Computing Robustness and Persistence for Images

Paul Bendich, Herbert Edelsbrunner, and Michael Kerber

Abstract—We are interested in 3-dimensional images given as arrays of voxels with intensity values. Extending these values to a continuous function, we study the robustness of homology classes in its level and interlevel sets, that is, the amount of perturbation needed to destroy these classes. The structure of the homology classes and their robustness, over all level and interlevel sets, can be visualized by a triangular diagram of dots obtained by computing the extended persistence of the function. We give a fast hierarchical algorithm using the dual complexes of oct-tree approximations of the function. In addition, we show that for balanced oct-trees, the dual complexes are geometrically realized in $\mathbb{R}^3$ and can thus be used to construct level and interlevel sets. We apply these tools to study 3-dimensional images of plant root systems.

Index Terms—Voxel arrays, oct-trees, persistent homology, persistence diagrams, level sets, robustness, approximations, plant roots.

1 Introduction

The last decade has witnessed the development of persistent homology from a mathematical idea to a full-blown methodology for capturing essential features of data in a robust manner; see eg. the recent textbook [10]. Datasets to which persistent homology has been successfully applied include natural images [5], trademark images [6], sensor networks [8], protein structures [20], gene expression profiles [9], and brain structures [7]. These and other applications are facilitated by efficient algorithms and software for computing persistence for filtered complexes. While efficient, the algorithms are not yet fast enough to support broad applications to 3-dimensional image data consisting of millions if not billions of voxels.

Our first contribution is a fast algorithm for computing persistence for 3-dimensional images. It uses standard image processing tools, such as oct-trees [17, 18], and produces the persistence information in a series of progressively more accurate approximations. An important step here is the construction of a simplicial complex that serves as input to the persistence algorithm. We analyze under which conditions this complex is geometrically realized in $\mathbb{R}^3$; if these conditions are satisfied, we may use versions of the marching cube algorithm [15] to compute approximations of level sets. The output of our algorithm is a converging series of multi-sets of dots in the plane, referred to as *persistence diagrams*, where each diagram gives an easy-to-read encoding of the homology of all level sets and interlevel sets of an oct-tree approximation of the image, as well as the robustness of this information under perturbations of the input data. The theoretical underpinnings of this representation of homological robustness can be found in [3, 11]. The second contribution is the detailed analysis of the approximations of the diagram and the robustness information obtained for piecewise constant and piecewise linear approximations of the image data. These results quantify the convergence of the information provided in a series of refinements by the hierarchical algorithm. The third contribution is the application of the new tools to analyze root systems of agricultural plants.

As mentioned above, the persistence diagram of a 3-dimensional image is a compact representation of the families of level and interlevel sets. It can therefore be compared with other summary representations of level sets, such as Reeb graphs [16, 19] and the contour spectrum [2]. While both contain information not readily available from the persistence diagrams, the diagrams go beyond them by displaying topological information of dimension higher than 0 as well as the robustness of all information under perturbations of the input data. We believe the latter to be a generally useful tool for the analysis of level sets; possible applications are the selection or the simplification of level sets, which have been studied with more specialized approaches in [1, 12]

Outline. Section 2 gives the necessary background. Section 3 explains an algorithm for computing the persistence diagram of a function defined via an oct-tree. Section 4 analyzes the resulting approximations of persistence and robustness. Section 5 applies the tools to root system data and reports on the run-time characteristics of our software. Section 6 concludes the paper.

2 Background

In this section, we introduce background from topology, in particular homology groups and persistence, as well as terminology on images and their representation as oct-trees.

Images and oct-trees. We think of an *image* as a real-valued function. Since we are primarily interested in the 3-dimensional case, we write $f : \mathbb{R}^3 \to \mathbb{R}$. In practical applications, we can know the value of the function only at a finite number of points. We are interested in the important special case in which this set contains the n^3 integer points in a cube. A *voxel* is a cube with unit side-length centered at such an integer point. Since all we know about the function are its values at the n^3 integer points, we may as well use the voxels to construct a first piecewise-constant approximation. Specifically, we define $f_{\text{vox}} : \mathbb{R}^3 \to \mathbb{R}$ by setting $f_{\text{vox}}(x) = f(i, j, k)$ if x lies inside the voxel centered at the integer point (i, j, k), assuming some tie-breaking mechanism, and $f_{\text{vox}}(x) = 0$ if x is not contained in any of the n^3 voxels.

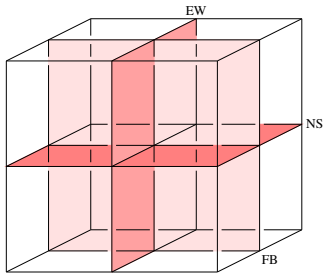

Fig. 1: The east-west, north-south, and front-back planes decompose a node in the oct-tree into its eight children.

We get additional piecewise constant approximations of f from partial hierarchical representations of the array of voxels, which we now describe. The type of hierarchies we have in mind are called *oct-trees*.

- *Paul Bendich is with IST Austria. E-mail: bendich@ist.ac.at.*
- *Herbert Edelsbrunner is with IST Austria, Duke University, and Geomagic. Email: edels@cs.duke.edu.*
- *Michael Kerber is with IST Austria. Email: mkerber@ist.ac.at.*

- *This research is partially supported by the National Science Foundation (NSF) under grant DBI-0820624 and the Defense Advanced Research Projects Agency (DARPA) under grants HR0011-05-0057 and HR0011-09-0065.*

Assume $n = 2^\ell$. The *root node* of an oct-tree is the union of the n^3 voxels. It has either zero or eight children, and in the latter case each *child* is the root node of an oct-tree for the corresponding smaller array of voxels. The *level* of a node in this tree is $\lambda = 0$ for the root node and $\lambda + 1$ for the child of a node at level λ. A node of the oct-tree is *internal* if it has eight children, as in Figure 1, and it is *external* if it has no children. Nodes at level ℓ are single voxels and cannot have children. The *height* of the oct-tree is the maximum level of any of its nodes, which is at most ℓ. Given an oct-tree, B, we use its external nodes to construct a function $f_B : \mathbb{R}^3 \to \mathbb{R}$. Specifically, we set $f_B(x)$ equal to a consensus value of the voxels that make up the external node that contains x, assuming again some tie-breaking mechanism for points that belong to two or more external nodes. We set $f_B(x) = 0$ if x lies outside all external nodes. We have $f_B = f_{\text{vox}}$ if all external nodes are voxels.

Complexes. Here we explain the terminology relating to simplicial complexes that we use in later sections. By an *i-simplex*, we mean the convex hull of $i+1$ affinely independent points. For $i = 0, 1, 2, 3$, an i-simplex is a *vertex*, an *edge*, a *triangle*, a *tetrahedron*. Given a simplex σ, the set of simplices spanned by the subsets of the vertices of σ are called *faces* of σ. A *simplicial complex* K in $\mathbb{R}^3$ is a set of simplices drawn in $\mathbb{R}^3$ in such a way that any two simplices $\sigma, \tau \in K$ intersect, if at all, along an entire mutual face. For example, one cannot have an edge piercing a triangle. The *underlying space* $|K|$ of a simplicial complex K is the union of all the simplices in K together with the topology inherited from the ambient Euclidean space.

There is also a notion of an *abstract simplicial complex*. This is a set $\mathscr{K}$ of subsets of some vertex set that is closed under containment; in other words, $\sigma \in \mathscr{K}$ and $\tau \subseteq \sigma$ requires $\tau \in \mathscr{K}$. For example, an abstract simplicial complex can be obtained by taking the *nerve* [10, page 59] K of a collection of convex sets. Letting K contain one vertex for each convex set, K will contain a simplex σ for each collection of convex sets with a non-empty intersection. Given an abstract simplicial complex $\mathscr{K}$, a *geometric realization* is a drawing of $\mathscr{K}$ as a simplicial complex K in some Euclidean space. Finally, a *triangulation* of a topological space $\mathbb{X}$ is a simplicial complex K along with a homeomorphism between $|K|$ and $\mathbb{X}$. In the rest of this section, we will often blur the difference between a topological space and a simplicial complex by imagining that we have in mind a triangulation of the former.

Homology. Here we give an informal description of the homology basics needed for this paper; for a more thorough and general treatment of homology theory, see for example [14]. We first describe the $\mathbb{Z}/2\mathbb{Z}$*-homology groups* $\mathsf{H}_i(\mathbb{X})$ of a topological space $\mathbb{X}$, restricting ourselves to dimensions $i = 0, 1, 2$. Each such group will be a vector space over $\mathbb{Z}/2\mathbb{Z}$. By a 0*-cycle* in $\mathbb{X}$ we mean a point, by a 1*-cycle* we mean a closed curve possibly with self-intersections, and by a 2*-cycle* we mean a closed surface. We can add any two i-cycles α and β by defining $\alpha + \beta$ to be the i-cycle formed by the closure of their symmetric difference. An i-cycle α *bounds* if α is the boundary of an $(i+1)$-dimensional subspace of $\mathbb{X}$; intuitively a cycle bounds if it can be pulled taut to a point entirely within the space $\mathbb{X}$. We also say that α is *homologous* to another i-cycle β if there exists an $(i+1)$-dimensional subspace whose boundary is $\alpha + \beta$. An i-dimensional *homology class* is a collection of mutually homologous i-cycles. For example, a 0-dimensional class would be a connected component, while we imagine a 1-dimensional class as a loop going around a tunnel, and a 2-dimensional class as a surface enclosing a void. The set of i-dimensional classes forms the vector space $\mathsf{H}_i(\mathbb{X})$ with addition again corresponding to taking symmetric difference. The rank of this vector space, denoted as $\text{rank}(\mathsf{H}_i(\mathbb{X}))$, is often referred to as the *i-th Betti number* of $\mathbb{X}$. It will often be convenient to talk about homology whole-sale, considering all dimensions at once. To do this formally, we define $\mathsf{H}(\mathbb{X})$ as the direct sum of the $\mathsf{H}_i(\mathbb{X})$. We extend the above notion of absolute homology assuming a second topological space, $\mathbb{X}_0 \subseteq \mathbb{X}$. An i-dimensional subspace $\alpha \subseteq \mathbb{X}$ is called a *relative i-cycle* of the pair $(\mathbb{X}, \mathbb{X}_0)$ if the boundary of α is entirely contained within $\mathbb{X}_0$; of course this includes the possibility that α has no boundary. Substituting relative for absolute cycles, we define when α bounds and when it is homologous to another relative cycle, as before. An i-dimensional *relative homology class* is a collection of homologous relative i-cycles. Finally, these classes form a vector space, as before, which we denote as $\mathsf{H}_i(\mathbb{X}, \mathbb{X}_0)$, and call the *i-th relative homology group* of the pair $(\mathbb{X}, \mathbb{X}_0)$. Similar to before, we write $\mathsf{H}(\mathbb{X}, \mathbb{X}_0)$ for the direct sum of the $\mathsf{H}_i(\mathbb{X}, \mathbb{X}_0)$.

We illustrate these definitions using the torus $\mathbb{X}$ that bounds the donut-shaped space in the lower-right quadrant of Figure 2. Since $\mathbb{X}$ is connected, we have $\text{rank}(\mathsf{H}_0(\mathbb{X})) = 1$. There is one void formed by the torus itself, and hence $\text{rank}(\mathsf{H}_2(\mathbb{X})) = 1$ as well. The group $\mathsf{H}_1(\mathbb{X})$ is more interesting. Let α and β be two meridian 1-cycles, both passing through the visible tunnel of the torus, and let γ be a longitudinal 1-cycle, going around that tunnel. Note that α and β are homologous since their sum forms the boundary of the tube between them. On the other hand, γ is homologous to neither α nor β. With a little work, we can see that any other non-bounding 1-cycle on $\mathbb{X}$ is homologous to α or to γ or to their sum. In other words, the homology classes represented by α and γ form a basis for $\mathsf{H}_1(\mathbb{X})$, and hence $\text{rank}(\mathsf{H}_1(\mathbb{X})) = 2$. Now suppose we use a plane to cut the torus into two cylinders, letting one be $\mathbb{X}_0$. The pair $(\mathbb{X}, \mathbb{X}_0)$ defines relative cycles of dimension 0, 1, and 2. We have $\text{rank}(\mathsf{H}_0(\mathbb{X}, \mathbb{X}_0)) = 0$ because every 0-cycle bounds, that is, every point can be connected to $\mathbb{X}_0$ by a curve whose boundary is that point plus some point in $\mathbb{X}_0$. Furthermore, $\text{rank}(\mathsf{H}_1(\mathbb{X}, \mathbb{X}_0)) = \text{rank}(\mathsf{H}_2(\mathbb{X}, \mathbb{X}_0)) = 1$, with the two vector spaces spanned by the longitudinal 1-cycle and the torus surface.

We have described homology here in the context of a topological space. There is also a nearly equivalent notion of homology for a simplicial complex. Using this latter idea, the vector spaces can be computed by reducing a matrix whose columns are indexed by the simplices in the complex; of course, the more simplices in the complex, the longer the running time. Hence, the oct-tree construction described in this paper will produce a faster running time via a controlled decrease in the number of input simplices. For details on simplicial homology and the matrix reduction algorithm, see [10].

Persistence. One disadvantage of homology, as described above, is its sensitivity to small changes in the space. For example, consider the twice-broken circle $\mathbb{Y}$ shown in the upper-left quadrant of Figure 2. As it is currently drawn, there are two components and no other cycles. On the other hand, a small change in $\mathbb{Y}$, perhaps caused by some inaccuracy in rendering, might cause radical homological changes; for example, the space in the lower-right quadrant of the same figure has only one component but also has a non-bounding 1-cycle. We now give a brief sketch of persistent homology, which deals with this problem by turning homology into a multi-scale concept; as before, we refer the reader to [10] for more precision and detail.

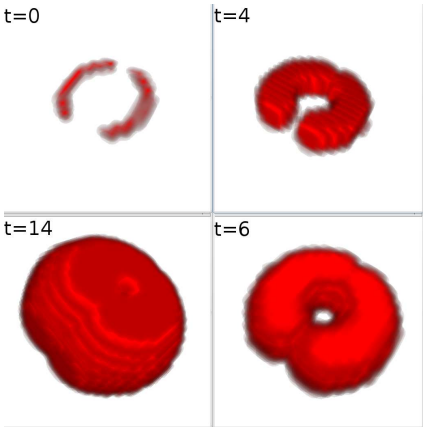

Fig. 2: Four sublevel sets of the function $d_{\mathbb{Y}}$. As we move clockwise from the upper-left, the threshold parameter increases.

We start with a topological space $\mathbb{M}$ and a continuous real-valued function $f : \mathbb{M} \to \mathbb{R}$. The main idea is to construct a sequence of vec-

tor spaces with maps between them, calling this the *filtration* of $\mathbb{M}$ induced by the function f. As a working example, we consider $\mathbb{M} = \mathbb{S}^3$, the 3-sphere, which we picture as $\mathbb{R}^3$ compactified by adding a point at infinity, and we let $f = d_{\mathbb{Y}}$, the function that maps each point in $\mathbb{S}^3$ to its distance from the space $\mathbb{Y} \subseteq \mathbb{M}$. In a first phase, we consider *sublevel sets* $\mathbb{M}_r = f^{-1}(-\infty, r]$, where $r \in \mathbb{R}$. Whenever $r \leq s$, there is an inclusion of sets $\mathbb{M}_r \subseteq \mathbb{M}_s$. In our example, the sublevel sets are empty until we reach $r = 0$, when we have $\mathbb{Y}$ itself. As r increases further, the sublevel set gradually thickens $\mathbb{Y}$ until finally the entire 3-sphere is filled in. During this process, some homology classes disappear and some new ones are formed. For example, in the passage from the upper-left quadrant of Figure 2 to the upper-right, two components are merged into one, and so we say that one component *dies* while another component remains alive. Moving to the sublevel set in the lower-right, a new 1-cycle is *born*; this 1-cycle subsequently dies moving into the lower-left, as there is now a surface for it to bound. At this moment, we have one 0-cycle alive and nothing else. When we reach the point at infinity, the 3-cycle defined by the sphere itself is formed.

During a second phase of the filtration, we consider pairs $(\mathbb{M}, \mathbb{M}^r)$, where $\mathbb{M}^r = f^{-1}[r, \infty)$ is a *superlevel set* of f, and we let r decrease from positive infinity to negative infinity. We may visualize this by moving in a counterclockwise order in Figure 2, interpreting each shape as the piece of the 3-sphere that remains if we remove the superlevel set. The first change in this second phase is the death of the remaining 0-dimensional class. Moving from the lower-left to the lower-right quadrant, we see the birth of a 2-dimensional relative class, which dies when we move into the upper-right quadrant. Going back to the upper-left quadrant, we see the birth of a 3-dimensional relative class, which dies together with the other such class when the two components are subsumed by $\mathbb{X}_0$. As a general rule, all births and deaths happen at critical values of f. The *persistence* of a class is defined to be the absolute difference between its birth and death values.

Diagrams. Given a function f, the persistence diagram $\mathrm{Dgm}(f)$, consisting of two triangles and one diamond, records the birth and death of homology classes during the filtration defined by f. We explain this for the example $f = d_{\mathbb{Y}}$, whose diagram is drawn in Figure 3.

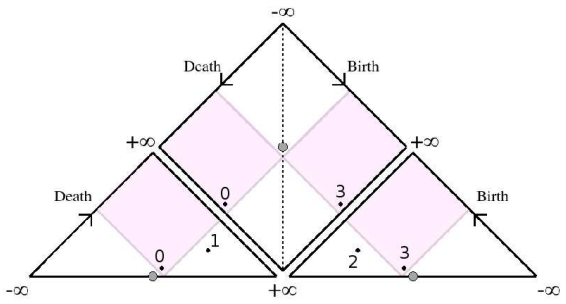

Fig. 3: The persistence diagram for the function $d_{\mathbb{Y}}$ illustrated through four of its sublevel sets in Figure 2. Each axis goes from minus to plus infinity, and we mark the origins with grey dots for improved readability. The two shaded rectangles correspond to the level set of $d_{\mathbb{Y}}$ at $a = 0.6$, and the dots in these rectangles give the homology.

We focus first on the left triangle, as it is the easiest to understand. There is a birth axis, going downhill on the right, and a death axis, going uphill on the left. For each class that is born and dies during the first phase of the filtration, we record its birth and death values and plot them as a dot in this triangle. Note that the diagram in Figure 3 contains two dots in this region, one labeled dimension 0, the other dimension 1: these correspond to the component that was there from the start, at time 0, but was merged at some early time, and a 1-cycle that formed at a later time and died at a still later time. In general, the closer a dot is to the birth=death diagonal (rendered as the horizontal baseline), the more likely it is to be interpreted as noise, although this statement cannot of course be given a precise meaning. Next we consider the diamond, in which we notice two dots, one corresponding to the component that was born at the beginning and dies right after the end of the first phase and the other to the 3-cycle that was born right before the end of the first phase. In general, the diamond contains one dot for each class that is born during the first phase and dies during the second phase of the filtration. Finally, the right triangle contains a dot for each class born and dying during the second phase. Notice that this diagram is symmetric, with points of dimension i being reflected into points of dimension $3 - i$ across the vertical line which separates the two halves of the diamond. This symmetry will be present whenever $\mathbb{M}$ is a compact manifold [10, Chapter VII].

Stability. These diagrams are useful because they are *stable* under small changes in the function, in a manner which we now describe, first via example and then the general theory. Suppose that we slightly perturb the values of our function $d_{\mathbb{Y}}$ to produce a new function g; the diagram of g is drawn in Figure 4. Notice that the diagrams of

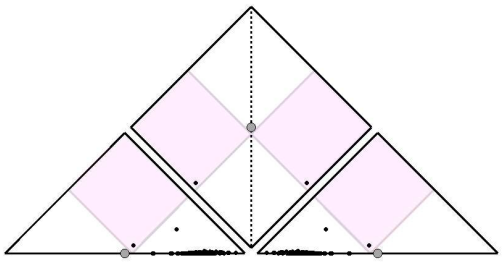

Fig. 4: The persistence diagram for a perturbed version of the function $d_{\mathbb{Y}}$.

$d_{\mathbb{Y}}$ and of g are virtually identical, except for the presence of a large number of dots along the birth=death diagonal in the latter; these extra dots are all of very low persistence and hence are probably noise. This example illustrates a general fact and motivates us to add infinitely many copies of the dots on the diagonal to the diagram. Given two real-valued functions f, g on the topological space $\mathbb{M}$, we define the L_∞-distance between them to be $\|f - g\|_\infty = \sup_{x \in \mathbb{M}} |f(x) - g(x)|$. We also define a similar distance between dots in the plane. If $u = (a, b)$ and $v = (c, d)$, then $\|u - v\|_\infty = \max\{|c - a|, |d - b|\}$. To compare the diagrams $\mathrm{Dgm}(f)$ and $\mathrm{Dgm}(g)$, we consider a bijection $\Gamma : \mathrm{Dgm}(f) \to \mathrm{Dgm}(g)$, and find the largest distance between matched dots; we then minimize this distance over all possible bijections. In symbols, we define the *bottleneck distance* between the diagrams to be:

$$W_\infty(\mathrm{Dgm}(f), \mathrm{Dgm}(g)) \quad = \quad \inf_{\Gamma} \sup_{u} \|u - \Gamma(u)\|_\infty,$$

where Γ ranges over all bijections and u ranges over all dots in $\mathrm{Dgm}(f)$. As stated in [10, Chapter VIII], the diagram is stable under measuring difference with the bottleneck distance:

$$W_\infty(\mathrm{Dgm}(f), \mathrm{Dgm}(g)) \quad \leq \quad \|f - g\|_\infty. \tag{1}$$

Robustness. The persistence diagram $\mathrm{Dgm}(f)$ contains a great deal of information, namely the homology of *every* level set $f^{-1}(a)$ and *every* interlevel set $f^{-1}[a, b]$, as well as the robustness of this homology. We describe this here in some detail, while referring the reader to [3, 11] for precise definitions and statements. First, it can be shown that for any regular value a of f, the homology of the level set $f^{-1}(a)$ corresponds to the dots in $\mathrm{Dgm}(f)$ that lie within two particular rectangles. These rectangles have one shared corner located at the dot a on the vertical line in the middle diamond, and their sides are parallel to the two coordinate axes; the dimensions of the dots in the right rectangle also must be reduced by 1. For example, the dots in the two shaded rectangles in Figure 3 give the homology of $d_{\mathbb{Y}}^{-1}(0.6)$. This level set looks like the boundary of two slightly thickened circular arcs; hence its homology consists of two components and two 2-cycles. Sliding a along the vertical line produces different rectangle pairs, each of which gives the homology of a different level set.

We now describe what is meant by the robustness of the homology of a level set $f^{-1}(a)$. Given an $r \geq 0$, we consider the interlevel set $\mathbb{M}_r(f,a) = f^{-1}[a-r, a+r]$, and we define an *r-perturbation* h of f to be a real-valued function h such that $\|h-f\|_\infty \leq r$. Note then that the level set $h^{-1}(a)$ is a subset of $\mathbb{M}_r(f,a)$, and so there is an induced map on homology groups $\mathsf{j}_h : \mathsf{H}(h^{-1}(a)) \to \mathsf{H}(\mathbb{M}_r(f,a))$. The image of this map $\mathrm{im}\,\mathsf{j}_h$ can be understood intuitively as the set of homology classes in $\mathsf{H}(\mathbb{M}_r(f,a))$ that are homologous, within $\mathbb{M}_r(f,a)$, to some homology class in $\mathsf{H}(h^{-1}(a))$. We then define the *well group* at radius r to be the intersection of the images of the maps induced by all r-perturbations of f; that is,

$$\mathsf{U}_r(f,a) \quad = \quad \bigcap_{\|h-f\|_\infty \leq r} \mathrm{im}\,\mathsf{j}_h.$$

It can be shown [11] that these well groups can only drop in rank as the radius increases. A radius r at which such a drop occurs will be called a *terminal critical value*. We plot these terminal critical values as dots on the non-negative real line. The resulting multi-set of dots displays the robustness of the level set, and we call it the *well diagram* $\mathrm{Ugm}(f,a)$,

These well diagrams can be read directly from $\mathrm{Dgm}(f)$ in the following manner. We draw the rectangle pair corresponding to a. For each dot u within this pair, there will be a dot $\rho_f(u) \in \mathrm{Ugm}(f,a)$, where $\rho_f(u)$ is the distance between this dot and the boundary of the rectangle. In the example shown in Figure 3, with $a = 0.6$, all four dots appear to have fairly small ρ-values. This is not surprising. If we consider, for example, the function $h = d_{\mathbb{X}} + 0.7$, then the level set $h^{-1}(0.6) = d_{\mathbb{X}}^{-1}(-0.1) = \emptyset$, and thus h cannot support any homology classes whatsoever; hence the robustness of every class in the level set is certainly less then 0.7.

3 The Algorithm

Given a real-valued function $f : \mathbb{R}^3 \to \mathbb{R}$, the function f_{vox} is a piecewise constant approximation of f. By constructing an oct-tree B, we may further simplify f_{vox}, approximating it with a new function f_B defined on the external nodes of the oct-tree, and then compute the persistence diagram of this new function. In this section, we describe an algorithm which accomplishes this. In brief, the main steps of the algorithm are as follows:

Step 1. Beginning with the one-node oct-tree, we subdivide external nodes to produce an oct-tree B.

Step 2. We construct the nerve of a slightly perturbed version of the cube complex consisting of the external nodes of B.

Step 3. We filter this nerve by lower stars (defined below) and compute the associated persistence diagram.

We now describe each of these steps in turn.

3.1 Oct-Tree

We assume a real-valued *priority function* φ that assigns a priority $\varphi(\mu) \geq 0$ to any oct-tree node μ. Intuitively, $\varphi(\mu)$ tells us how urgently μ must be subdivided. We require that φ is zero if the oct-tree node only contains one voxel, and that φ decreases in value along paths of the oct-tree. The *fitness* of an oct-tree B is the maximal priority of its external nodes: $\Phi(B) = \max_\mu \varphi(\mu)$. Different priority functions can be utilized to accomplish different tasks. In Section 4.2, we will in particular analyze the properties of the following two:

- $\varphi(\mu) = M(\mu) - m(\mu)$, where $M(\mu) = \max_{y \in \mu} f_{\mathrm{vox}}(y)$ and $m(\mu) = \min_{z \in \mu} f_{\mathrm{vox}}(z)$. In words, the priority of a node is the range of the function values inside the corresponding cube.

- $\varphi(\mu) = \frac{M(\mu)-m(\mu)}{m_0(\mu)}$, where M and m are as before, and $m_0(\mu) = \min_{y \in \mu} |f_{\mathrm{vox}}(y)|$. This assigns a higher priority to nodes having function values closer to zero.

We construct the oct-tree in a top-down manner, starting with the one-node oct-tree B_0 that defines a constant function. We store the external nodes in a priority queue in decreasing order of φ. In each iteration, we remove the external node with highest priority from the queue and subdivide it into its eight children. These children become external nodes and are inserted into the priority queue. We continue this process until we satisfy a certain termination condition. For that, we have again two choices. We can fix an integer N and stop when the next iteration would produce an oct-tree with more than N external nodes. With this condition, we obtain an oct-tree with best fitness that can be achieved using N external nodes. Alternatively, we can fix a real value θ and stop if the fitness of the oct-tree becomes less than or equal to θ. In this way, we obtain a smallest oct-tree whose fitness is not larger than θ. Instead of fixing a single integer N, we can also fix a sequence of integers $N_1 < N_2 < \ldots < N_k$ and produce oct-trees B_i with best fitnesses, having at most N_i external nodes. The corresponding functions f_i give progressively more accurate approximations of f_{vox}. Of course, the same can be done for a sequence of values $\theta_1 > \theta_2 > \ldots > \theta_k$.

3.2 Dual Complex

We now describe a procedure which takes an oct-tree B and produces from it a simplicial complex, which we call its *dual complex*. In Section 4 we will prove that, under an added balancing assumption on the tree, this dual complex is geometrically realized in $\mathbb{R}^3$.

Perturbation. A direct approach to constructing the dual complex runs into difficulties because a corner can be shared by as many as eight cubes. To cope with this difficulty, we think of each voxel as a *Voronoi cell*: the set of points in $\mathbb{R}^3$ for which the voxel center is a nearest integer point. We perturb the points slightly such that these Voronoi cells are simple polyhedra that meet in three along a shared edge and in four around a shared corner. A particularly straightforward such perturbation moves the integer points along the direction of the main diagonal: $(i,j,k) \mapsto (i-\varepsilon m, j-\varepsilon m, k-\varepsilon m)$, where $\varepsilon > 0$ is sufficiently small and $m = i+j+k$. The perturbation has the same effect

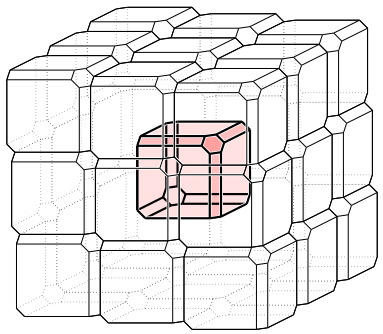

Fig. 5: A configuration of $27 = 3^3$ truncated cubes. Only 14 of the 26 surrounding truncated cubes touch the middle cube.

on every voxel, changing it to a particular type of truncated cube. As shown in Figure 5, the truncation flattens six edges to thin rectangular faces and it flattens two diagonally opposite vertices to small hexagonal faces. We call this truncated cube a *perturbed voxel*. Recall that the *level* of a node in the oct-tree is its distance from the root node. If that node has level λ, then it represents a cube of side length $n/2^\lambda = 2^{\ell-\lambda}$, containing $8^{\ell-\lambda}$ voxels. After the perturbation, the node represents an equally large configuration of perturbed voxels. We refer to this union as a *perturbed cube*, noting that it is not convex unless $\lambda = \ell$. However, we can control the amount of non-convexity by choosing ε sufficiently small.

Voxel complex. Before considering the general case, assume B is the complete oct-tree, that is, its external nodes are the n^3 given voxels. We form the *dual complex*, denoted as $K = K(B)$, by calling each voxel a *vertex*. We connect two, three, and four vertices by an *edge*, a *triangle*, and a *tetrahedron* if the corresponding perturbed voxels have

a non-empty common intersection. More concisely, K is isomorphic to the *nerve* of the collection of perturbed voxels. Geometrically, we draw each vertex as the point at the center of the voxel. Edges, triangles, and tetrahedra are drawn as convex hulls of the two, three, and four centers.

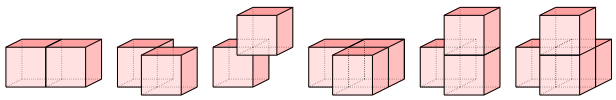

Fig. 6: Configurations of two, three, and four voxels defining edges, triangles, and tetrahedra in the dual complex.

Ignoring boundary effects, the dual complex looks the same way at every vertex. There are three types of edges: aligned to a coordinate direction, to the diagonal in a coordinate plane, and to the space diagonal. Similarly, there are two types of triangles and one type of tetrahedron; see Figure 6. We can assemble six tetrahedra around the shared space diagonal to triangulate a cube. This cube is of the same size as a voxel, but dual as its corners are the centers of eight voxels. We get the entire dual complex by assembling $(n-1)^3$ such configurations. A single interior vertex belongs to eight such triangulated cubes. For two diagonally opposite cubes, all six tetrahedra share this vertex, while for each of the other six cubes, only two tetrahedra do. The resulting 24 tetrahedra form the *star* of the vertex, and the boundary of the star is its *link*, as shown in Figure 7. Note that the star of a vertex consists of 14 edges, 36 triangles, and 24 tetrahedra. Multiplication with the number of voxels and compensation for double-counting gives upper bounds on the number of simplices in the dual complex of the voxel array: #vertices $= n^3$, #edges $\leq 7n^3$, #triangles $\leq 12n^3$, #tetrahedra $\leq 6n^3$. These bounds are tight up to lower-order terms.

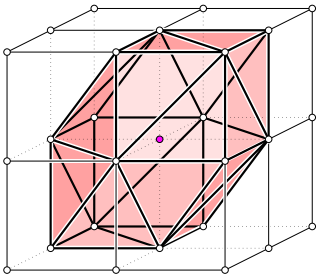

Fig. 7: The link of the shaded vertex in the middle, which corresponds to the middle cube in Figure 5. Its vertices correspond to the 14 truncated voxels that touch the middle truncated voxel.

Oct-tree complex. We generalize the construction of the dual complex to oct-trees that are not necessarily complete. Letting B be such an oct-tree, the vertices of $K = K(B)$ are the external nodes of B and K is isomorphic to the nerve of the collection of corresponding perturbed cubes. The external nodes of B do not all have the same level so that cubes of different sizes can touch each other. As a consequence, K is not quite as regular as in the voxel array case. For example, K is no longer a degenerate Delaunay triangulation of the cube centers. Because we use the perturbed cubes to decide upon the simplices in the dual complex, we have $0 \leq \dim \bigcap S \leq 3-i$ for every collection S of $i+1$ external nodes whose corresponding perturbed cubes have a non-empty common intersection. This is because the non-empty common intersection of a collection of perturbed cubes may shrink, as we let ε go to zero, and in the limit drop in dimension, but it cannot disappear. This implies that the dual complex of an oct-tree is a 3-dimensional abstract simplicial complex. We note that the common intersection of a collection of perturbed cubes does not change its homotopy type as we decrease ε. In the limit, the intersection is convex. This implies that the intersection is contractible, also for small $\varepsilon > 0$. Hence the Nerve Theorem applies, showing that the homotopy type of the dual complex is the same as that of the union of perturbed or unperturbed cubes. We will use this fact in Section 4. Geometrically, we draw each vertex at the center of the corresponding cube, and we draw each edge, triangle, and tetrahedron as the convex hull of its vertices. However, as we will see in Section 4.1, this recipe does not necessarily give a geometric realization in $\mathbb{R}^3$. Recall that the dual complex of the array of $N = n^3$ voxels has at most $26N$ simplices. We now show the same bound for oct-trees.

1 (Dual Complex Bound) *The dual complex of an oct-tree with N external nodes has at most $26N$ simplices.*

PROOF. It suffices to prove that the average number of edges in a vertex star is at most 14. Since the link of the vertex is a triangulated 2-sphere, this gives an average of at most 36 triangles and 24 tetrahedra in the star. Summing over all vertices and compensating for double-counting, we get #vertices $= N$, #edges $\leq 7N$, #triangles $\leq 12N$, #tetrahedra $\leq 6N$, and therefore at most $26N$ simplices altogether.

To count the edges, we replace each external node by the corresponding perturbed cube, μ. We call another perturbed cube ν a *neighbor* of μ if they intersect. Surrounding μ by perturbed cubes of the same size, we get 14 intersections with neighbors, calling each intersection a *side* of μ; Figure 5 shows the sides of a perturbed voxel.

We look at the neighbors of μ that correspond to external nodes of the oct-tree. If all such neighbors have the same size as μ, we have an edge for each side and therefore exactly 14 edges, as desired. If there are neighbors larger than μ then we have possibly fewer edges and nothing to worry about. The difficult case is when there are neighbors smaller than μ. If there is one such neighbor across a side ϕ then all neighbors across ϕ are smaller than μ. Furthermore, except for one, each such smaller neighbor ν has at least two of its sides in ϕ. We can therefore charge the corresponding edge in the dual complex to the additional side of ν. Repeating this reassignment of edges for each cube, we eventually charge at most one edge to each side. The upper bound on the average number of edges follows. ∎

3.3 Persistence

As described in Section 2, we compute the persistence diagram of a function $f : \mathbb{M} \to \mathbb{R}$ by pairing the births and deaths in the corresponding filtration. In the application at hand, the function is defined on $\mathbb{R}^3$, which we compactify by adding a point at infinity, so $\mathbb{M}$ is the 3-sphere. We now describe how we use the dual complex to achieve this goal.

Extraction. We read the simplices off the oct-tree without explicit perturbation. As mentioned earlier, if a collection of perturbed cubes has a non-empty common intersection then so the corresponding unperturbed cubes. Furthermore, this common intersection includes a point that is a corner of at least one of these cubes. It follows that iterating through all corners of external nodes in the oct-tree B recovers all simplices in the dual complex, K. Now suppose we are at a corner point, 0, and we identified a set S of unperturbed cubes that have 0 in their boundaries. For every subset of S, we need to test whether or not the corresponding perturbed cubes have a non-empty intersection. Since the perturbation has the same effect on every corner, this reduces to checking whether the cubes are arranged in a certain configuration around the corner, such as the ones displayed in Figure 6.

Ordering. For the computation of persistence, we need more, namely the simplices in K in a particular ordering. We restrict the discussion to the first phase of the filtration, in which we need the simplices ordered in increasing function value. The second phase is symmetric, with the simplices ordered in decreasing function value.

We are now more precise about this ordering. For each vertex $\mu \in K$, let $f_B(\mu)$ be the value of the external node. For each simplex $\sigma \in K$, we define $f_B(\sigma)$ equal to the maximum value of any of the vertices of σ. Finally, we sort the simplices in the order of increasing value, breaking ties by ordering lower- before higher-dimensional simplices. This rule implies that the faces of every simplex precede the simplex in the ordering. Hence, every prefix of the ordering is a subcomplex of K. The sequence of these subcomplexes forms the *lower star filtration*

of K. As proved in [10], it gives the same persistence diagram as the continuous sequence of sublevel sets of $\mathbb{M}$, which we assume is triangulated by K. The most expensive operation here is the sorting of the simplices. We can therefore improve the performance of the algorithm by sorting the smaller collection of external nodes. This is equivalent to sorting the vertices of K. We then collect the remaining simplices by iterating through the external nodes in order. For each node μ, we find the corners in its boundary, and for each corner, we get the corresponding simplices as described. However, at this time, we have use only for the simplices whose highest vertex is μ. These are exactly the simplices in the *lower star* of μ. Concatenating the lower stars gives the sequence of simplices for the first phase. Adding the symmetric sequence of upper stars and feeding both to the persistence algorithm, we get the persistence diagram of f.

4 Analysis

In this section, we describe and analyze two different choices for the priority function φ. The first choice is tailored to produce a persistence diagram that is not too far off from $\mathrm{Dgm}(f_{\mathrm{vox}})$, while the second gives a good approximation of the robustness for a particular level set. To overcome a technical obstacle to this theory, we use the dual complex to get a piecewise linear, continuous function. It defines the level set, and because critical values remain unchanged, it produces the same persistence diagram to which we can now formally apply the stability result (1) mentioned in Section 2.

4.1 Piecewise Linear Approximation

The main result in this subsection is that the dual complex of an oct-tree that satisfies an additional balancing condition is geometrically realized in $\mathbb{R}^3$. We can therefore use it to define a piecewise linear approximation of the original function.

Balanced oct-trees. We motivate the balancing condition by an example in which the above recipe for drawing the dual complex in $\mathbb{R}^3$ does not give a geometric realization. Take two nodes at level λ and a third node at level $\lambda+2$. Arrange them so that the two big cubes share a line segment (an edge common to both), and the little cube is nested between them, sharing a face with each and sharing a quarter at one end of the line segment. The dual edge connecting the centers of the two big cubes passes through the middle of the line segment and is therefore contained in the union of the two cubes. In contrast, the dual triangle is not contained in the union of the three cubes. It is now easy to find two additional cubes sharing a face so that their dual edge crosses the triangle.

This counterexample to geometric realization uses the existence of two neighboring cubes whose levels differ by at least two. Following [4], we consider oct-trees that have no such cubes. Specifically, we call two external nodes in an oct-tree *neighbors* if the corresponding perturbed cubes have a non-empty intersection. The oct-tree is *balanced* if every neighbor of an external node at level λ is at level $\lambda-1$, λ, or $\lambda+1$. Given an unbalanced oct-tree, it is not difficult to find the smallest balanced oct-tree that refines it. We now show that balanced oct-trees have significant advantages over unbalanced ones.

Non-degeneracy. Recall that we draw the simplices in the dual complex as convex hulls of the centers of cubes that are external nodes in the oct-tree. We can show that these simplices are non-degenerate provided the oct-tree is balanced.

2 (Non-degeneracy Lemma) *Let S be a collection of $i+1$ external nodes in a balanced oct-tree such that the corresponding perturbed cubes have a non-empty common intersection. Then the centers of the $i+1$ (unperturbed) cubes in S span a non-degenerate i-simplex in $\mathbb{R}^3$.*

PROOF. All triangles and edges are faces of tetrahedra in the dual complex. It thus suffices to prove the claim for $i=3$, when S contains four cubes whose corresponding perturbed cubes share a single point, which we denote as 0. The four cubes contain four voxels, one in each cube, whose corresponding perturbed voxels share 0. They form the last of the six configurations in Figure 6. We consider the planes EW, NS, and FB, as drawn in Figure 1. We say such a plane *separates* the four cubes if it does not intersect their interiors and there is at least one cube on each side of the plane. Note that due to structural limitations of balanced oct-trees, at least one of the planes EW and NS is separating. We now distinguish between three cases, proving each time that the centers of the four cubes span a non-degenerate tetrahedron.
Case 1. All three planes separate the four cubes. The centers of the cubes then lie on open half-lines that emanate from 0 in space diagonal directions, that is, all their points are equidistant from the three planes. Since two voxels in the configuration are diagonally opposite from each other, two of these half-lines belong to the same line. Any plane that intersects both must contain the line. But then, this plane can intersect only one of the other two half-lines.
Case 2. Two planes separate the four cubes, say EW and FB. Since NS does not separate, at least one of the two cubes in the back occupies both available octants. By assumption of balance, its size (measured as the length of its sides) is twice the size of the two stacked cubes in the front, which are of equal size. Their three centers thus lie on a vertical plane. The center of the remaining cube does not lie in this plane, no matter whether it has the size of the front cubes or it is twice that size. The case in which NS and FB separate is symmetric. The case in which EW and NS separate is different but similar enough to permit the same argument.
Case 3. Only one plane separates the four cubes, say EW. Then the cube on the left occupies all four octants on the left. By assumption of balance, the right cubes are all of the same size and half the size of the left cube. There are three of them and their centers lie in a plane parallel to and to the right of EW. In contrast, the center of the left cube lies to the left of EW. The case in which NS separates is symmetric. In all cases, we conclude that there is no plane that contains the centers of all four cubes in S. Equivalently, the four centers span a non-degenerate tetrahedron. □

Since the tetrahedron spanned by the centers of the four cubes in S is non-degenerate, it has a well-defined orientation. In all three cases in the proof, this orientation is the same for the tetrahedron spanned by the centers of the four voxels that share the same point, 0.

Geometric realization. It is now not difficult to show that the dual complex of a balanced oct-tree is geometrically realized in $\mathbb{R}^3$. To this end, we use the Nerve Theorem, which implies that the dual complex has the same homotopy type as the union of cubes, namely that of a point; see [10, page 59]. Because the tetrahedron defined by the cubes have consistent orientation with those defined by the voxels, we also know that any two tetrahedra that share a triangle lie on opposite sides of the plane that contains this triangle. To prevent global violations of geometric realization, such as tetrahedra that wrap around an edge or a vertex more than once, we just need to show that the boundary of K is geometrically realized in $\mathbb{R}^3$. This is easiest if we add layers of cubes around the cubes defined by the oct-tree, getting finer toward the outside until the entire outermost layer consists of voxels. Thus, the boundary of our dual complex is the same as the boundary of the dual complex of the voxel array that covers the same space. The latter is geometrically realized by construction. We summarize:

3 (Geometric Realization Theorem) *The dual complex of a balanced oct-tree is geometrically realized in $\mathbb{R}^3$.*

4.2 Thresholds and Error Bounds

In the following, whenever we need a continuous function, such as in the definition of a level set, we implicitly substitute the piecewise linear function defined by the dual complex for the piecewise constant function defined by the oct-tree.

Uniform thresholding. We first imagine a situation in which f_{vox} varies only a little in certain regions, while varying quite sharply in others. A typical example is a grey-scale image, which usually has several almost constant regions and a few regions of drastic change. In such a situation, we wish to simplify f_{vox} in the more uniform regions while maintaining detail in the areas of change. To accomplish this, we define our priority function as $\varphi(\mu)=M(\mu)-m(\mu)$ and we stop

subdividing when we have reached an oct-tree B whose fitness is better than or equal to a chosen threshold. Then we define f_B on this oct-tree via $f_B(x) = \frac{1}{2}(M(\mu) + m(\mu))$, where μ is the external node that contains x; if x is not in any such node, we set $f_B(x) = 0$. The new function f_B is a simplified version of f_{vox} since we only need to know its values on the external nodes. On the other hand, the values of the two functions do not disagree by much. Hence their respective persistence diagrams do not differ by much. More precisely, we show:

4 (Diagram Inference Theorem) *The bottleneck distance between the diagrams is bounded:* $W_\infty(\text{Dgm}(f_B), \text{Dgm}(f_{\text{vox}})) \leq \frac{\Phi(B)}{2}$.

PROOF. According to (1), we need only show that $\|f_B - f_{\text{vox}}\|_\infty \leq \Phi(B)/2$. To this end, we fix $x \in \mathbb{R}^3$ and prove $-\Phi(B)/2 \leq f_B(x) - f_{\text{vox}}(x) \leq \Phi(B)/2$. If x is not contained in any external node, then $f_B(x) = 0 = f_{\text{vox}}(x)$ and the claim is trivial. So suppose that μ is the external node containing x, and put $m = m(\mu), M = M(\mu)$, and $\varphi = \varphi(\mu)$. Since $f_B(x) = \frac{1}{2}(M+m)$ and $m \leq f_{\text{vox}}(x) \leq M$, we have

$$\frac{1}{2}(m-M) \;\leq\; f_B(x) - f_{\text{vox}}(x) \;\leq\; \frac{1}{2}(M-m).$$

We get the desired inequalities by noting that $M - m = \varphi \leq \Phi(B)$. □

The above theorem demonstrates that our algorithm trades speed for accuracy, as we now explain. Given an oct-tree B with fitness Φ, the bottleneck distance between $\text{Dgm}(f_B)$ and $\text{Dgm}(f_{\text{vox}})$ is at most $\Phi/2$. This implies that there is a bijection between subsets of the two diagrams that contain all dots of persistence larger than Φ and no dots on the baseline. In other words, all dots with persistence larger than Φ in $\text{Dgm}(f_B)$ represent genuine features of f_{vox} and all features with persistence larger than Φ in f_{vox} are represented by dots with positive persistence in $\text{Dgm}(f_B)$. Hence if we are only interested in the most persistent classes of f_{vox}, we need only subdivide until we reach an oct-tree with reasonable fitness. Such an oct-tree will not have too many nodes. Indeed, assuming our original function is Lipschitz, all external nodes of B have side length at most some constant times Φ, and hence the number of simplices in the dual complex is at most some constant times n^3/Φ^3.

Value-dependent thresholding. We have just described how to use an oct-tree to simplify a function while maintaining some confidence in the resulting persistence computation. Suppose, on the other hand, that we are interested in the robustness of a particular level set of f, say $f^{-1}(0)$. In this case, we do not wish to change the level set at all. Here, we describe a construction that maintains the level set exactly, while simplifying other regions of the cube; we also discuss the consequences for the robustness computation. For each node μ, we define $m_0(\mu) = \min_{y \in \mu} |f_{\text{vox}}(y)|$, and then set $\varphi(\mu) = (M(\mu) - m(\mu))/m_0(\mu)$. Note that $\varphi(\mu)$ can take ∞ as a value if some voxel in μ is part of the level set; we also use the convention that $\frac{0}{0} = 0$ in the priority queue. We again start with the one-node oct-tree and subdivide external nodes until we reach an oct-tree with fitness $\Phi < 2$. Because $f_B(x) = \frac{1}{2}(M+m)$ and $m \leq f_{\text{vox}}(x) \leq M$, we get $-\frac{1}{2}(M-m) \leq f_B(x) - f_{\text{vox}}(x) \leq \frac{1}{2}(M-m)$, as before. Substituting $\varphi \cdot m_0$ for $M - m$ and using $\varphi \leq \Phi$ as well as $0 \leq m_0 \leq |f_{\text{vox}}(x)|$, we get

$$-\frac{\Phi}{2}|f_{\text{vox}}(x)| \;\leq\; f_B(x) - f_{\text{vox}}(x) \;\leq\; \frac{\Phi}{2}|f_{\text{vox}}(x)|, \qquad (2)$$

for every point $x \in \mathbb{R}^3$. In words, the error is bounded from above by the absolute value of half the fitness times the function value. Since $\Phi/2 < 1$, f_B and f_{vox} share the same level set at $a = 0$: $f_B^{-1}(0) = {f_{\text{vox}}}^{-1}(0)$. In particular, the homology of these two level sets is identical, and hence it makes sense to compare the robustness of these level sets to perturbations of the two different functions. The relationship is quite simple. The following result follows from (1) after taking logarithms of the two functions in question. We omit any further details of the proof.

5 (Robustness Inference Theorem) *There is a bijection* $\Gamma : \text{Dgm}(f_B) \to \text{Dgm}(f_{\text{vox}})$ *such that, for every dot* $u \in \text{Dgm}(f_B)$ *which corresponds to a homology class* $\alpha \in \mathsf{H}(f_B^{-1}(0)) = \mathsf{H}({f_{\text{vox}}}^{-1}(0))$*, the dot* $\Gamma(u)$ *also corresponds to* α *and we have:*

$$(1 - \frac{\Phi(B)}{2})\rho_{f_B}(u) \;\leq\; \rho_{f_{\text{vox}}}(\Gamma(u)) \;\leq\; (1 + \frac{\Phi(B)}{2})\rho_{f_B}(u).$$

In other words we may use the persistence diagram of the simplified function f_B to make inferences about the robustness of the level set under perturbations of f_{vox}.

5 Computational Experiments

In this section, we discuss the application of our methods to 3-dimensional images of root systems. Our computations are part of a larger effort to characterize their shape, and to form a relationship between the phenotype and the genotype of agricultural plants. More specifically, we focus here on how the root system explores the space in the soil and how root systems of two or more plants interact.

Our software. We have written a preliminary C++ implementation (about 2500 lines of code) of the algorithm presented in Section 3 to compute an oct-tree and its dual complex. For the persistence computation itself, we used Dionysus[1] written by Dmitriy Morozov. Moreover, we have implemented an interactive interface to visualize persistence diagrams in Python (about 1000 lines of code). All diagrams in this paper are generated by this software. We remark that it contains additional features like showing the birth- and death-coordinates of a dot (by just moving the mouse on it), zooming into regions, or moving the location of the level set box (the shaded pair of rectangles in the diagrams). As mentioned earlier, we distort coordinates in order to draw the extended real line, $[-\infty, \infty]$, as a finite interval. The tool provides several scaling functions for this purpose. In this paper, we consistently used a linear scale for the interval between the minimum and the maximum function value. The software is available from the third author on request.

All experiments are performed on a general computing server clocked with 2.53 GHz, 8 MB Cache and 96 GB RAM, running under Linux. The data sets are obtained from 2-dimensional images of root systems grown and photographed in the Benfey Lab at Duke University. An array of such photographs are then converted into a 3-dimensional voxel array representation using the reconstruction software by Ying Zheng.

Distance from a root. In our first example, we study the Euclidean distance from the root. This defines a real-valued function on $\mathbb{R}^3$ whose sublevel sets arise from uniformly thickening the root. The topological analysis of this function provides a characterization of the way the root system distributes itself in the available space. We can therefore interpret the persistence diagram as a measurement of the extrinsic geometry of the root system. The left triangle of the diagram is shown in Figure 8. The two most persistent homology classes are

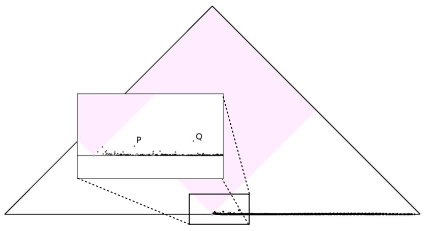

Fig. 8: Left triangle of the persistence diagram of the distance function defined by a single root system.

both of dimension 1, and the corresponding dots in the diagram are $P = (4, 6.5)$ and $Q = (11, 14.8)$. Intuitively, a 1-cycle is formed when two branches of the root meet while thickening. Looking at the appro-

[1] http://hg.mrzv.org/Dionysus/

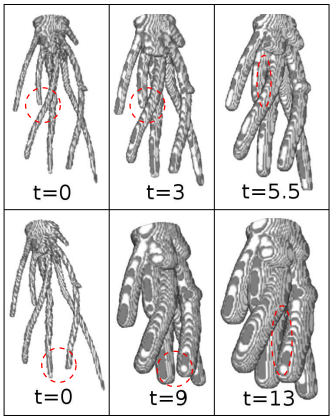

Fig. 9: Sublevel sets of the distance function of a root system. The dashed curves mark upcoming births and deaths.

priate sublevel sets, we can easily detect the corresponding 1-cycles of the root; see Figure 9. At $t = 3$, we see two thickened branches getting close to each other (dashed circle) and forming a 1-cycle at $t = 4$, the birth-coordinate of P. At $t = 5.5$, this 1-cycle is almost filled out and dies shortly after at $t = 6.5$, the death-coordinate of P. In the bottom row, we see two thickened branches getting close to each other at $t = 9$ (dashed circle) and forming a 1-cycle at $t = 11$, the birth-coordinate of Q. This 1-cycle still exists at $t = 13$ and eventually dies at $t = 14.8$, the death-coordinate of Q.

Trading speed for accurate diagram. The persistence diagram in Figure 8 takes about 30 minutes to compute; the corresponding complex has about 7.6 million vertices. We approximated the distance function using an oct-tree generated by uniform thresholding, stopping after a certain number of external nodes. Two examples, for 40,000 and 160,000 nodes, are displayed in Figure 10; these examples took 16 and 41 seconds to construct. Figure 10 also shows the exact diagram. We can see that the 40,000-node approximation is quite inaccurate but already provides evidence for the existence of two 1-cycles with high persistence. Furthermore, the 160,000-node approximation is very close to the exact persistence diagram, with a dot $P' = (3.5, 5.6)$ close to P and a dot $Q' = (9.7, 13.5)$ close to Q. The fitness of the tree is about 5.2, which would allow any dot to move up to 2.6 units of length before reaching the location in the exact diagram; see the Diagram Inference Theorem from Section 4.2. While this would allow P' to move to the baseline and disappear, we already know that Q' corresponds to a class with non-zero persistence. Comparing the approximate 1-cycles with the exact ones, we see that the actual movement is about a factor of 2 less than the bound given in the mentioned theorem.

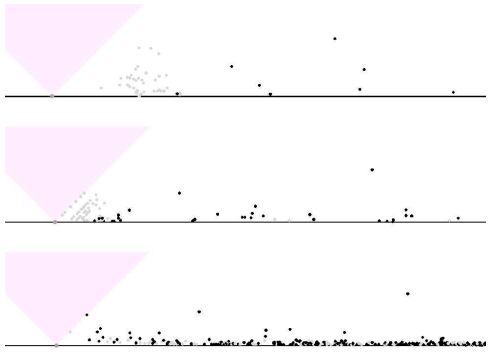

Fig. 10: Cutout of the persistence diagrams for the approximated distance function defined by trees with 40,000 and 160,000 external nodes, and for the exact distance function. The dots of 1-dimensional classes are black, and those of other homology classes are gray.

Linking between roots. As a second example, we consider the case of two root systems growing in the same container. We are interested in a computational criterion that decides whether the two roots stay at a safe distance from each other or they interact, with branches of one root entering the area of the other. We use topology to distinguish between these two cases. Writing $d_i : \mathbb{R}^3 \to \mathbb{R}$ for the Euclidean distance function defined by the i-th root, we consider their difference, $d = d_1 - d_2$ and in particular the zero set, $d^{-1}(0)$, which consists of the points with equal distance from both roots. If the roots are separated, the zero set is a topological plane with trivial homology. On the other hand, the presence of a 1-cycle in the zero set suggests an interaction between the two root systems.

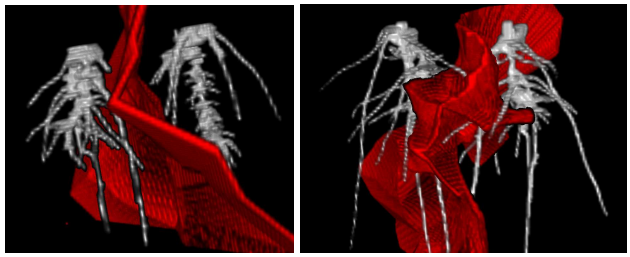

Fig. 11: Two roots along with the surface of equidistant points. Left: the surface has trivial homology, suggesting the roots do not interact. Right: the surface has genus 1, suggesting the roots interact.

Figure 11 shows two pairs of roots and the zero sets of the corresponding functions. We computed the persistence diagrams for both pairs, using an oct-tree generated with value-dependent thresholding that maintains the zero set (Figure 12). The zero set of the first pair of roots has trivial homology, while that of the second pair has a 1-dimensional class. The robustness of that 1-dimensional class is sufficiently large to be reassured that it is not an artifact of small inaccuracies in the measurements or the reconstruction.

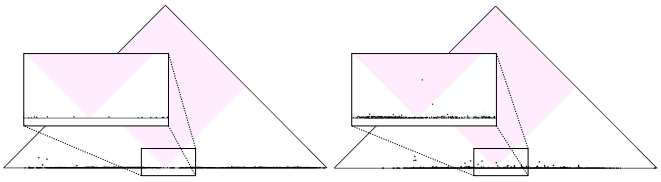

Fig. 12: Left: triangle in the persistence diagram of the difference in distance from two non-interacting root systems. Right: the same for two interacting root systems.

Trading speed for accurate robustness. Computing the exact diagrams in Figure 12 took 1.5 hours for the non-interacting, and 4 hours for the interacting roots.[2] For the interacting roots, the resulting dual complex contains about 14 million vertices. As a more practical alternative, we approximated the persistence diagrams using oct-trees. Aiming to analyze the robustness of the 1-cycle in the zero set, we used value-dependent thresholding when assigning priorities to external nodes. Figure 13 shows two approximations, using 40,000 and 640,000 external nodes, together with the exact diagram. The approximations took 23 and 166 seconds to compute. We can see that the 1-cycle in the zero set already appears for the rather coarse approximation with 40,000 nodes. For 640,000 nodes, we get a (visually) closer approximation, the fitness of the oct-tree is about 1.36, and the dot representing the 1-cycle has coordinates $(-3.88; 4.99)$. The corresponding approximation to its robustness is 3.88. Using the Robustness Inference Theorem from Section 4.2, the robustness of that 1-cycle with respect to f_{vox} is greater than 1.2. We can infer that this cycle is still present under 1-perturbations of the function d; for example, if we have made a rendering error that either expanded or retracted the boundary of one root by one voxel.

[2]The difference in timing is explained by different input resolutions.

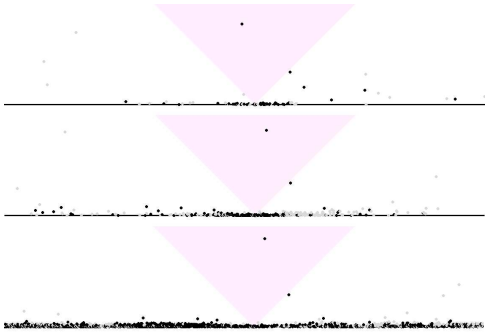

Fig. 13: Cutout of the triangle diagram for the interacting roots in Figure 11. From top to bottom, we display approximations for 40,000 and 640,000 external nodes as well as the exact diagram. The dots of 1-dimensional classes are black, and those of other homology classes are gray.

n^3	#Nodes	t_{scan}	t_{pri}	t_{nodes}	Φ
75^3	160K	0.34	0.09	0.02	0.025
	320K	0.34	0.09	0.03	0.022
	640K	0.34	0.09	0.04	0.000
150^3	160K	2.71	0.69	0.03	0.037
	320K	2.65	0.69	0.07	0.032
	640K	3.06	0.68	0.14	0.013
300^3	160K	22.30	7.89	0.05	0.043
	320K	22.17	8.01	0.13	0.038
	640K	22.23	7.91	0.28	0.020

Table 1: Results for computing the oct-tree of the distance function of a single root system. We write t_{scan} for the time to scan the input data, t_{pri} for the time to compute the priorities (and perform a few other technical operations related to B_{vox}), t_{nodes} for the time to compute the external nodes of the oct-tree, and Φ for the fitness of the resulting oct-tree. All timings are given in seconds.

Performance. We finally investigate the performance of the three main steps of the algorithm, which are: 1. compute the oct-tree with a predefined number of external nodes, 2. turn the oct-tree into a simplicial complex via the dualization, 3. compute the persistence diagram of the filtration of the simplicial complex; see Section 3.

Part of Step 1 is the computation of the priority of an oct-tree node, which requires us to scan through all the voxels contained in the node. Our implementation computes the priorities of all possible oct-tree vertices that can appear during the subdivision in a bottom-up fashion. This requires us to store the completely subdivided oct-tree, B_{vox}, which occupies a lot of memory and also creates some computational overhead. On the other hand, all priorities can be computed by scanning the input voxels only once. Also, this step can be seen as a preprocessing step; the persistence diagram of more and more refined oct-trees can afterwards be computed without reconsidering the input data again. Table 1 displays results for the case of one plant root, using uniform thresholding (the results are essentially the same for all tested instances). We see that the running times for getting the external nodes is negligible, once the node priorities are known. Computing priorities (and other overhead caused by B_{vox}) takes less time than scanning the input once. The last column denotes the fitness of the oct-tree, B, where we have normalized the distance function to the unit interval.

Steps 2 and 3 of the algorithm are independent of the image size. The performance of Step 2 is determined by the number of external nodes, whereas Step 3 mainly depends on the number of simplices created in Step 2. In all tested examples, the ratio between produced simplices and input nodes was in a range between 23 and 26, closely matching the upper bound proved in Section 3.2. Table 2 lists the running times for computing the simplicial complex and its persistence diagram. We conclude that the time needed to construct the complex is linear in the number of external nodes, as expected. Computing persistence needs more time, but it still shows a roughly linear behavior. This is consistent with previous observations that the running time of the persistence algorithm seems to be roughly linear although it has been shown to be cubic in the worst case [13]. In summary, we observe that all operations other than scanning the input voxels take time roughly linear in the number of external nodes of the approximating oct-tree.

#Nodes	#Simplices	Ratio	t_{cplx}	t_{pers}	#dots
40K	1,004K	25.11	1.33	4.85	25
80K	2,008K	25.10	2.69	10.20	39
160K	4,037K	25.23	5.31	20.84	55
320K	8,170K	25.53	10.26	42.58	86
640K	16,250K	25.39	21.10	97.45	255
1,280K	32,678K	25.53	42.33	205.47	310

Table 2: We write t_{cplx} for the time to compute the dual simplex of an oct-tree, and t_{pers} to compute its persistence. The number of simplices in the complex is shown, as well as the ratio between number of simplices and number of external nodes. Finally, the number of dots in the resulting persistence diagram is given. All timings are in seconds.

6 Discussion

The main contributions of this paper are a fast and hierarchical algorithm for computing persistence and robustness information for 3-dimensional image data, a detailed analysis of its accuracy depending on the accuracy of the image representation, and an application of these tools to root systems of agricultural plants. The results raise a number of questions:

To what extent is the observed performance of our algorithm dependent on design details? How would it change if we use mean and standard deviation of function values to control the construction of the oct-tree? Can a subdivision rule that aims at improving the piecewise linear as opposed to the piecewise constant approximation improve the performance? Do the results reported in this paper generalize to 4- and higher-dimensional images? In particular, is the dual complex of a balanced d-dimensional orth-tree geometrically realized in $\mathbb{R}^d$? How does the time and memory required to compute the persistence diagram scale as the dimension increases?

Regarding the software part, it would be nice to have a mapping between the persistence diagram and the 3-dimensional image. This could be achieved by displaying the simplices that give birth and death to a homology class represented by a user-selected dot. A somewhat more advanced implementation would display a representative cycle of the class. The authors feel it is important to mention that the tools described in the paper are in no way tailored to the root systems data studied in Section 5. Indeed, they apply to any voxel data, including medical images.

Acknowledgments

We thank Philip Benfey's Lab at Duke University for sharing the photographs of their rice root systems. We thank Ying Zheng for sharing her software that reconstructs a 3-dimensional model from a series of photographs. We thank Dmitriy Morozov for sharing his Dionysus software implementing persistent homology for simplicial complex data and for answering our questions regarding this software. The first and third authors thank Duke University for hosting them during the Spring semester of 2010.

References

[1] C. L. Bajaj, A. Gillette and S. Goswami. Topology based selection and curation of level sets. In *Topology-Based Methods in Visualization II*, H.-C. Hege, K. Polthier, G. Scheuermann (eds.), Springer-Verlag, 45–58, 2009.

[2] C. L. Bajaj, V. Pascucci and D. R. Schikore. The contour spectrum. *In* "Proc. 8th IEEE Conf. Visualization, 1997", 167–173.

[3] P. Bendich, H. Edelsbrunner, D. Morozov and A. Patel. Robustness of level and interlevel sets. Manuscript, IST Austria, Klosterneuburg, Austria, 2009.

[4] M. Bern, D. Eppstein and J. Gilbert. Provably good mesh generation. *J. Comput. Sys. Sci.* **48** (1994), 384–409.

[5] G. Carlsson, T. Ishkhanov, V. de Silva and A. Zomorodian. On the local behavior of spaces of local images. *Internat. J. Comput. Vision* **76** (2008), 1–12.

[6] A. CERRI, M. FERRI AND D. GIORGI. Retrieval of trademark images by means of size functions. *Graphical Models* **68** (2006), 451–471.

[7] M. K. CHUNG, P. BUBENIK AND P. T. KIM. Persistence diagrams of cortical surface data. In *Information Processing in Medical Imaging*, Springer-Verlag, LNCS **5636**, 2009, 386–397.

[8] V. DE SILVA AND R. GHRIST. Coverage in sensor networks via persistent homology. *Alg. Geom. Topology* **7** (2007), 339–358.

[9] M.-L. DEQUÈANT, S. AHNERT, H. EDELSBRUNNER, T. M. A. FINK, E. F. GLYNN, G. HATTEM, A. KUDLICKI, Y. MILEYKO, J. MORTON, A. R. MUSHEGIAN, L. PACHTER, M. ROWICKA, A. SHIU, B. STURMFELS AND O. POURQUIÉ. Comparison of pattern detection methods in microarray time series of the segmentation clock. *PLoS ONE* **3** (2008), e2856, doi:10.1371/journal.pone.0002856.

[10] H. EDELSBRUNNER AND J. L. HARER. *Computational Topology. An Introduction.* Amer. Math. Soc., Providence, Rhode Island, 2009.

[11] H. EDELSBRUNNER, D. MOROZOV AND A. PATEL. Quantifying transversality by measuring the robustness of intersections. Manuscript, Dept. Comput. Sci., Duke Univ., Durham, North Carolina, 2009.

[12] A. GYULASSY, V. NATARAJAN, V. PASCUCCI, P.-T. BREMER AND B. HAMANN. A topological approach to simplification of three-dimensional scalar functions. *IEEE Trans. Vis. Comput. Graph.* **12** (2006), 474–484.

[13] D. MOROZOV. Persistence algorithm takes cubic time in worst case. In *BioGeometry News*, Dept. Comput. Sci., Duke Univ., Durham, North Carolina, 2005.

[14] J. R. MUNKRES. *Elements of Algebraic Topology.* Perseus, Cambridge, Massachusetts, 1984.

[15] T. S. NEWMAN AND H. YI. A survey of the marching cube algorithm. *Computers and Graphics* **30** (2006), 854–879.

[16] V. PASCUCCI, G. SCORZELLI, P.-T. BREMER AND A. MASCARENHAS. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. Graphics* **26** (2007), 58.

[17] H. SAMET. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Massachusetts, 1990.

[18] M. SONKA, V. HLAVAC AND R. BOYLE. *Image Processing, Analysis and Machine Vision.* Second edition, PWS Publishing, Pacific Grove, California, 1999.

[19] M. VAN KREFELD, R. VAN OOSTRUM, C. L. BAJAJ, V. PASCUCCI AND D. R. SCHIKORE. Contour trees and small seed sets for isosurface traversal. *In* "Proc. 13th Ann. Sympos. Comput. Geom., 1997", 212–220.

[20] Y. WANG, P. K. AGARWAL, P. BROWN, H. EDELSBRUNNER AND J. RUDOLPH. Coarse and reliable geometric alignment for protein docking. *In* "Proc. Pacific Sympos. Biocomput., 2005", 65–75.