Memory-Efficient Computation of Persistent Homology for 3D Images using Discrete Morse Theory

David Günther, Jan Reininghaus, Ingrid Hotz
Zuse Institute Berlin
Berlin, Germany
david.guenther@zib.de reininghaus@zib.de
hotz@zib.de

Hubert Wagner
Institute of Computer Science
Jagiellonian University
Krakow, Poland
hubert.wagner@ii.uj.edu.pl

***Abstract*—We propose a memory-efficient method that computes persistent homology for 3D gray-scale images. The basic idea is to compute the persistence of the induced Morse-Smale complex. Since in practice this complex is much smaller than the input data, significantly less memory is required for the subsequent computations. We propose a novel algorithm that efficiently extracts the Morse-Smale complex based on algorithms from discrete Morse theory. The proposed algorithm is thereby optimal with a computational complexity of $O(n^2)$. The persistence is then computed using the Morse-Smale complex by applying an existing algorithm with a good practical running time. We demonstrate that our method allows for the computation of persistent homology for large data on commodity hardware.**

***Keywords*-persistent homology, Morse-Smale complex, discrete Morse theory, large data**

I. Introduction

It is clear that with the rapid increase of the amount of data produced, availability of efficient tools to analyze these data is of great importance. Computational topology [1], due to its ability to extract essential features of the analyzed data, is becoming a widely-used method. In particular, persistent homology introduced by Edelsbrunner et al. [2] has drawn much attention, since it robustly extracts the topological structure of the data.

While algorithms with good practical running times have been proposed [3], exact computation of persistence for large 3D image data remains a challenging problem due to huge memory requirements.

Forman's discrete Morse theory [4], [5], which is the theoretical foundation of our algorithm, allows us to reduce data in a way which preserves the topological structure. This representation of the data, called the Morse-Smale complex, is much more compact but still contains all the necessary topological information for persistent homology computation.

Inspired by the work of Robins et al. [6], we use discrete Morse theory to compute persistent homology. Our main contribution over their approach, is an *efficient* and *optimal* algorithm to compute the Morse-Smale complex and exploiting the regular structure of the cubical complex induced by the image data.

To the best of our knowledge, our algorithm is the first optimal algorithm for the computation of the Morse-Smale complex. Its worst case computational complexity is $O(cn) \subset O(n^2)$, where n denotes the size of the input data and c the number of its critical points.

We present results of an efficient implementation, which show that our algorithm is suitable for real-world applications. The method introduced in this paper, allows for memory-efficient computation of persistent homology of large 3D images. For example, we only need about 32GB of memory for a data set of size $1120 \times 1131 \times 1552$, in contrast to the 500GB that would be necessary using standard algorithms.

The remaining part of this paper is organized as follows. The related research is described in Section II. In Section III the theoretical background of persistence and discrete Morse theory is introduced. In Sections IV and V we present our method and show computational results. Finally, we summarize the paper with a brief discussion in Section VI.

II. Related Work

Persistence: We will focus on previous work on *computing* persistence. For general applications of persistence see [1]; for application in the context of image data, see [7], [8].

The standard, algebraic algorithm [1] for persistence has cubic running time in the size of the input (i.e. image). While an example was constructed by Morozov [9], showing that this pessimistic execution can actually occur, the behavior of this algorithm is only slightly super-linear in practical situations [3].

When focusing on 0-dimensional homology, union-find data structures can be used to compute persistence in time $O(n\alpha(n))$ [1], where α is the inverse of the Ackermann functions and n is the input size.

Milosavljevic et al. [10] computed persistent homology in matrix multiplication time $O(n^\omega)$ where the currently best estimation of ω is 2.376. Chen and Kerber [11] proposed a randomized algorithm to compute only pairs with persistence above a chosen threshold. Despite showing better theoretical

complexity, it is unclear whether these methods are better than the standard persistence algorithm in practice.

A recent variation of the standard algebraic algorithm [1], called *killing*, introduced by Chen and Kerber [3] significantly reduces the amount of computations. This idea was also used in [12], to compute persistence for $n-$dimensional images.

In general, purely algebraic methods suffer from high memory requirements. In our approach, we alleviate this effect, by reducing the size of data.

Discrete Morse Theory: Morse Theory [13] is a mathematical theory which relates the topology of the domain of a function with critical points of this function. For example, every continuous function defined on a sphere has at least one critical point. The set of critical points extracted should therefore satisfy the constraints described by Morse theory. Note that due to the global nature of topological consistency it is difficult to enforce these constraints in local numerical algorithms. Fortunately, Forman [4], [5] developed a discrete version of Morse theory, which allows for algorithms that provably result in a consistent set of critical points.

The first such algorithm was proposed by Lewiner et al. [14], [15] who also conjectured that persistence could be efficiently computed using discrete Morse theory. Recently, several other such algorithms were suggested [16], [17], [18]. Gyulassi et al. [19] introduced a fast streaming approach to extract the essential critical points of large data. Its complex is herein iteratively simplified to differentiate between spurious and important critical points. However, this approach is not suited for exact persistence computation since not all points in his complex can be paired.

For our method, we build on the method by Robins et al. [6], the first algorithm which is provably correct in 3D, in a sense that the computed critical points correspond one-to-one to the topological changes in the sub-level sets of the image data.

III. Theoretical Background

Complexes: The input of the persistent homology computation is a 3D gray-scale image: an array $\Omega = m \times n \times \ell$ and a function $f : \Omega \to \mathbb{R}$. To capture the topological information, we need to represent this as a *complex*, which is a decomposition of a space into *cells* of *different dimensions*. See Figure 1a) for an example. During the first part of computations we use *cubical complexes* [20], whose cells consist of vertices, edges, squares and full cubes. The *Morse-Smale* complex we extract later belongs to the class of *CW-complexes*, which is more general and its p-cells are only required to be homeomorphic to spheres of dimension p [21].

Boundary maps and matrices: Cells of different dimensions are connected by boundary relations. For example, the boundary of an edge $E = (a, b)$ are the vertices a and b. If a $(p-1)$-cell α is in the boundary of a p-cell β, we say α is a proper face of β. Note that if a complex contains a cell c, it must also contain all the faces of c.

For any p-dimensional cell c, its *boundary*, denoted by $\partial_p c$, is the set of its $(p-1)$-dimensional faces. We now define this relation algebraically. Let *p-chain* be a formal sum of p-cells with $\mathbb{Z}_2$ coefficients (other groups of coefficients can be used, but this one is the most suitable for our task). This enables us to extend the boundary operator linearly to p-chains. For any p-chain $c = \sum a_i c_i$, we have $\partial_p c = \sum a_i \partial_p c_i$. The p-chains, together with (modulo 2) addition form a *group of p-chains*, denoted by C_p.

If we specify a unique index for each cell, a p-chain corresponds to a vector in $\mathbb{Z}_2^{n_p}$, where n_p is the number of p-dimensional cells in the complex. The p-dimensional boundary operator ∂_p can be written as a $n_p \times n_{p-1}$ binary matrix (also denoted ∂_p) whose columns are the boundaries of the p-cells.

The above is summarized by the *chain complex*, which can be viewed as an *algebraic representation* of a complex C [4]

$$C : C_3 \xrightarrow{\partial_3} C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0. \tag{1}$$

Filtration: For a given complex K, a filtration is a nested sequence of complexes: $\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n = K$. In our case, it is induced by the input data $f : \Omega \to \mathbb{R}$ as follows. First, the values given by f on the 0-cells of K, are extended to all cells of K by a so-called *lower-star filtration*: each cell is assigned the maximum function value of the vertices it contains. The filtration of K with respect to f is then defined by the sub-level complexes $K^t = f^{-1}(-\infty, t]$. Imagine that we start with an empty complex and at each step of the filtration one or more cells are added.

Persistence: First, we will give a basic intuition behind homology and persistent homology. For this paper, we can say that homology detects topological features: connected components, tunnels, and voids for a *fixed* thresholding (sub-level set) of a gray-scale image. *Persistent* homology, in turn, describes the *evolution* of topological features looking at consecutive thresholds.

More precisely, given a complex K and a *filtering function* $f : K \to \mathbb{R}$, *persistent homology* studies homological changes of the sub-level complexes, $K^t = f^{-1}(-\infty, t]$. The algorithm captures the birth and death times of homology classes of the sub-level complexes, as the threshold t grows from $-\infty$ to $+\infty$. By birth, we mean that a homology feature comes into being; by death, we mean it either becomes trivial or becomes identical to some other class born earlier. The *persistence*, or lifetime of a class, is the difference between the death and birth times. Homology classes with larger persistence reveal information about the global structure of the space K, described by the function f.

The overall output of the computations is the list of *persistence pairs* of the form (birth, death). This information can be visualized in different ways. One well-accepted idea is the persistence diagram [22], which is a set of points in a two-dimensional plane, each corresponding to a persistent homology class. The coordinates of such a point are the birth and death time of the related class.

An important justification of the usage of persistence is the stability theorem. Cohen-Steiner et al. [22] proved that for any two filtering functions f and g, the difference of their persistence is always upperbounded by the L^∞ norm of their

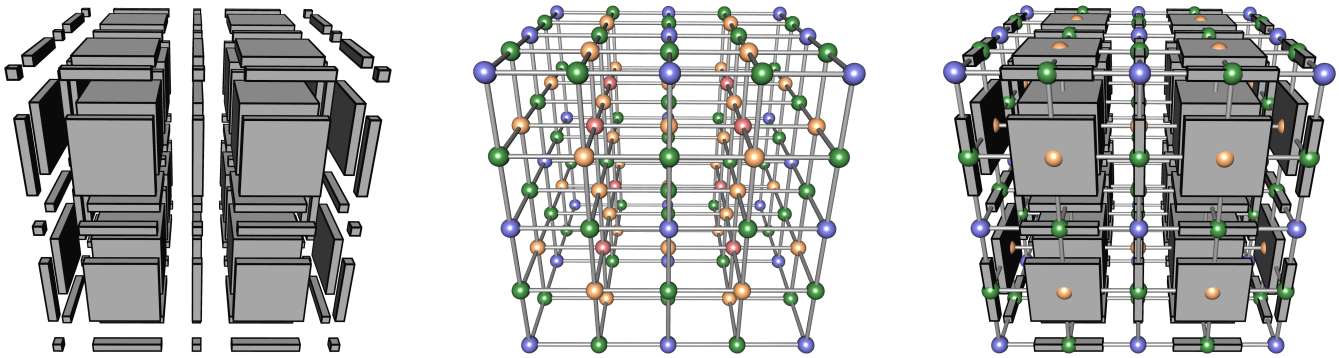

Fig. 1. Illustration of a cubical complex and its derived cell graph. Image a) shows the cells of a small uniform grid in an exploding view. A single voxel is represented by eight 0-cells, twelve 1-cells, six 2-cells, and one 3-dimensional cell. These cells and their boundary relation define the cubical complex C. Image b) shows the derived cell graph G_C. The nodes representing the 0-, 1-, 2-, and 3-cells are shown as blue, green, yellow and red spheres respectively. The adjacency of the nodes is given by the boundary relation of the cells. Image c) shows the cubical complex and the cell graph illustrating the neighborhood relation of the cells.

difference:

$$\|f - g\|_\infty := \max_{x \in K} |f(x) - g(x)|. \tag{2}$$

This enables robust estimation of how persistence is affected by perturbation of the input (e.g. adding noise). Also, this guarantees that persistence can be used as a signature. Whenever two persistence outputs are different, we know that the functions are definitely different.

Discrete Morse Theory: In the following we assume that a three dimensional cubical complex C is given. We use the lower-star filtration defined above, to extend the input function to all cells. The *cell graph* $G_C = (N, E)$ encodes the combinatorial information contained in C. The nodes N of the graph consist of the cells of the complex C and each node u^p is labeled with the dimension p of the cell it represents. The edges E of the graph encode the neighborhood relation of the cells in C. If the cell u^p is in the boundary of the cell w^{p+1}, then $e^p = \{u^p, w^{p+1}\} \in E$. We label each edge with the dimension of its higher dimensional node. An illustration of a cell graph is shown in Figure 1b). Note that the node indices, their adjacency and their geometric embedding in $\mathbb{R}^3$ are given implicitly by the regular grid structure of Ω.

A subset of pairwise non-adjacent edges is called a *matching* $M \subset E$. Using these definitions, a *combinatorial gradient field* V on a regular cell complex C can be defined as a certain acyclic matching of the cell graph G_C [23]. The set of combinatorial gradient vector fields on C is given by the set of these matchings, i.e., the set of *Morse matchings* $\mathcal{M}^\phi$ of the cell graph G_C. An illustration of a 2D Morse matching is shown in Figure 2b).

We now define the extremal structures of a combinatorial gradient vector field V in G_C. The unmatched nodes are called *critical nodes*. If u^p is a critical node, we say that it has *index* p. A *critical node* of index p is called minimum $(p = 0)$, 1-saddle $(p = 1)$, 2-saddle $(p = 2)$, or maximum $(p = 3)$. A combinatorial *p-streamline* is a path in the graph whose edges are of dimension p and alternate between $V \subset E$ and its complement, $E \setminus V$. In a Morse matching, there are no closed p-streamlines. This defines the acyclic constraint for Morse matchings. A p-streamline connecting two critical nodes is called a *p-separatrix*. A *p-separation surface* is given by all combinatorial 2-streamlines that emanate from a critical point of index p. An illustration of extremal structures is shown in Figure 2.

Using the above definitions, we can now define the chain complex associated to the *Morse-Smale complex* C_V with coefficients in $\mathbb{Z}_2$

$$C_V : C_3 \xrightarrow{\partial_3} C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0. \tag{3}$$

The Morse-Smale complex C_V is induced by a combinatorial gradient field $V \subset E$ in a cell graph $G_C = (N, E)$. The chain groups C_ℓ are generated by the critical nodes of V with index ℓ. The boundary maps ∂_ℓ are defined by the

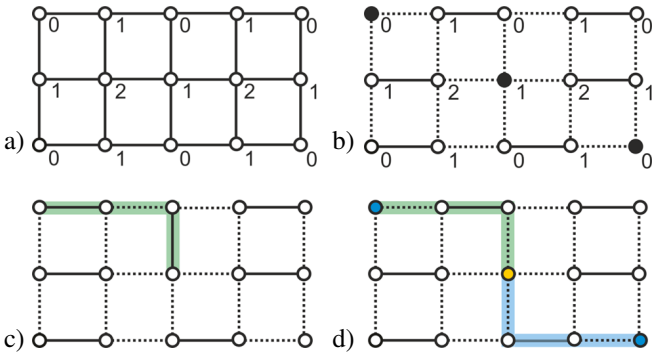

Fig. 2. Basic definitions of discrete Morse theory: a) the cell graph G_C, the node labels indicate the dimension of the represented cells; b) a combinatorial gradient field V defined on G_C, the edges contained in V are depicted by solid lines, the unmatched nodes – the critical nodes – are shown as black spheres; c) a combinatorial streamline alternating between V and its complement; d) two 1-separatrices of V (blue and green) emanating at a 1-saddle (yellow) and ending in a minimum (blue).

combinatorial streamlines of V: if $u^\ell \in C_\ell$ is connected to $w^{\ell-1} \in C_{\ell-1}$ by an *odd number* of combinatorial streamlines, then $w^{\ell-1}$ is in the boundary of u^ℓ. This is a version of a general formula by Forman [4], simplified for $\mathbb{Z}_2$ coefficients.

Forman proves, that the homology of C is always isomorphic to the homology of C_V [4]. If the critical nodes contained in V correspond one-to-one to the topological changes in the sub-level complexes the persistent homology of C therefore coincides with the persistent homology of C_V [6].

Since, in practice, C_V is a lot smaller than C, we can use discrete Morse theory to devise a memory-efficient algorithm for persistent homology.

IV. Method

In this section we describe our overall algorithmic pipeline to compute persistent homology in a memory-efficient manner. The input of the pipeline consists of a 3D gray-scale image. It is represented by a 3D array $\Omega = m \times n \times \ell$ and a function $f : \Omega \to \mathbb{R}$. To compute persistent homology, we initially represent Ω by the cubical complex C.

The pipeline consists of three steps. In Section IV-A we describe the construction of the discrete gradient field V associated to C and f. In Section IV-B, we propose a novel algorithm to extract the Morse-Smale complex C_V defined by V. For completeness, we also describe the computation of the persistent homology of C_V in Section IV-C. We conclude this section with a brief analysis of the computational complexity, memory consumption, and some implementational details in Section IV-D and IV-E.

A. Discrete Gradient Field

To compute the discrete gradient vector field V, we use the algorithm *ProcessLowerStar* [6]. The basic idea of this algorithm is to apply simple homotopic expansions in the lower star of each 0-node. The algorithm results in a combinatorial gradient field V whose critical nodes coincide with the changes of the topology of the sub-level complexes of C. For more algorithmic details and the proof of this property, we refer the interested reader to [6].

B. Morse-Smale Complex Extraction

We now describe how we compute the chain complex associated to the Morse-Smale complex (3), induced by the combinatorial gradient field $V \subset E$.

While the chain groups C_ℓ can be easily extracted from N by collecting the nodes not covered by V, efficient computation of ∂_ℓ is challenging.

Algorithm 1 shows a simple approach introduced in [6] to compute ∂_ℓ with a worst case complexity of $O(n^3)$, where n denotes the number of vertices of C. Its input consists of the cell graph $G_C = (N, E)$, a discrete gradient field $V \subset E$, a flag j, and the index ℓ of the resulting boundary map ∂_ℓ. If $j = 0$, the algorithm computes ∂_ℓ by finding the boundaries of the elements contained in C_ℓ. If $j = 1$, the algorithm computes ∂_ℓ by finding the *co*-boundaries of the elements contained in $C_{\ell-1}$. Note that both cases result in the same ∂_ℓ, the choice of j only affects the running time, see Section IV-E.

Algorithm 1 ComputeBoundary_A(G_C, V, j, ℓ)

Input: $G_C = (N, E), V \subset E, j \in \{0, 1\}, \ell \in \{1, 2, 3\}$
Output: Binary matrix ∂_ℓ

```
1:  E_ℓ ← {e^p ∈ E : p = ℓ}
2:  for all c^p ∈ C_{ℓ−j} do
3:      Q.push({c^p, false})
4:      while Q ≠ ∅ do
5:          {u^p, flag} ← Q.pop()
6:          W ← AlternatingEdges(G_C, V, u^p, flag)
7:          W ← W ∩ E_ℓ
8:          for all {u^p, w^k} ∈ W do
9:              Q.push({w^k, ¬flag})
10:             if w^k ∈ C_V then
11:                 if k < p then
12:                     ∂_ℓ(c^p, w^k) ← ∂_ℓ(c^p, w^k) + 1
13:                 else
14:                     ∂_ℓ(w^k, c^p) ← ∂_ℓ(w^k, c^p) + 1
```

The main idea of Algorithm 1 is to start a breadth-first search in each critical node, which possibly results in multiple traversals of each node. The breadth-first search is constrained by the definition of a combinatorial streamline – the edges of a traced path must always alternate with respect to V. This is encoded in Algorithm 2 which is called in Line 6. Note that the additions in Line 12 and 14 are modulo 2.

Algorithm 2 AlternatingEdges$(G_C, V, u^p, flag)$

Input: $G_C = (N, E), V \subset E, u^p \in N, flag \in \{false, true\}$
Output: $W \subset E$

```
1: if flag = true then
2:     W ← {{u^p, w^k} ∈ E : {u^p, w^k} ∈ V}
3: else
4:     W ← {{u^p, w^k} ∈ E : {u^p, w^k} ∈ E \ V}
```

We now present our novel method to compute ∂_ℓ with a worst case complexity of $O(n^2)$. The main idea of the algorithm is the following. We first collect all critical (unmatched) nodes in V. For each of these nodes we then integrate the corresponding manifolds to collect the critical nodes in the respective (co-)boundaries. The challenging task is now to check whether these nodes are connected by an odd number of separatrices. If this is the case, these nodes are connected in the sense of $\mathbb{Z}_2$ and are inserted in the boundary matrix. To

Algorithm 3 ComputeBoundary_B(G_C, V, j, ℓ)

Input: $G_C = (N, E), V \subset E, j \in \{0, 1\}, \ell \in \{1, 2, 3\}$
Output: Binary matrix ∂_ℓ

```
1: for all c^p ∈ C_{ℓ−j} do
2:     S ← GetManifold(G_C, V, c^p, ℓ)
3:     I ← GetIntersection(G_C, V, S, ℓ, j)
4:     C_c ← CountPaths(G_C, V, I, c^p)
5:     for all w^k ∈ C_c do
6:         if k < p then
7:             ∂_ℓ(c^p, w^k) ← ∂_ℓ(c^p, w^k) + 1
8:         else
9:             ∂_ℓ(w^k, c^p) ← ∂_ℓ(w^k, c^p) + 1
```

count the number of connections, we compute the multiplicity of paths from one critical node to an other critical node but restricted to the intersection of the corresponding manifolds. This main idea is given in Algorithm 3.

For notational simplicity, we only describe the algorithm in detail for $j = 0$ – we consider the boundary of $c^\ell \in C_\ell$.

We first compute the edges $S \subset E$ that are covered by the combinatorial ℓ-streamlines emanating from c^ℓ (Line 2) using Algorithms 4 and 7.

Algorithm 4 GetManifold(G_C, V, c^p, ℓ)

Input: $G_C = (N, E), V \subset E, c^p \in N, \ell \in \{1, 2, 3\}$
Output: $S \subset E$
1: $E_\ell \leftarrow \{e^k \in E : \ k = \ell\}$
2: $S \leftarrow AlternatingRestrictedBFS(G_C, V, E_\ell, c^p)$

We then collect all critical nodes $C_S \subset C_{\ell-1}$ that are covered by S. These nodes are the possible boundary nodes of c^ℓ. To compute the boundary of c^ℓ we need to count the number of combinatorial streamlines connecting c^ℓ with $c^{\ell-1} \in C_S$.

To do this efficiently, we first compute the set of edges $I \subset S$ of all combinatorial streamlines connecting c^ℓ with C_S (Line 3) using Algorithm 5 and 7.

Algorithm 5 GetIntersection(G_C, V, S, ℓ, j)

Input: $G_C = (N, E), V \subset E, S \subset E, \ell \in \{1, 2, 3\}, j \in \{0, 1\}$
Output: $I \subset E$
1: $C_S \leftarrow \{u^{\ell-1+j} \in N : \ \exists\{u^{\ell-1+j}, w^k\} \in S\} \cap C_{\ell-1+j}$
2: $I \leftarrow \emptyset$
3: **for all** $c^p \in C_S$ **do**
4: $\quad S \leftarrow S \setminus I$
5: $\quad I \leftarrow I \cup AlternatingRestrictedBFS(G_C, V, S, c^p)$

For each node $c^{\ell-1} \in C_S$ we then need to count the number of paths in I that connect $c^{\ell-1}$ to c^ℓ. This is done in Line 4 using a simple graph algorithm shown in Algorithm 6. Since we are only interested in the Morse-Smale complex with coefficients in $\mathbb{Z}_2$, it suffices to count the number of paths modulo 2. This is obtained by taking the symmetric difference $\triangle$ in Line 12.

C. Persistence

To compute persistence we use the standard algebraic algorithm [1] with a modification by Chen and Kerber [3]. This algorithm operates on a boundary matrix, ∂_ℓ, of the cubical complex C, representing the input data. A *reduced matrix* is computed, from which the list of *persistent pairs*, as defined in Section III, can be easily read. The performance modification introduced by Chen and Kerber exploits the fact that cells being the *creators* of homology classes are zeroed in the reduced matrix. We refer the reader to [3] for more details.

In contrast to previous work [3], [12], we apply the matrix reduction algorithm to the Morse-Smale complex C_V instead of the initial cubical complex C. Since C_V is much smaller than C in typical situations, storing the boundary matrices consumes significantly less memory (see Table I).

Algorithm 6 CountPaths(G_C, V, I, c^p)

Input: $G_C = (N, E), V \subset E, I \subset E, c^p \in N$
Output: $C_c \subset N$
1: $C_V \leftarrow \{u^p \in N : \ \nexists\{u^p, w^k\} \in V\}$
2: $P \leftarrow \emptyset$
3: $L \leftarrow \{c^p\}$
4: $Q.push(\{c^p, false\})$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad \{u^p, flag\} \leftarrow Q.pop()$
7: $\quad P \leftarrow P \cup u^p$
8: $\quad W \leftarrow AlternatingEdges(G_C, V, u^p, flag)$
9: $\quad W \leftarrow W \cap I$
10: $\quad$ **for all** $\{u^p, w^k\} \in W$ **do**
11: $\quad\quad$ **if** $u^p \in L$ **then**
12: $\quad\quad\quad L \leftarrow L \triangle w^k$
13: $\quad\quad Z \leftarrow AlternatingEdges(G_C, V, w^k, flag)$
14: $\quad\quad Z \leftarrow Z \cap I$
15: $\quad\quad N_Z \leftarrow \{z^q \in N : \ \exists\{z^q, w^k\} \in Z\}$
16: $\quad\quad$ **if** $N_Z \subset P$ **then**
17: $\quad\quad\quad Q.push(\{w^k, \neg flag\})$
18: $C_c \leftarrow L \cap C_V \setminus c^p$

Algorithm 7 AlternatingRestrictedBFS(G_C, V, R, c^p)

Input: $G_C = (N, E), V \subset E, R \subset E, c^p \in N$
Output: $T \subset R \subset E$
1: $T \leftarrow \emptyset$
2: $Q.push(\{c^p, false\})$
3: **while** $Q \neq \emptyset$ **do**
4: $\quad \{u^p, flag\} \leftarrow Q.pop()$
5: $\quad W \leftarrow AlternatingEdges(G_C, V, u^p, flag)$
6: $\quad W \leftarrow (W \cap R) \setminus T$
7: $\quad$ **for all** $\{u^p, w^k\} \in W$ **do**
8: $\quad\quad T \leftarrow T \cup \{u^p, w^k\}$
9: $\quad\quad Q.push(\{w^k, \neg flag\})$

D. Computational complexity

We now give a brief analysis of the computational complexity of our method. We denote the number of vertices of C by n and the number of critical nodes in a combinatorial gradient field by c. Note that the pseudo code shown in the algorithms in this section has been optimized for compactness and clarity instead of a best computational complexity. In the following analysis, we consider an optimal implementation of these algorithms. The realization of such an implementation from the pseudo code only poses some minor technical difficulties.

The complexity for the construction of the combinatorial gradient field using the algorithm proposed in [6] is $O(n)$ – for each node of index 0 we only work on its lower star which has a constant size in the case of cubical complexes.

Analyzing the complexity of the Morse-Smale complex extraction described in Algorithm 3 is more intricate. The loop in Line 1 is executed $O(c)$ often. The complexity of its body is given by the sum of the complexities of the invoked subfunctions.

We start with Algorithm 7. Due to Line 6, the union in Line 8 is disjoint, which implies that the complexity of Algorithm 7

TABLE I

RUNNING TIMES AND MEMORY CONSUMPTION FOR 3D IMAGES OF DIFFERENT SIZE AND TOPOLOGICAL COMPLEXITY. THE FOURTH COLUMN SHOWS THE TOTAL MEMORY CONSUMPTION OF OUR METHOD AND THE MEMORY REQUIREMENT OF A STANDARD PERSISTENCE METHOD AS WELL AS THE PEAK MEMORY CONSUMPTION OF [12]. THE RUNNING TIMES FOR THE CONSTRUCTION OF THE COMBINATORIAL GRADIENT FIELDS (USING 48 CORES), THE BOUNDARY MATRIX, AND THE PERSISTENCE COMPUTATION IS SHOWN IN THE FIFTH COLUMN. THE TOTAL RUNNING TIME OF OUR METHOD IS COMPARED TO ANOTHER PERSISTENCE ALGORITHM [12] IN THE LAST COLUMN.

Data	**Size**	**# critical nodes**	**Total memory** (MB)			**Individual times** (sec)			**Total time** (min)	
			standard	this paper	[12]	IV.A	IV.B	IV.C	this paper	[12]
Silicium	$98 \times 34 \times 34$	1'237	29	2	30	2	1	1	0.03	0.09
Fuel	$64 \times 64 \times 64$	773	66	4	82	4	1	1	0.06	0.02
Neghip	$64 \times 64 \times 64$	6'671	66	5	82	4	12	1	0.25	0.03
Hydrogen	$128 \times 128 \times 128$	30'409	528	35	538	26	43	1	1.15	0.62
Engine	$265 \times 256 \times 128$	1'350'001	2112	194	2127	112	963	3	17.95	1.37
Christmas Present	$246 \times 246 \times 221$	6'544'279	3367	496	3112	184	823	402	23.46	266.78
Aneurysm	$256 \times 256 \times 256$	90'045	4224	289	4250	142	6962	1	118.41	3.52
Bonsai	$256 \times 256 \times 256$	413'561	4224	317	4250	145	13214	2	222.68	2.91
Foot	$256 \times 256 \times 256$	2'044'799	4224	434	4250	142	18199	6	305.79	2.85
Supine	$512 \times 512 \times 426$	34'151'733	28116	3534	26133	1030	2274	119	57.07	24.94
Prone	$512 \times 512 \times 463$	35'942'153	30558	3422	28406	1616	2433	107	69.29	36.34
Christmas Tree	$512 \times 499 \times 512$	63'596'463	32934	4709	*	1266	2950	445	77.70	*
Molecule	$1120 \times 1131 \times 1552$	2'082'895	494967	31950	*	16639	2157	7	313.38	*

is $O(|T|)$. Since Algorithm 4 only calls Algorithm 7, its complexity is $O(|S|)$. The complexity of Algorithm 5 is $O(|I|)$, since due to Line 4, the union in Line 5 is disjoint. Finally, we need to consider the complexity of Algorithm 6. The complexity of the body of the $while$ loop in Line 5 is constant. It therefore suffices to count the number of times that Line 17 is executed. The node w^k is only inserted into Q if all neighboring nodes $z^q \in N_Z$ have already been processed (Line 7 and 16). Since w^k can only be inserted by a neighboring node z^q, it can therefore only be inserted once. The complexity of Algorithm 6 is hence $O(|I|)$, since only nodes contained in I can enter the queue at all.

Note that there holds $O(|I|) \subseteq O(|S|) \subseteq O(n)$. The overall complexity of Algorithm 3 is hence $O(c|I| + c|S|) \subseteq O(cn)$. Since there is a lower bound on the computational complexity for the Morse-Smale complex extraction problem in 3D of $O(n^2)$ [6], our proposed algorithm is optimal.

The choice of j does not affect the overall computational complexity – it only affects the practical running time of the algorithm which is discussed in Section IV-E.

The computational complexity for the matrix reduction algorithm, which we used to compute the persistent homology, is $O(c^2\, n \log n)$ in the case of cubical complexes. Since we apply it to the Morse-Smale complex, the complexity is $O(c^3)$.

The complete complexity for our algorithm is therefore $O(cn + c^3)$.

E. Implementational details

1) Running time: To compute the combinatorial gradient field, the cell graph is decomposed into lower stars of the 0-nodes. Since this is a disjoint decomposition, each lower star can be processed in parallel. Algorithm 3 can also be easily parallelized since the boundaries of the critical points are independent of each other.

We now discuss the influence of j on the running time of Algorithm 3. In 3D, the combinatorial 1-streamlines can only merge, while the 3-streamlines can only split. As shown in [6], the computation of the co-boundaries of all 0-nodes ($j = 1, \ell = 1$) has thereby only a complexity of $O(n)$. The same applies to the boundaries of the 3-nodes ($j = 0, \ell = 3$). In contrast, the computation of ∂_2 has a worst case complexity of $O(n^2)$, regardless of j. The choice of j thereby does not affect the overall complexity of Algorithm 3. The practical running time, however, depends on j. For most inputs, the best choice is $(j = 0, \ell = 1), (j = 0, \ell = 2), (j = 1, \ell = 3)$, since the computation of the (co-) boundaries of the 2- and 1-nodes only amounts to a line integration, as in this setting, $|W| \leq 1$ in Algorithm 7, Line 5.

2) Memory requirements: We only need to compute the boundary matrices ∂_ℓ of the Morse-Smale complex C_V, which does not require much memory. On the other hand, explicit representation of the initial cubical complex C would require enormous amounts of memory. We therefore represent C only implicitly, using the regular structure induced by the grid [12], [18]. The adjacency information represented in the cell graph $G_C = (N, E)$ is always computed on-the-fly using index calculations. Since we enumerate the nodes N and the edges E without gaps, we can represent the combinatorial gradient field V simply by an array of bits of length $|E|$. The sets used in the algorithms depicted in this section can also be represented using such boolean arrays. This allows for efficient set operations. If the data values on the 0-cells of the complex are defined by 32-bit single precision floats, then the total memory overhead factor of our method is about 5 in our current implementation. If running time is less important, one can also implement the sets using a more memory-efficient representation.

V. RESULTS

In the following, we present some examples to illustrate our method. All experiments were performed on a machine with four AMD Opteron 6174 CPUs.

Table I shows the running time and memory consumption for different 3D data sets provided by [24], [25], [26]. We measured the total memory usage of our method as well as the construction time of the combinatorial gradient field,

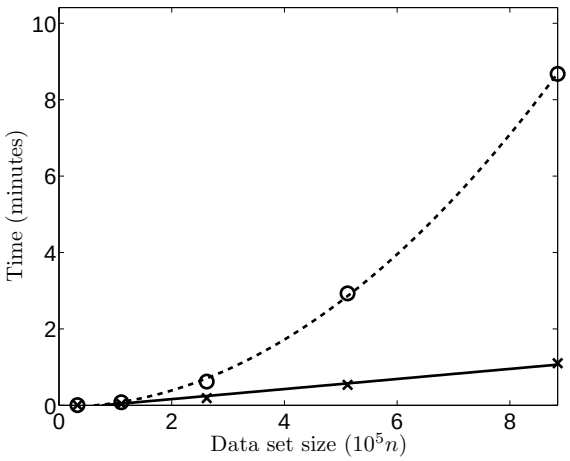

Fig. 3. Comparison of Algorithm 1 and Algorithm 3. The circle and cross markers show the running times over the size of a synthetic data set for Algorithm 1 and Algorithm 3, respectively. The solid line depicts a least-square fitting of a linear function for the cross markers. The dashed line depicts a least-square fitting of a quadratic function for the circle markers.

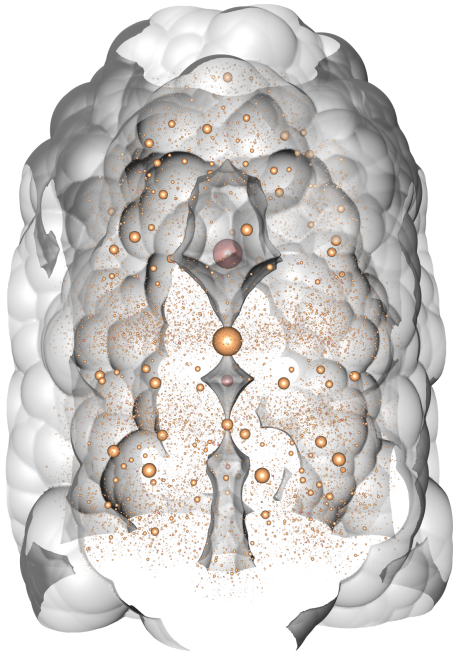

Fig. 4. Distance field of a molecule. An isosurface of a distance field, computed from a molecule, as gray transparent surface is shown. The 277'478 maxima and the 982'578 2-saddles are shown as red and yellow spheres, respectively. Each sphere is scaled by its persistence.

the Morse-Smale complex extraction using Algorithm 3, and the persistence running times. We also included the memory consumption using a standard persistence method working on the boundary matrices of the initial cubical complex, which can be computed exactly. Assuming a sparse representation of the boundary matrix $\partial_\ell : N^\ell \times N^{\ell-1} \to \{0,1\}$, the amount of its memory is given by

$$\sum_{\ell=1}^{3} 8\,|\partial_\ell| + 16\,|N^\ell|$$

where $|\partial_\ell|$ denotes the number of matrix entries and $|N^\ell|$ the number of nodes of type ℓ. The measured peak memory consumption of [12] is also given as reference. In the last column we compared the total running time of our method with the method proposed in [12]. Note that the combinatorial gradient field was constructed in parallel using all 48 cores, while all other steps were single threaded.

The total memory consumption of our method is about a factor 8-17 less than using a standard persistence approach. In practice, the Morse-Smale complex C_V is much smaller than the cubical complex C. This enables the persistence computation of large data. The overall running time is for most of the examples about a factor 2-11 greater than the times presented in [12]. The most time, however, consumes the construction of the boundary matrix, which could be further reduced using a parallel implementation. Although the pessimistic computational complexity is cubical, the running time for the matrix reduction is surprisingly low, when applied to the Morse-Smale complex (see for example the Christmas Present). In general, the running times and the memory consumptions depend on the number of critical nodes and their connectivity, as can be seen for instance at the bonsai or aneurysm example. Although both examples only contain a small number of critical nodes, their connectivity is much more intricate than in the prone or supine data set, for example.

Figure 3 shows the running times of the Morse-Smale complex extraction step using the existing Algorithm 1 and our novel Algorithm 3. The data set is given by sampling an analytic function g on a uniform grid of increasing resolution and adding a small amount of uniform noise in the range of $[-0.5, 0.5]$ to the samples. Algorithm 1 scales quadratically with the number of vertices n in the complex. In contrast, our method scales only linearly. The individual running times of Algorithm 3 applied to the analytic function g sampled on a uniform 96^3 grid are: (IV.A) 9.87 sec, (IV.B) 62.49 sec and (IV.C) 0.77 sec. In contrast, Algorithm 1 needs (IV.B) 521.59 sec.

The reason for this behavior stems from the structure of the data. The function g contains some large scale structures. Adding noise to it results in many critical points and the combinatorial 2-streamlines often merge and split. While this property dramatically increases the practical running time of Algorithm 1, our Algorithm 3 is not affected by this perturbation of the data.

We applied our method to a distance field, computed from a Chaperone protein. The objective is the extraction of the maxima and 2-saddles. While the maxima represent the points with the greatest distance to the atoms, the 2-saddles correspond to the narrow points of the field. These points define the minimal size of an atom to enter the molecule from the outside. The

data set is of dimension $1120 \times 1131 \times 1552$ and contains 2'082'895 critical points. A standard persistence computation would require about 500GB memory. Our approach, in contrast, only requires about 32 GB, and can thereby be applied on commodity hardware. The total running time as well as the memory consumption of this example are shown in Table I.

VI. Conclusion and Future Work

We presented a novel algorithm to extract the Morse-Smale complexes from a given combinatorial gradient field induced by a 3D gray scale image. This allows for a memory-efficient persistent homology computation. As shown in Section IV and V, our algorithm combines many useful properties:

1) Our novel algorithm for the Morse-Smale complex extraction is optimal: $O(cn) \subset O(n^2)$.
2) The overall complexity for the persistence computation is reduced to $O(cn + c^3)$.
3) The computation of persistence using the Morse-Smale complex requires significantly less memory.

There are some limitations of our approach:

1) Extending our techniques to more general inputs like simplicial complexes is possible, but would result in high memory-usage – we heavily exploit the compact representation of the initial, cubical complex.
2) Our current method is limited to three dimensions.
3) For medium-sized data our current implementation can be considerably slower than the algebraic approach.

Despite these drawbacks, we believe that our method enables the application of persistent homology in new fields. Our current implementation can already be used to analyze very large, complex data-sets. While the running-time is currently traded-off for memory-efficiency, our algorithm is more suitable for a parallel implementation, than the faster algebraic approach.

The efficient computation of homological persistence now allows for practical user interfaces that enables an interaction of the user with persistence. This could be helpful in the understanding of specific applications. Potentially, an out-of-core realization further increases the efficiency herein.

It would be also interesting to see how our proposed algorithm scales for higher dimensional data. The challenging part is thereby that the combinatorial gradient field may contain extra superfluous critical points in contrast to the three dimensional case [6].

A fundamental question, which is still an open problem in the homological persistence literature, is the relation of the topological complexity of a given input data and the persistence computation times. Since matrix reduction is a global operation, the structure of the underlying Morse-Smale complex is crucial. This structure also depends on the imaging process and the data format. For instance, the aneurysm and bonsai data are given as 8-bit integer while the prone and supine data are 16-bit integer CT scans. This may also contribute to the different timings shown in Table I.

Acknowledgment

This work was supported by the Max-Planck Institute of Biochemistry, Martinsried, the DFG Emmy-Noether research program and Foundation for Polish Science Geometry and Topology in Physical Models program. We thank Daniel Baum for providing the molecule data set. We also thank Herbert Edelsbrunner and Chao Chen for many fruitful discussions.

References

[1] H. Edelsbrunner and J. Harer, *Computational Topology. An Introduction*. American Mathematical Society, January 2010.
[2] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete & Computational Geometry*, vol. 28, no. 4, pp. 511–533, 2002.
[3] C. Chen and M. Kerber, "Persistent homology computation with a twist," in *27th European Workshop on Comp. Geometry (EuroCG 2011)*, 2011.
[4] R. Forman, "Morse theory for cell complexes," *Advances in Mathematics*, vol. 134, pp. 90–145, 1998.
[5] ——, "A user's guide to discrete Morse theory," in *Seminaire Lotharingien de Combinatoire*, vol. B48c, 2002, pp. 1–35.
[6] V. Robins, P. J. Wood, and A. P. Sheppard, "Discrete Morse theory for grayscale digital images," in *IEEE Transactions on Pattern Analysis and Machine Learning*, 2010, pp. 1–14.
[7] P. Bendich, H. Edelsbrunner, and M. Kerber, "Computing robustness and persistence for images," in *Proceedings of IEEE Visualization*, vol. 16, no. 6, 2010, pp. 1251–1260.
[8] M. Mrozek and T. Wanner, "Coreduction homology algorithm for inclusions and persistent homology," *Computers and Mathematics with Applications, accepted*, 2010.
[9] D. Morozov, "Persistence algorithm takes cubic time in the worst case," in *BioGeometry News*. Duke Computer Science, Durhmam, NC, 2005.
[10] N. Milosavljevic, D. Morozov, and P. Skraba, "Zigzag Persistent Homology in Matrix Multiplication Time," in *Proceedings of the 27th annual symposium on Computational geometry*, 2011.
[11] C. Chen and M. Kerber, "An Output-Sensitive Algorithm for Persistent Homology," in *Proceedings of the 27th annual symposium on Computational geometry*, 2011.
[12] H. Wagner, C. Chen, and E. Vucini, "Efficient computation of persistent homology for cubical data," in *Proc. TopoInVis*, April 2011.
[13] J. Milnor, *Topology from the differentiable viewpoint*. Univ. Press Virginia, 1965.
[14] T. Lewiner, "Geometric discrete Morse complexes," Ph.D. dissertation, Dept. of Mathematics, PUC-Rio, Aug 2005.
[15] T. Lewiner, H. Lopes, and G. Tavares, "Optimal discrete Morse functions for 2-manifolds," *Computational Geometry: Theory and Applications*, vol. 26, no. 3, pp. 221–233, November 2003.
[16] U. Bauer, C. Lange, and M. Wardetzky, "Optimal topological simplification of discrete functions on surfaces," *CoRR*, 2010.
[17] J. Reininghaus, D. Günther, I. Hotz, S. Prohaska, and H.-C. Hege, "TADD: A computational framework for data analysis using discrete Morse theory," in *Mathematical Software – ICMS 2010*. Springer, 2010, pp. 198–208.
[18] D. Günther, J. Reininghaus, S. Prohaska, T. Weinkauf, and H.-C. Hege, "Efficient computation of a hierarchy of discrete 3d gradient vector fields," in *Proc. TopoInVis*, Zürich, Switzerland, April 2011.
[19] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci, "A practical approach to Morse-Smale complex computation: scalability and generality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1619–1626, 2008.
[20] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*, ser. Applied Math. Sciences. Springer-Verlag, 2004, vol. 157.
[21] A. Hatcher, *Algebraic Topology*. Cambridge, U.K.: Cambridge University Press, 2002.
[22] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Disc. and Comp. Geometry*, vol. 37, pp. 103–120, 2007.
[23] M. K. Chari, "On discrete Morse functions and combinatorial decompositions," *Discrete Math.*, vol. 217, no. 1-3, pp. 101–113, 2000.
[24] S. Röttger, http://www9.informatik.uni-erlangen.de/External/vollib/.
[25] The Institute of Computer Graphics and Algorithms http://www.cg.tuwien.ac.at/research/vis/datasets/#Data.
[26] "Volvis: Voxel data repository," 2010, http://www.volvis.org/.