)$ contains $m = O(n)$ entries and is sorted in time $O(n \log n)$ in the decreasing order of $\mathrm{pers} = \mathrm{birth} - \mathrm{death}$. We output the initial segmentation where all triangles from $\mathrm{core}(c)$ belong to the region having the new sorted $\mathrm{ind}(c)$. We get a segmentation into a smaller number of regions by merging regions of large indices (low persistence) with regions of smaller indices (high persistence).

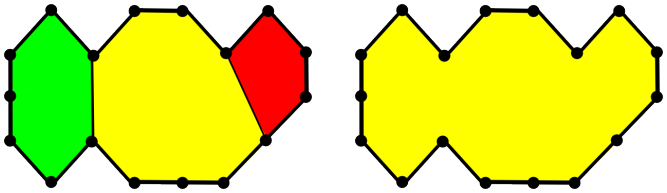

Fig. 9. Initial segmentation: 3 regions. Final segmentation: 1 region.

The widest gap in persistence. We find the maximum gap between persistences of successive entries in sorted $\mathrm{Map}(\alpha)$ in time $O(n)$. In the conditions of Theorem 20, this widest gap separates m pairs $(\mathrm{birth}, \mathrm{death})$ with a high persistence from all other pairs. Even if the conditions of Theorem 20 do not hold, the widest gap gives an approximation to the expected number m of regions for a final segmentation below.

Indices supr **of superior components.** We go through each entry $c \in \mathrm{Map}(\alpha)$ and build the 1-1 correspondence old index $\mapsto$ new index in sorted $\mathrm{Map}(\alpha)$. We go again through each c and access the triangle $\mathrm{heir}(c)$. The bar index $\mathrm{bar}(\mathrm{heir}(c))$ of this triangle is the original (non-sorted) index of the *superior* component that absorbed c after merger. Using the 1-1 correspondence of indices in $\mathrm{Map}(\alpha)$ above, we know the new index $\mathrm{supr}(c)$ of the *superior* or more persistent component (the region with a higher persistence) that absorbed c.

Final segmentation into m **regions.** Now we shall output m regions instead of k in the initial segmentation. For go through all entries c of sorted $\mathrm{Map}(\alpha)$ starting from the least persistent region with the maximum index $k > m$. If the current index $\mathrm{ind}(c) > m$, we add the list $\mathrm{core}(c)$ of the current component c to $\mathrm{core}(\mathrm{supr}(c))$, which enlarges the superior region $\mathrm{supr}(c)$ with a higher persistence. If $\mathrm{ind}(\mathrm{supr}(c)) > m$, the region $\mathrm{supr}(c)$ will also merge with its superior later. After merging all regions of $\mathrm{ind}(c) > m$ with their superiors one by one, we output lists $\mathrm{core}(c)$ of triangles for $\mathrm{ind}(c) = 1, \dots, m$.

VI. Main Theorems 15, 20 and Conclusion

Theorem 15 (fast computation of persistent contours): For any point cloud C of n points in the plane, the algorithm in section V to compute the persistent contours of C has the time complexity $O(n \log n)$ and memory space $O(n)$.

Proof: A Delaunay triangulation $\mathrm{Del}(C)$ for a cloud $C \subset \mathbb{R}^2$ of n points has $k = O(n)$ triangles and is found in time $O(n \log n)$ [2, section 9.1]. The 'while' loop in Algorithm 1 goes once through not more than $O(n)$ edges in $\mathrm{Del}(C)$. We can associate to each edge e its two neighboring triangles $T_u, T_v \subset \mathrm{Del}(C)$ in advance, so identifying the corresponding nodes u, v in line 8 is easy. We shall prove that finding $\mathrm{root}(u), \mathrm{root}(v)$ in line 9 by going along uplinks in any tree of $\mathrm{Forest}(\alpha)$ with k nodes requires $O(\log k) = O(\log n)$ steps.

The height of a tree can increase only in Case 2 (from 1 to 2) or in Case 3, where the height jumps by 1 after we link two trees of the same height. In Cases 3–4 we always link a smaller tree to a taller one. So any two paths from a root to terminal nodes (leaves) can differ by at most 1, where we include all trivial paths consisting of a single node. Hence almost any node is linked to at least nodes one level down,

Algorithm 1 Build $\mathrm{Map}(\alpha)$ of persistent regions in a cloud C

1: Input: a cloud C of n points $(x_1, y_1), \dots (x_n, y_n)$
2: Compute $\mathrm{Del}(C)$ with k triangles on the n points of C
3: Sort edges of $\mathrm{Del}(C)$ in the decreasing order of length
4: $\mathrm{Forest} \leftarrow$ isolated nodes $v_0, \dots, v_k$ with all birth times 0 except $\mathrm{birth}(v_0) \leftarrow +\infty$ and for each acute triangle $T_v \subset \mathrm{Del}(C)$ we update $\mathrm{birth}(v) \leftarrow$ circumradius of T_v
5: Set the number of links in $\mathrm{Forest}(\alpha)$: $l \leftarrow 0$
6: **while** $l < k$ (stop when $\mathrm{Forest}(\alpha)$ becomes a tree) **do**
7: Take the next longest edge e, set $\alpha \leftarrow \frac{1}{2}\mathrm{length}(e)$
8: Find u, v dual to the triangles T_u, T_v that share e
9: Find $\mathrm{root}(u), \mathrm{root}(v)$ going along uplinks from u, v
10: **if** $\mathrm{root}(u) = \mathrm{root}(v)$ **then** Case 1: no changes
11: **else** $\mathrm{birth}(\mathrm{root}(u)) = 0$ and $\mathrm{birth}(\mathrm{root}(v)) > 0$
12: Case 2: run Algorithm 2 below, set $l \leftarrow l + 1$
13: **end if**
14: **if** $\mathrm{birth}(\mathrm{root}(u)) > \mathrm{birth}(\mathrm{root}(v))$ **then** swap u, v,
15: so we assume that the component of u is younger
16: **end if**
17: Create a new entry c, $\mathrm{ind}(c) \leftarrow i = 1+$ old size
18: Set $\mathrm{birth}(c) \leftarrow \mathrm{birth}(\mathrm{root}(u))$, $\mathrm{death}(c) = \alpha$, $\mathrm{core}(c) \leftarrow \mathrm{live}(\mathrm{root}(u))$ and $\mathrm{heir}(c) \leftarrow$ 1st triangle from $\mathrm{live}(\mathrm{root}(v))$, $\mathrm{bar}(w) \leftarrow i$ for each $w \in \mathrm{live}(\mathrm{root}(u))$
19: **if** $\mathrm{height}(\mathrm{root}(u)) \le \mathrm{height}(\mathrm{root}(v))$ **then**
20: Case 3: $\mathrm{uplink}(\mathrm{root}(u)) \leftarrow \mathrm{root}(v)$ and add 1 to $\mathrm{height}(\mathrm{root}(v))$ if $\mathrm{height}(\mathrm{root}(u)) = \mathrm{height}(\mathrm{root}(v))$
21: **else** Case 4: $\mathrm{uplink}(\mathrm{root}(v)) \leftarrow \mathrm{root}(u)$, $\mathrm{live}(\mathrm{root}(u)) \leftarrow \mathrm{live}(\mathrm{root}(v))$
22: **end if**
23: Update the number of links $l \leftarrow l + 1$ in $\mathrm{Forest}(\alpha)$
24: **end while**
25: Create the final entry with the external node v_0
26: Return array $\mathrm{Map}(\alpha)$ generated by Cases 3 and 4

Algorithm 2 Case 2: link the node u to the component of v

1: Set $\mathrm{uplink}(u) \leftarrow \mathrm{root}(v)$, $\mathrm{birth}(u) \leftarrow \mathrm{birth}(\mathrm{root}(v))$
2: **if** $\mathrm{height}(\mathrm{root}(v)) = 1$ **then** $\mathrm{height}(\mathrm{root}(v)) \leftarrow 2$
3: **end if**
4: **if** $\mathrm{bar}(v)$ is already defined **then** set $\mathrm{bar}(u) \leftarrow \mathrm{bar}(v)$
5: **else** Add u to $\mathrm{live}(\mathrm{root}(v))$ in the subtree at $\mathrm{root}(v)$
6: **end if**

except all terminal nodes and some nodes only one level up. Then any tree of height $h \ge 1$ should contain at least 2^{j-1} nodes at level $1 \le j \le h-1$ plus at least one node at level h, so at least $1 + 2 + 2^2 + \dots + 2^{h-2} + 1 = 2^{h-1}$ nodes in total. If $k \ge 2^{h-1}$, then the height is $h \le 1 + \log_2 k = O(\log k)$.

All other steps in the 'while' loop need $O(1)$ time. Then we spend $O(n \log n)$ time for sorting $O(n)$ entries in $\mathrm{Map}(\alpha)$. The lists $\mathrm{live}(v)$ of triangles are disjoint in $\mathrm{Forest}(\alpha)$ as well as similar lists $\mathrm{core}(c)$ in $\mathrm{Map}(\alpha)$. Then $O(n)$ Delaunay edges, triangles or corresponding nodes with attributes need only $O(n)$ space in $\mathrm{Forest}(\alpha)$ and similarly in $\mathrm{Map}(\alpha)$. ■

Let L be any closed non-self-intersecting loop in $\mathbb{R}^2$. We consider all offsets L^α when α is increasing. There is a first critical scale α when the internal boundary of L^α touches itself, so L^α is no longer an annulus. There is a last critical

scale α when the internal boundary of L^α shrinks to a point, so all holes of L^α are filled, which is true for all larger α.

Definition 16: Maps $f_0, f_1 : X \to Y$ are *homotopic* if they can be included into a continuous family of maps $f_t : X \to Y$, $t \in [0,1]$. A set $S \subset \mathbb{R}^2$ is *contractible* to a point $q \in S$ if $\mathrm{id} : S \to S$ is homotopic to $S \to q$. A set $S \subset \mathbb{R}^2$ is *circular* if there is a projection $S \to S^1 \subset S$ homotopic to $\mathrm{id} : S \to S$.

Below we consider closed loops that go along boundaries of regions in $\mathbb{R}^2 - G$. The internal loop of the graph ① goes along the short vertical edge twice, namely up and down.

Definition 17 (simple contours, a simple graph $G \subset \mathbb{R}^2$): A closed contour $L \subset \mathbb{R}^2$ is *simple* if there is a scale $\rho(L)$ called the *radius* such that the α-offset L^α is circular for any $0 \le \alpha < \rho(L)$ and L^α is contractible for any $\alpha \ge \rho(L)$. A graph $G \subset \mathbb{R}^2$ is *simple* if G is connected and the boundaries of all bounded regions in $\mathbb{R}^2 - G$ are simple contours.

Definition 17 means that a simple contour L has only one critical scale α when L^α stops being circular and becomes contractible. So the hole initially enclosed by L^α for small α dies at $\alpha = \rho(L)$ without splitting into other smaller holes.

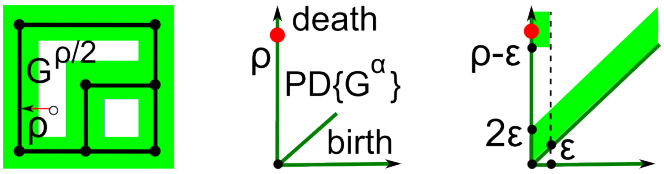

Fig. 10. A simple graph G with $G^{\rho/2}$, $\mathrm{PD}\{G^\alpha\}$ and its ε-neighborhood.

The boundary of any convex region in $\mathbb{R}^2$ is simple, but a simple contour can enclose a non-convex region. A typical non-simple contour is the circular graph $L = C(\sqrt{2})$ in Fig. 3. as the offset $L^2 \subset C(2)$ in Fig. 4 includes the vertical edge that splits the initial hole enclosed by $L = C(\sqrt{2})$ into 2 holes. Lemma 18 says that any simple graph G has the persistence diagram $\mathrm{PD}\{G^\alpha\}$ with points only in the vertical axis.

Lemma 18: Let $G \subset \mathbb{R}^2$ be a simple graph. Denote by $\rho_1, \dots, \rho_m$ the radii of the boundaries of all m bounded regions in $\mathbb{R}^2 - G$. Then the persistence diagram $\mathrm{PD}\{G^\alpha\}$ has exactly m off-diagonal points $(0, \rho_i)$, $i = 1, \dots, m$.

Lemma 19 says that, for any ε-sample C of a simple graph, the diagram $\mathrm{PD}\{C^\alpha\}$ of the filtration of α-offsets has all points ε-close to the diagonal or to the vertical axis, see Fig. 10.

Lemma 19: Let $C \subset \mathbb{R}^2$ be any ε-sample of a simple graph G with radii $\rho_1 \le \dots \le \rho_m$. Then $\mathrm{PD}\{C^\alpha\}$ has m points in the vertical strip $\{\mathrm{birth} \le \varepsilon,\ \mathrm{death} \ge \rho_1 - \varepsilon\}$. All other points are in the diagonal strip $\{0 \le \mathrm{death} - \mathrm{birth} \le 2\varepsilon\}$.

Theorem 20 gives conditions on a simple graph G when our algorithm uses only an ε-sample C of G to find all expected contours that are 2ε-close to the true simple contours.

Theorem 20 (guarantees for persistent contours): Let $G \subset \mathbb{R}^2$ be a simple graph. Denote by $\rho_1 \le \dots \le \rho_m$ the radii of the boundaries of all m bounded regions in $\mathbb{R}^2 - G$ as in Definition 17. If $\rho_1 > 7\varepsilon + \max\limits_{i=1,\dots,m-1}\{\rho_{i+1} - \rho_i\}$, then any finite ε-sample C of the graph G has exactly m persistent contours, which are contained in the 2ε-offset $G^{2\varepsilon} \subset \mathbb{R}^2$.

The cloud C in Fig. 2 can be considered as an ε-sample of the graph $G = C(\sqrt{2})$ in Fig. 3 for $\varepsilon = \sqrt{2}$. This graph G is not a simple contour as two smaller contours are born at $\alpha = 2$ and $\alpha = \sqrt{5}$. The conditions of Theorem 20 do not hold. But our algorithm correctly reconstructs the original graph G in Fig. 9 due to a low persistence of the smaller contours. So the algorithm may work beyond the guarantees of Theorem 20.

The condition $\rho_1 > 7\varepsilon + \max\limits_{i=1,\dots,m-1}\{\rho_{i+1} - \rho_i\}$ implies that the diagonal gap $\{0 < y - x < \rho_1\}$ is the widest in $\mathrm{PD}\{G^\alpha\}$, see Lemma 18 and contains a widest gap in $\mathrm{PD}\{C^\alpha\}$. The diagram $\mathrm{PD}\{C^\alpha\}$ is in the ε-neighborhood of $\mathrm{PD}\{G^\alpha\}$ by Stability Theorem 12, see Lemma 19 and Fig 10. Hence without extra input parameters we can find the widest gap in $\mathrm{PD}\{C^\alpha\}$ and filter out all noise near the diagonal.

Detailed proofs of Lemmas 18, 19 and Theorem 20 and more experiments are in the full version at author's website http://kurlin.org. Here is a summary of our key contributions.

• The $O(n \log n)$ time algorithm from section V accepts any cloud of points in $\mathbb{R}^2$ without extra parameters and outputs a hierarchy of closed contours selected by their persistence.

• Theorem 20 proves that the persistent contours approximate true contours of a graph G given only by a noisy sample C.

Potential further problems include computing persistent cycles in clouds from high-dimensional or metric spaces, and extending Theorem 20 to non-simple graphs and unbounded noise. The author is open to collaboration on any related projects and thanks all reviewers for comments and helpful suggestions.

References

[1] Attali, D., Glisse, M., Hornus, S., Lazarus, F., Morozov, D.: Persistence-sensitive simplification of functions on surfaces in linear time. TopoInVis'09: Topological Methods in Data Analysis and Visualization.

[2] de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer (2008).

[3] Chazal, F., A. Lieutier, A.: Weak feature size and persistent homology: computing homology of solids in $\mathbb{R}^n$ from noisy data samples. Proceedings of SoCG 2005: Symposium on Computational Geometry.

[4] Chen, C., Freedman, D., Lampert, C.: Enforcing topological constraints in random field image segmentation. Proceedings of CVPR 2011: Computer Vision and Pattern Recognition, 2089–2096.

[5] Chernov, A., Kurlin, V.: Reconstructing persistent graph structures from noisy images. Image-A, v. 3 (2013), no. 5, p. 19-22.

[6] Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discrete and Computational Geometry **37** (2007) 103–130.

[7] Edelsbrunner, H.: The union of balls and its dual shape. Discrete Computational Geometry **13** (1995) 415–440.

[8] Edelsbrunner, H., Harer, J.: Computational topology. An introduction. AMS, Providence, Rhode Island (2010).

[9] Kurlin, V.: A fast and robust algorithm to count topologically persistent holes in noisy clouds. In Proceedings of CVPR 2014: Computer Vision and Pattern Recognition. Available at http://arxiv.org/abs/1312.1492.

[10] Letscher, D., Fritts, J.: Image segmentation using topological persistence. Proceedings of CAIP 2007: Computer Analysis of Images and Patterns, 587–595.

[11] Saund, E.: Finding Perceptually Closed Paths in Sketches and Drawings. Transactions PAMI, v. 25 (2003), 475-490.

[12] Skraba, P., Ovsjannikov, M., Chazal, F., Guibas, L.: Persistence-based Segmentation of Deformable Shapes. Proceedings of CVPR 2010 NORDIA Workshop on Deformable Shape Analysis, 2146–2153.