The Discrete Morse Complex of Images: Algorithms, Modeling and Applications

Ricardo Dutra da Silva
Department of Informatics, Federal University of Technology, Brazil
http://www.dainf.ct.utfpr.edu.br/~rdutra/
rdutra@dainf.ct.utfpr.edu.br

Helio Pedrini
Institute of Computing, University of Campinas, Brazil
helio@ic.unicamp.br

Bernd Hamann
Department of Computer Science, University of California, Davis, California, U.S.A.
hamann@cs.ucdavis.edu

Abstract
The Morse complex is used for studying the topology of a function, e.g., an image or terrain height field. We present an algorithm for the computation of the discrete Morse complex of two-dimensional images using an edge-based data structure. By using this data structure, it is possible to perform fast local operations that are important to construct the complex and make the model useful for applications such as visualization, persistent homology computation or construction of a topological hierarchy. We present theoretical and applied results to demonstrate benefits and usages of our method.

2012 ACM Subject Classification Computing methodologies → Image processing

Keywords and phrases Discrete Morse Complex, Image Topology, Cell Complexes


1 Introduction
Morse theory [37] is primarily used for the study of a function’s topology by establishing the topological relationships between its critical points, i.e., its extrema and saddles [18]. Algorithms and models for Morse complexes have become increasingly important as fundamental building blocks of computational topology with applications in image analysis, e.g., for image segmentation, object skeletonization and classification. Relevant literature for Morse-Smale complexes includes [8, 27, 42], discrete Morse complexes are covered in [17, 30, 42, 50] and persistent homology is discussed in [11, 15, 19, 20, 33, 50].

In many image processing problems, functions represent a space with scalar measures over it. This is, for instance, the case of height fields (terrains). The domain of such functions is sampled and, therefore, the continuous Morse theory is not directly applied. The adaptation of the continuous theory to sampled data has been studied in [9, 18, 53], for example, where the authors use the simulation of smooth notions to guide the computation and produce what is called a Morse-Smale complex [53].

The discrete Morse theory, formulated by Forman [21, 22], is another adaptation of the Morse theory to discrete structures. It was explored in works such as King et al. [31] and Lewiner et al. [34]. Robins et al. [44] provided algorithms for computing discrete Morse complexes through an image analysis approach.

Our contribution is an algorithm- and data structure-driven approach to compute the discrete Morse complex of two-dimensional images. The complex is similar to the Morse-Smale [9, 18, 26, 45, 53] approach and the method described by Robins et al. [44]. We
introduce a specialization for the specific case of 2D image functions of the approach
described by Robins et al. [44] and show that the specialized algorithms are optimized for the
case where paths emanate from saddles, instead of using the breadth-first search discussed
in [44]. We contribute a complete, detailed description of an algorithm to extract paths
from a vector field given as input, resulting in the discrete Morse complex of an image. The
idea is similar to the one presented by Edelsbrunner et al. [18] for Morse-Smale complexes;
we differ by using the discrete approach of [44]. In addition, we show that the complexes
can be easily implemented by means of an edge-based data structure. Furthermore, since
we use an edge-based data structure, the complex is suitable for computing topological
persistence [6, 10, 12, 14, 35, 40, 51] and topological hierarchies [43, 49], as well as for
visualization purposes.

In Sections 2 to 4, we present basic concepts on topology of images, discrete Morse
complexes and data structures. These concepts are used in the algorithms and in the
modeling of the resulting complexes, which are discussed in detail in Sections 5 to 7. In
Section 8, we show the application of the discrete Morse complex to compute the persistence
and hierarchical representations of images. Finally, we present some concluding remarks in
Section 9.

2 Images as Cell Complexes and Topological Data Structures

A p-cell, of dimension p, is the basic element to construct the discrete domain which is a cell
complex. It is defined as a topological space homeomorphic to a p-ball \( B^p = \{ x \in \mathbb{R}^p : |x| \leq 1 \} \). The first three low-order p-cells are 0-cells (also known as nodes), 1-cells (edges) and 2-cells
(faces). Even though the secondary names are commonly used in computer science, the
concept of a face is a different one in topology, as we will state soon. Therefore, to be
consistent with the topology and computational topology literature, we choose to use the
primary names. We will denote a p-cell as \( \alpha^p \) or simply \( \alpha \) whenever the dimension is clear
from the context.

The boundary of a p-cell consists of cells of lower dimension that limit the cell. A face of
a p-cell \( \sigma^p \) is a cell \( \tau^k \), with \( k \leq p \), which is part of the boundary of the p-cell. The p-cell \( \sigma^p \)
is called a coface of \( \tau^k \). The bounding relations of face and coface will be stated as \( \tau^k \prec \sigma^p \)
and \( \sigma^p \succ \tau^k \).

A cell complex \( K \) is a finite collection of cells that satisfies the following requirements:
(1) all the faces of a cell in the complex also belong to the complex, and (2) the intersection
of any two cells is either empty or a face of both cells. A p-complex is a cell complex such
that all its cells have dimension less than or equal to \( p \). A subcomplex of a cell complex \( K \) is
a subset of cells \( L \subseteq K \) such that \( L \) is also a cell complex.

An image is viewed as a function \( f : D \rightarrow \mathbb{R} \), defined on a subset of the discrete lattice,
\( D = \{(x,y) \in \mathbb{Z}^2 | 1 \leq x \leq M, 1 \leq y \leq N \} \). A point in \( D \), and its value, is called a
pixel. An image can be modeled by a regular two-dimensional cell complex \( K \), as described
by Kovalevsky [32]. Kovalevsky adopts a model where pixels are 2-cells since both are
area-related elements. The dual of this representation is also commonly used, for example,
in Robins et al. [44]. We adopt this latter model, such that an image is a regular complex \( K \)
with 0-cells corresponding to the pixels in \( D \). The 2-cells are usually squares or triangles
defined by a pixel and some of its closest pixels in \( D \). The 1-cells are faces of the 2-cells. An
example of such a model is shown in Figure 1.

Different data structures can be used to model cell complexes [16, 20, 24, 27, 52]. Since
we are interested in two-dimensional cell complexes, planar edge-based data structures, such
as the half-edge or the quad-edge, are efficient for our purposes [5, 25, 36]. As we will show in our algorithms, efficient computation can be performed through an edge-based data structure, since it allows adjacency operations to be computed in constant time. Hereafter, we will assume that the complexes are all modeled by means of an edge-based data structure.

3 Discrete Morse Theory

The discrete Morse theory, formulated by Forman [21], is an adaptation to discrete structures of the Morse theory [37], which relates the critical points of a function $f$ to the topology of the domain.

A discrete Morse function on a cell complex $K$ is a function $f: K \rightarrow \mathbb{R}$ such that, for every $\sigma \in K$, $f$ takes a value less than or equal to $f(\sigma)$ in at most one coface of $\sigma$ and takes a value greater than or equal to $f(\sigma)$ in at most one face of $\sigma$. Figure 2 shows an example of a discrete Morse function.

Given a discrete Morse function $f$, a discrete vector field $V$ is a collection of pairs of cells $(\alpha^p, \beta^{p+1})$ in $K$ defined whenever $\alpha^p < \beta^{p+1}$ and $f(\beta^{p+1}) \leq f(\alpha^p)$. A pair $(\alpha^p, \beta^{p+1})$ can be thought of as a discrete tangent vector leaving $\alpha$ and oriented towards $\beta$. Pictorially, the vector is represented by an arrow from $\alpha$ to $\beta$. Figure 3 shows the vector field of the Morse function presented in Figure 2.

A cell $\sigma^p \in K$ is critical of index $p$ if all of its cofaces take strictly greater values in $f$ and all of its faces are strictly lower in $f$ [21, 22, 44]. In a two-dimensional cell complex, minima are 0-cells, saddles are 1-cells and maxima are 2-cells. Figure 3 shows examples of critical cells of a discrete Morse function. Every cell belongs to a pair in the vector field, except for critical cells.

Another important concept related to vector fields is that of flow. The analogue of a streamline in a continuous function is called a $V$-path, which is a sequence of cells $\alpha^p_1, \alpha^p_1, \alpha^p_2, \alpha^p_2, \alpha^p_3, \ldots, \alpha^p_{r-1}, \alpha^p_r$, such that $(\alpha^p_i, \alpha^p_{i+1}) \in V$, $\alpha^p_i \prec \alpha^p_{i+1}$, and $\alpha^p_i \neq \alpha^p_{i+1}$, for all $i = 1, \ldots, r - 1$. A $V$-path is a non-trivial closed $V$-path if $\alpha^p_r = \alpha^p_1$ for $r \geq 2$. 

![Figure 1](image1.png) Digital image modeled as a cell complex $K$. Pixels are 0-cells of the complex which also have cells of dimension 1 and 2 to explicitly define the topology of the image.

![Figure 2](image2.png) Example of a discrete Morse function. The values of the function are placed over each cell.
Forman [21] presented a discrete vector field $V$ without non-trivial closed paths as the discrete analogue of the continuous gradient vector field. Notice that $V$-paths alternate between cells of dimension $p$ and $p + 1$, therefore, we could explicitly refer to a type of path as a $(p, p + 1)$-path, for instance, a $(0, 1)$-path alternates between cells of dimension 0 and 1. Figure 3 shows a $V$-path in blue. The vector field $V$ of the input complex $K$ is computed using the algorithm presented by Robins et al. [44], which returns the pairs of cells in $V$ and the critical cells.

We process two complexes. The first one is the complex $K$ of the input image with its discrete vector field $V$ derived from a discrete Morse function. Pairs of cells can be stored as an attribute of cells in $K$. The second complex is the discrete Morse complex $M$ that we will compute. Both complexes are modeled through an edge-based data structure, such as the half-edge or the quad-edge.

In order to avoid confusion when referring to cells in $K$ or cells in $M$, from now on, we will use Greek letters for cells in $K$ and Latin letters for cells in $M$. Particularly, we will use $v^p$ or $u^p$ to denote a 0-cell in $M$ and $e$ to denote a 1-cell in $M$. A cell $v^p$ will correspond to a $p$-cell in $K$ and $e$ will denote a boundary relation between cells in $K$. Figure 4 shows two complexes $K$ and $M$ and the corresponding cells. The 0-cells in $M$ can be regular or critical according to the cells they correspond to in $K$. The 1-cells in $M$ explicitly model the boundary relation between two cells in $K$. The symbols $\oplus$, $\odot$ and $\circ$ will henceforth denote critical 0-cells in $M$ related to 0-, 1- and 2-cells, which are also critical in $K$. If not critical, the 0-cell will be drawn with the symbol $\circ$.

Before presenting the algorithms, we introduce alternative definitions of $V$-paths to reflect the explicitly modeled paths used in the algorithms. In two dimensions, a discrete Morse
complex can have two types of $V$-paths: $(0,1)$-paths and $(1,2)$-paths. We denote such paths as $QV$-paths in the complex $M$ we are computing. Definitions for both cases are presented next.

**Definition 1.** Given a $(0,1)$-path $\alpha_0^0, \alpha_1^0, \alpha_2^0, \ldots, \alpha_r^0$ in $V$, a $(0,1)$-path in $M$, leaving $v_0^0$, is a sequence of 0-cells (denoted by $v_i^{(0,1)}$) and 1-cells (denoted by arrows) $v_0^0 \rightarrow v_1^0 \rightarrow v_2^0 \rightarrow \ldots \rightarrow v_{r-1}^0 \rightarrow v_r^0$ where $v_i^0 \rightarrow v_j^0$ are the cells related to a pair $(\alpha_i^0, \alpha_j^0)$ in $V$ and $v_0^0$ is related to a critical cell $\alpha^1 \succ \alpha_i^0$, $\alpha^1 \neq \alpha_j^0$.

Notice from this definition that the $(0,1)$-path leaves a 1-cell and arrives to a 0-cell, making explicit the connection of critical endpoints and following the intuitive notion of a flow going downwards from a saddle to a minimum cell. In a similar way, a $(1,2)$-path definition is presented, but now the path leaves a 2-cell towards a 1-cell.

**Definition 2.** Given a $(1,2)$-path $\alpha_1^1, \alpha_2^1, \alpha_2^0, \ldots, \alpha_r^1$ in $V$, a $(1,2)$-path in $M$, arriving to $v_i^1$, is a sequence of 0-cells (denoted by $v_i^{(1,2)}$) and 1-cells (denoted by arrows) $v_0^1 \rightarrow v_1^1 \rightarrow v_2^1 \rightarrow \ldots \rightarrow v_{r-1}^1 \rightarrow v_r^1$ where $v_i^1 \rightarrow v_j^1$ are the cells related to a pair $(\alpha_i^1, \alpha_j^1)$ in $V$ and $v_0^1$ is related to the cell $\alpha^2 \succ \alpha_i^1$, $\alpha^2 \neq \alpha_j^1$.

### 5 Extraction of $QV$-Paths

The algorithm for extracting the $QV$-paths connecting critical cells is based on searching the paths out of 1-cells. This approach enables efficient computation of the paths in the 2D setting when compared to a breadth-first search [44], since the algorithms consider only paths that may connect critical cells and avoid multiple traversals of cells along sub-paths common to more than one path.

#### 5.1 Initializing $QV$-Paths from a Saddle

The initialization of $QV$-paths out of a saddle in $K$ is performed by identifying all of its possible $V$-paths and including them in the resulting complex $M$. Up to four $V$-paths can be expected to go out of a 1-cell since a 1-cell has two faces and at most two cofaces [4]. Given a critical 1-cell $\alpha^1$ of $K$, we start all $QV$-paths arriving and leaving a 0-cell $v_0^1 \in M$ related to $\alpha^1$ (see Figure 5). In our examples, we use cell complexes of triangles. Regardless, the concepts discussed are not dependent on an image’s representation by triangular or quadrilateral 2-cells.

The 1-cell $\alpha$ has two faces, $\beta^0$ and $\gamma^0$; and also two cofaces, $\sigma^2$ and $\tau^2$ (Figure 5a). We create the 0-cells in $M$ related to each face and coface of $\alpha^1$ in $K$, as well as the four 1-cells that model the boundary relations. This can be observed in Figures 5b to 5e. At the end, the ring of 1-cells out of $v_0$ (a cyclical ordering of the 1-cells leaving a 0-cell $v$) needs to be adjusted to maintain consistency of the edge-based data structure. All the operations take constant time when using an edge-based data structure. For a better understanding of the operations to create and connect the cells, one could refer to [25]. The connection of cells, for example, can be easily performed through the Splice operation described by Guibas and Stolfi [25]. As part of the creation of 0-cells their attributes are properly set, i.e., index $p$ of $v^p$ and indicator of a cell’s critical status.

It is worth noticing how the resulting paths agree with Definitions 1 and 2. Let $v_0$ be $v_0^1$ in Definition 1, two $QV$-paths of the form $v_0^0 \rightarrow v_1^0$ are now in the complex $M$, with $v_0^0$ being $v_3$ in one path and $v_2$ in another. Both paths are related to a trivial $V$-path of the form $\alpha_0^0$ in $V$. Similarly, we have two paths of the form $v_0^0 \rightarrow v_1^1$. However, as explained as part of...
the discussion of the path extraction algorithm, it is more convenient to think of (1, 2)-paths
as being computed in a backwards fashion. In such a manner, one could think of \( v_\alpha \) to be \( v^1 \)
and \( v_\sigma \) to be \( v^2 \) in Definition 2. Therefore, two paths of the form \( v^2_{r-1} \to v^1_r \) are
obtained, both related to a trivial V-path of the form \( \alpha^2_r \) in \( V \).

5.2 Expanding QV-paths

After the initialization, we will iteratively expand the QV-paths. From the previous discussion,
we recall (0, 1)-paths are initialized to grow in a forward manner and (1, 2)-paths are to be
grown backwards. Expanding forward a (0, 1)-path means to grow the initial QV-path to
\( v^0_0 \to v^1_0 \to v^2_0 \) (related to a path \( \alpha^0_0, \alpha^1_0, \alpha^2_0 \) in \( V \)), according to Definition 1. Likewise,
expanding backwards the initial (1, 2)-path means to grow the QV-path to \( v^2_{r-2} \to v^1_{r-1} \to \)
\( v^2_{r-1} \to v^3_r \) (related to a path \( \alpha^2_{r-1}, \alpha^1_{r-1}, \alpha^2_r \) in \( V \)), according to Definition 2. Therefore,
the expansion of the QV-paths is basically obtained by considering a pair \((\alpha^0, \alpha^1)\) or a pair
\((\alpha^1, \alpha^2)\), in the input vector field \( V \), as well as a face or coface of \( \alpha^1 \) that is not paired with
it.

**Definition 3.** Given a (0, 1)-path \( v^0_0 \to v^1_0 \to \ldots \to v^1_{k-2} \to v^0_{k-1} \) and its related V-path
\( \alpha^0_0, \alpha^1_0, \ldots, \alpha^1_{k-2}, \alpha^0_{k-1}, \alpha^0_k \), the expanding triple of the path is the set of cells \( \{\alpha^0_{k-1}, \alpha^1_{k-1}, \alpha^0_k\} \) such
that \( \{\alpha^0_{k-1}, \alpha^1_{k-1}\} \in V \) and \( \alpha^1_{k-1} > \alpha^0_k \), with \( \alpha^0_{k-1} \neq \alpha^1_k \), that allows growing the QV-path to
\( v^0_0 \to v^1_0 \to \ldots \to v^1_{k-2} \to v^0_{k-1} \to v^0_k \) with related V-path \( \alpha^0_0, \alpha^1_0, \ldots, \alpha^1_{k-2}, \alpha^0_{k-1}, \alpha^1_{k-1}, \alpha^0_k \).

Consider the (0, 1)-path \( v_\alpha \to v_\beta \) in Figure 5b. The cells in the expanding triple are
shown in Figure 6a following the V-path out of \( \beta_0^0 \), namely, \( \beta_0^0, \beta^1 \) and \( \beta^0_{d} \). The endpoints
of an expanding triple are indexed with an “o” or a “d” to make it explicit where the flow
enters the path and where it goes to.

**Definition 4.** Given a (1, 2)-path \( v^2_{k+1} \to v^1_{k+2} \to \ldots \to v^1_{r-1} \to v^1_r \), its related V-path
\( \alpha^1_{k+2}, \alpha^0_{k+2}, \ldots, \alpha^1_{r-1}, \alpha^0_r \) and \( \alpha^1_{k+1} \) related to \( v^2_{k+1} \), the expanding triple of the path is the set
of cells \( \{\alpha^1_{k+1}, \alpha^0_{k+1}, \alpha^2_{k+1}\} \) such that \( \{\alpha^1_{k+1}, \alpha^0_{k+1}\} \in V \) and \( \alpha^1_{k+1} > \alpha^2_{k+1} \), with \( \alpha^0_{k+1} \neq \alpha^2_{k+1} \), which
allows growing the QV-path to (0, 1)-path \( v^0_k \to v^1_{k+1} \to v^2_{k+1} \to v^1_{k+2} \to \ldots \to v^1_{r-1} \to v^1_r \)
related to a V-path \( \alpha^1_{k+1}, \alpha^0_{k+1}, \alpha^1_{k+2}, \alpha^2_{k+2}, \ldots, \alpha^1_{r-1}, \alpha^1_r \).

The expanding triple for the (1, 2)-path \( v_\sigma \to v_\alpha \) in Figure 5d is composed of the cells \( \sigma^2_0, \sigma^1 \) and \( \sigma^2_d \), as shown in Figure 6b.

A QV-path may not have an expanding triple, however, if it does, then it is unique.
Proposition 5. The expanding triple of a (0,1)-path in $M$ is unique.

Proof. Given $v^1_0 \rightarrow v^0_1 \rightarrow \ldots \rightarrow v^1_{k-2} \rightarrow v^0_{k-1}$ related to $\alpha^0_1, \alpha^1_1, \ldots, \alpha^1_{k-2}, \alpha^0_{k-1}$ is a (0,1)-path that has an expansion, suppose the expanding triple is not unique. Let us then call two of the expansions $\alpha^1_1, \alpha^1_2, \ldots, \alpha^1_{k-2}, \beta^1_{k-1}, \alpha^0_1, \alpha^1_1, \ldots, \alpha^1_{k-2}, \beta^0_{k-1}, \gamma^1_{k-1}, \gamma^0_k$. If $\beta^1_{k-1} \neq \gamma^1_{k-1}$, it means there are pairs $(\alpha^1_1, \beta^1_{k-1})$ and $(\alpha^0_1, \gamma^1_{k-1})$ in $V$. However, by definition of a vector field, any cell is paired once in $V$. Consequently, $\alpha^1_{k-1} = \beta^1_{k-1} = \gamma^1_{k-1}$. The paths can still be different if $\beta^0_{k} \neq \gamma^0_k$. In such a case, the 1-cell $\alpha^1_{k-1}$ should have three faces: $\alpha^0_{k-1}$, $\beta^0_k$ and $\gamma^0_k$. That is not possible by definition.

Proposition 6. The expanding triple of a (1,2)-path in $M$ is unique.

Proof. Given $v^2_{k+1} \rightarrow v^1_{k+2} \rightarrow \ldots \rightarrow v^2_{r-1} \rightarrow v^1_r$ related to $\alpha^1_{k+2}, \alpha^2_{k+2}, \ldots, \alpha^2_{r-1}, \alpha^1_r$ is a (1,2)-path that has an expansion, suppose the expanding triple is not unique. Let us call two of these expansions $\beta^2_{k+1}, \beta^2_{k+2}, \alpha^1_{k+2}, \ldots, \alpha^2_r, \alpha^1_r$ and $\gamma^1_{k+1}, \gamma^2_{k+1}, \alpha^1_{k+2}, \alpha^2_{k+2}, \ldots, \alpha^2_r, \alpha^1_r$. If $\beta^2_{k+1} \neq \gamma^2_{k+1}$ then the 1-cell $\alpha^1_{k+2}$ should have three cofaces: $\alpha^2_{k+2}, \beta^2_{k+1}$ and $\gamma^2_{k+1}$. That is not possible. Since $\alpha^2_{k+1} = \beta^2_{k+1} = \gamma^2_{k+1}$, the paths can still differ if the pairs $(\beta^2_{k+1}, \beta^2_{k+1})$ and $(\gamma^1_{k+1}, \gamma^2_{k+1})$ are in $V$ and $\beta^2_{k+1} \neq \gamma^2_{k+1}$. Again, by definition of $V$, any cell is paired once, and $\beta^2_{k+1} = \gamma^2_{k+1}$ since $\beta^2_{k+1} = \gamma^2_{k+1}$.

Given a cell $\alpha^0$ in the input vector field $V$, we are able to retrieve the 1-cell paired with $\alpha^0$ in constant time. The second face of a 1-cell can also be computed in constant time with an edge-based data structure. Therefore, retrieving the expanding triple is an $O(1)$ operation.

Retrieving the expanding triple in a (1,2)-path is similar and also efficient.

Considering again the paths obtained after the initialization steps, these were of the form $v^3_0 \rightarrow v^1_1$ and $v^3_{r-2} \rightarrow v^1_r$. Now, given the paths in Figures 5b and 5d, the vertices $v_1$ and $v_r$ and the expanding triples of Figures 6a and 6b, we are able to form $v^3_0 \rightarrow v^0_1 \rightarrow v^1_1 \rightarrow v^2_2$ and $v^3_{r-2} \rightarrow v^0_{r-1} \rightarrow v^1_{r-1} \rightarrow v^2_r$, as shown in Figures 7a and 7b.

Figure 6 Expanding triples for a (0,1)-paths and a (1,2)-path.

Figure 7 One step expanded $QV$-paths obtained with the expanding triples of the initial (0,1)- and (1,2)-paths shown in Figures 5b and 5d.
5.3 Stop Conditions

The paths must be expanded until a termination condition is met. An expanding triple is not found for a \((0, 1)\)-path when the path reaches a critical cell. A \((1, 2)\)-path does not have an expanding triple either when it reaches a critical cell or the boundary of the cell complex.

Proposition 7. A \((0, 1)\)-path \(v^1_0 \rightarrow v^1_1 \rightarrow v^1_2 \rightarrow \ldots \rightarrow v^1_{k-1} \rightarrow v^1_k\) has no expanding triple if the 0-cell, \(\alpha^0 \in K\), related to \(v^0_k\) is critical.

Proof. If \(\alpha^0\) is not critical, then, by definition, it is paired in \(V\). A pair \((\alpha^0, \beta^1)\) in a \(V\)-path is always followed by a 0-cell \(\alpha^0_{i+1}\) which is a face of \(\beta^1\) other than \(\alpha^0\) which must also exist since every 1-cell has two faces. If \(\alpha^0\) is critical, then it is not paired in the vector field and, therefore, no expanding triple can be found.

Proposition 8. A \((1, 2)\)-path \(v^2_k \rightarrow v^1_{k+1} \rightarrow v^2_{k+1} \rightarrow v^1_{k+2} \rightarrow \ldots \rightarrow v^2_r \rightarrow v^1_r\) has no expanding triple if one of two conditions is true:
1. the cell \(\alpha^2_k\), related to \(v^2_k\), is critical;
2. the cell \(\alpha^2_k\) is paired with a 1-cell, \(\alpha^1 \in V\), which has no coface in \(K\) other than \(\alpha^2_k\).

Proof. Condition 1. If \(\alpha^2\) is critical, then it is not paired in \(V\) and, therefore, no expanding triple out \(\alpha^2\) can be found. Condition 2. If \(\alpha^2\) is not critical, then it is paired in \(V\), with a 1-cell \(\alpha^1\). The 1-cell is in \(K\) since all the faces of a cell must be in \(K\). Given the pair cell \(\alpha^1\), if its two cofaces are in \(K\), then an expanding triple for the path is found, since there is a coface \(\alpha^2_{k-1} \neq \alpha^2_k\). Otherwise, if there is no other coface other than \(\alpha^2_k\), then no expanding triple can be found.

From these conditions, it is possible to expand a whole \(QV\)-path. However, some additional stop conditions will be introduced. These are based on the fact that \(V\)-paths can merge and branch.

Consider a particular 0-cell \(\alpha^0\) in a complex \(K\) and suppose it is paired with a 1-cell in the vector field \(V\) of the complex. Let also \(\alpha^0\) be a face of \(k\) 1-cells in the complex. Except for the 1-cell paired with \(\alpha^0\), each one of the other 1-cells either is paired or is a critical 1-cell. If \(l \leq k\) of these 1-cells are paired with a 0-cell, then there are \(l\) \(V\)-paths containing the cell \(\alpha^0\) and, therefore, at least \(l\) paths merge into \(\alpha^0\). Figure 8a shows an example of a cell at which three \(V\)-paths merge.

\[\text{Figure 8} \text{ Example of an 0-cell shared by three } V\text{-paths and a 2-cell shared by two } V\text{-paths.}\]

Now let \(\alpha^2\) be a 2-cell in \(K\) paired in \(V\) with one of its 1-cell faces. If the \(\alpha^2\) has \(k\) 1-cell faces, then these cells may be paired with a 0-cell, paired with a 2-cell or be a critical cell. If \(l \leq k\) of them are paired with 2-cells, the \(V\)-path containing \(\alpha^2\) will have \(l\) \(V\)-paths out of it, in other words, the path branches into \(l\) \(V\)-paths. Figure 8b show a \(V\)-path which branches into two \(V\)-paths.

Both merging and branching cases are considered stop cases when a \(QV\)-path reaches a cell in a path already processed. In Figure 9a, the path ending at the 0-cell \(v\) is currently
being processed. When the path is expanded from \( v \), it merges with a second path that has
been already computed (Figure 9b). From that point on, the expansion of both paths is
exactly the same (a consequence of a unique expanding triple). It is also simple to deal with
branching cases. The two \((1, 2)\)-paths shown in Figure 9c become one in Figure 9d. Notice
that, since \((1, 2)\)-paths are traversed in a backward fashion, the branching case of \((1, 2)\)-paths
is algorithmically similar to the merging case of \((0, 1)\)-paths and we are able to deal with
both cases in a similar way.

\[
(a) \quad (b) \quad (c) \quad (d)
\]

**Figure 9** Examples of \(QV\)-paths that merge and branch.

A path can be fully extracted by iteratively computing the expanding triples and testing
the stop conditions. All these computations can be performed in constant time, therefore,
extracting a path takes linear time on the size of the \(V\)-path.

The paths of a discrete Morse complex are exactly the \(QV\)-paths with critical cells as
endpoints. However, there may be \((1, 2)\)-paths that do not end at a critical 2-cell. That
follows from the second condition of Proposition 8. In order to obtain only the paths of the
Morse complex, it suffices removing a \((1, 2)\)-path if one of its endpoints is not a critical cell
(see Figure 10). This process can be easily performed with the data structure in linear time
on the size of the path. Alternatively, one could add a dummy critical cell when the second
condition of Property 8 is satisfied.

**Figure 10** The rightmost path does not end in a critical cell. That could happen in case the path
reaches the image boundary.

It is now possible to extract all \(QV\)-paths in a discrete Morse complex. It suffices to
go through all the saddles of the input complex extracting all of its paths. If the complex
has $N$ saddles, then $4N$ paths will be processed. If the size of the $i$-th path is $L_i$, then the algorithm will require $O\left(\sum_{i=1}^{4N} L_i\right)$ time.

6 Disentangled Complex

Since the extracted paths can merge or branch, we call the complex described in the previous section an entangled complex. For some applications, it is important that certain cases, such as merging and branching, do not occur. In the following, we will show how to separate merged and branched paths and keep them independent of each other.

$QV$-paths merge or branch at a given 0-cell along their extent. We will define that 0-cell $v \in M$ a knot if it is not critical and its degree (number of paths arriving and leaving) is greater than two. Figure 9b shows a knot $v$ of degree three. A knot of branching paths is shown in Figure 9d.

If a knot is present, it means that $l$ paths have a common subpath that occurs after a merging case or before a branching case. The idea to disentangle all knots can be thought of as a method that pushes forward (backward) a knot, along the common subpath of merging (branching) paths, until a critical endpoint is reached.

The basic operation works locally at a knot separating the paths one step forward (backward). For such, we need a knot $v_1$ and the 0-cell $v_2$ towards which $v_1$ is pushed. The edge $e_1$ between $v_1$ and $v_2$ is the common subpath to be separated into the $k$ paths entangled in $v_1$. For the case shown in Figure 11a, two paths merge into $v_1$, the knot is pushed forward to $v_2$ along the edge $e_1$. Using the edge-based data structure, it suffices to disconnect each one of the $k$ paths from the ring of 1-cells of $v_1$. A new 0-cell $v$ is then created for each path in order to substitute $v_1$ in that path. A 1-cell $e$ is created to connect $v$ to $v_2$. In such a manner, all paths are disentangled by one step. Figure 11b shows the result for the example path. A new application of the process will disentangle the paths one step further, resulting in the paths shown in Figure 11c. The operation involves traversing the rings of 1-cells of $v_1$ and applying constant time computations. The operation takes constant time on average, since the average degree of a planar graph is strictly less than six [28].

![Figure 11](image-url) Example of untangling knots of a merged path. In (a), the common subpath between $v_1$ and $v_2$ is split so that the two paths arriving at $v_1$ are then separated in (b). The process is applied once again to obtain the separated paths in (c), and, by iterating along the whole common subpaths, fully separated paths are obtained (d).
It suffices to continue pushing forward (backward) the knot vertices to fully separate
paths. Similar to the approach to extracting the QV-paths, we proceed from saddles in M
and follow all of its paths disentangling the knots until another critical cell is found. The
resulting paths are then disjoint. For visualization purposes and to maintain the planarity
of the complexes, the geometrical position of the paths can be perturbed infinitesimally, so
that they are positioned parallel to each other. Figure 11d shows the result of disentangling
the knots. If the number of saddles is N and each saddle can have up to four paths, the
algorithm again will require $O(\sum_{i=1}^{4N} L_i)$ time.

7 Simplified Morse Complex

The complex obtained by the method discussed above can be used for simple visualization,
but it is not efficient for many practical tasks, such as computing Betti numbers or hierarchies
of the complex. For such tasks, one would intend to obtain the critical cells connected by
QV-paths in constant time, without traversing all paths.

We present a simplified version of our discrete Morse complex such that the 1-cells connect
critical cells. The path between the critical cells are stored in a list as an attribute of the
1-cells. With such a complex, we obtain the objective of a model that is easy to manipulate
through local operations, but that can also be used for visualization purposes.

We consider subpaths called simplification kernels, depicted in Figures 12a and 12b. The
kernel is formed by the set of cells \{e, v_1, e_1, v_2, e_2, v_3\}. We chose such a kernel to maintain
the property that a path alternates between cells of different dimensions. If the destination
of e is not a critical cell, then we can argue that a kernel can be found. Consider a (0, 1)-path
$u_0^0 \rightarrow u_1^0 \rightarrow u_2^0 \rightarrow \ldots \rightarrow u_{i-1}^0 \rightarrow u_i^0$, as in Definition 1. The saddle in the origin of e is
$u_0^0$ and the destination is $u_i^0$ in the path. By definition, the subpath $u_0^0 \rightarrow u_1^0 \rightarrow u_2^0$ should be
present in the path. These are exactly the cells needed to form the kernel. The same ideas
can be applied to a (1, 2)-path.

The 1-cell e has as origin a saddle in M. The cells v_1 and v_2, as well as e_1 and e_2, are
removed from the complex and the 1-cell e is connected to $v_3^0$. The result of the simplification
of the complex from Figure 12a is shown in Figure 12b.

Finding kernels and simplifying them must be repeated until the 1-cell e finally connects
critical 0-cells. The endpoints of e shown in Figure 12b still are not both critical. Therefore,
a new kernel is found and simplified. The result is the edge e shown in Figure 12c that
connects critical cells and the process stops. The time complexity is linear in the size of the
path. Again, the process must be repeated for each path, implying that a fully simplified
complex is obtained in $O(\sum_{i=1}^{4N} L_i)$ time. The resulting complex for our example is shown in
Figure 12d.

8 Persistence and Topological Hierarchies

We present applications of the discrete Morse complex for analysis of image functions and
important topological computations, such as persistence of critical cells [13, 23, 38, 39] and
simplification of a topological structure to produce a hierarchical representation [35, 46, 49].

We use elevation functions comprising one synthetic image and four real-world terrain
height field data [41, 48]. The synthetic image was computed as $h(x, y) = \sin x + \sin y$
Figure 12 Simplification of subpaths and resulting discrete Morse complex. In (a) a kernel is used to exchange a subpath by a unique 1-cell (b). The process is repeated in (b) and stops since critical cells are now directly connected (c). In (d) the discrete Morse complex connects only critical cells.

multiplied by an exponential $g(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$, with $x$ and $y$ in the interval $[-40, 40]$ and $\sigma = 6$. The image is shown in Figure 13a and its resulting discrete Morse complex in presented in Figure 13b. The information on the image set and the number of critical cells of each type in the resulting discrete Morse complexes are given in Table 1.

Figure 13 Discrete Morse complex (b) of the synthetic sinusoidal image (a). The colors in (b) represent the persistence of critical points, ranging from low persistence (blue) to high persistence (red), with intermediate values in yellow and green.

The complexes were employed to compute the persistence of critical points and hierarchical decompositions of the complexes through cancellation of low persistent topological features. These topological features are 0-cycles and 1-cycles, which, in a two-dimensional space, are related to connected components and holes in the level sets of the images. The number of cycles in the level sets, along with persistence, was computed with the algorithm described by Zomorodian [54]. The algorithm requires a total ordering of the critical points in the Morse complex, which was obtained by assigning the grayscale value of each critical cell to
Table 1 Images used in the experiments. The second column lists the resolution (in pixels) of the images, third to fifth columns report the number of cells in each dimension in the respective cell complexes, whereas sixth to eighth columns report the number of critical points in the Morse complex.

<table>
<thead>
<tr>
<th>Image</th>
<th>Resolution (pixels)</th>
<th>Image Complex</th>
<th>Morse Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>256 × 256</td>
<td>65536</td>
<td>130560</td>
</tr>
<tr>
<td>Crater Lake</td>
<td>336 × 459</td>
<td>154224</td>
<td>307653</td>
</tr>
<tr>
<td>Cumberland</td>
<td>1201 × 1201</td>
<td>1442401</td>
<td>2882400</td>
</tr>
<tr>
<td>Death Valley</td>
<td>1201 × 1201</td>
<td>1442401</td>
<td>2882400</td>
</tr>
<tr>
<td>Mars</td>
<td>936 × 949</td>
<td>888264</td>
<td>1774643</td>
</tr>
</tbody>
</table>

The corresponding point in the Morse complex. For details on the subject, one should refer to specialized literature [29, 53, 54].

The image in Figure 13b depicts in different colors the persistence of each critical point in the discrete Morse complex of the sinusoidal image (Figure 13a). The color scale varies from blue to red, representing low persistence and high persistence. Intermediate values are shown in yellow and green intensities. The critical points of higher persistence are positioned in the center of the image and the persistence decays from these points to points in the boundaries of the image. This effect is due to the exponential function added to the sinusoidal function.

The 0-cycles and 1-cycles of the image are summarized in the graphs of Figure 14. The cycles are sorted by persistence and it can be noticed that the persistence increases following an exponential behavior, as expected.

Figure 14 Graphs that illustrate the respective persistence for each 0- and 1-cycle in the sinusoidal image. The sorted persistence of the cycles grows exponentially, except for the last 0-cycle, which has infinite persistence. The horizontal line presents the median for reference.

The graphs for number of cycles versus persistence for the real images are shown in Figure 15. The persistence axis is shown in logarithmic scale, so that the values can be better analyzed. These graphs show a common characteristic: a small amount of cycles with high persistence and a great amount of cycles with very low persistence. These very low persistent cycles may be due to topological noise of the data. An important task is then the removal of such topological noise, as explored in [9] and [18]. Cleaning the topological noise may be interesting for better understanding the function or the phenomenon being studied.

The data structure of our complex allows easy and fast manipulation of the complex for simplifications. We implemented the operation as described in [18].

It can be noticed that noisy topological features are mostly concentrated in low persistent points. By suppressing low persistent features, the discrete Morse complex becomes cleaner. This effect is reflected in the hierarchy of simplified discrete Morse complexes, shown in
Figure 15 Graphs that illustrate the respective persistence for each 0- and 1-cycle of real elevation terrain images. The sorted persistence shows a small amount of cycles with high persistence and a great amount of cycles with very low persistence. The horizontal line presents the median for reference.

Figure 16. In the initial complexes, the island in the middle of the lake does not stand out, since there are many low persistence features. As the simplification progresses, topological features of low prominence are removed and important topological features are preserved.

We have also computed the hierarchy for a face image, as shown in Figure 17. Initially, there is a high concentration of feature areas with non-relevant details. The details are removed as the levels in hierarchy are traversed and features located at the eyes, nose and mouth stand out. This example shows how the simplification of the complexes can be used to capture significant topological information of images and suggests that it can be useful for image classification. The image was obtained from the AT&T Laboratories Cambridge [2].

Finally, we discuss computational time and memory usage (Table 2) for each step in
the construction of the discrete Morse complex. The algorithms were implemented\(^1\) using
the C programming language and the experiments were performed on a computer with an
Intel i7 core (1.8GHz and 8GB RAM memory) with Ubuntu 18.04. The quad-edge [25] data
structure was used for the complexes. In addition, for the images already covered in the
previous discussion, we provide information for the Berkeley Segmentation Data Set and
Benchmarks 500 (BSDS500) [1]. The dataset consists of 500 images of two different sizes,
i.e., 152 images of 312×481 pixels and 348 images of 481×312 pixels.

Table 2 shows the mean computational time and memory usage for the BSD dataset. Even
though our implementation used the quad-edge data structure for the image complex and for
the discrete vector field, it would be possible to reduce memory cost with memory-efficient
data structures [20]. In addition, the image complex and the vector field are no longer
necessary after computing the discrete Morse complex. The Topology Toolkit (TTK) [7, 47]
supports Morse-Smale computation. In Table 3, we provide execution time and memory usage
using TTK (version 0.9.8) filters in the ParaView software (version 5.6.1) [3]. The software
output values are reported. Our implementation of our algorithms is efficient, as can be seen
in columns 4 to 6 in Table 2. The computation of the vector field, the input to our algorithms,
needs to be improved, as has not yet been optimized for images of large dimensions. This
efficiency improvement is possible, since TTK also computes the vector field at the reported
times. TTK triangulates complexes as well, instead of using a quadrilateral grid, which could
potentially increase computational time. It is our plan to improve efficiency related to path
computations and fine-tune our implementation for better run time performance.

\(^1\) The code will be publicly available after paper acceptance.
# Table 2

Computational time and memory usage for each step of the algorithm. Time and memory is reported for each step: image cell complex (Img), computation of the vector field (VF), computation of the paths (Paths), disentangling the paths (Dis.) and contracting the paths (Con.). The contract step does not increase the memory, therefore we report the complex storage size after disentangling and contracting steps as a unique value.

<table>
<thead>
<tr>
<th>Image</th>
<th>Time (seconds)</th>
<th>Memory (megabytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Img</td>
<td>VF</td>
</tr>
<tr>
<td>BSD</td>
<td>0.067</td>
<td>1.542</td>
</tr>
<tr>
<td>Sine</td>
<td>0.027</td>
<td>0.569</td>
</tr>
<tr>
<td>Crater Lake</td>
<td>0.067</td>
<td>1.493</td>
</tr>
<tr>
<td>Cumberland</td>
<td>0.611</td>
<td>15.743</td>
</tr>
<tr>
<td>Death Valley</td>
<td>0.618</td>
<td>15.948</td>
</tr>
<tr>
<td>Mars</td>
<td>0.383</td>
<td>9.236</td>
</tr>
</tbody>
</table>

# Table 3

Computational time (four threads) and memory usage from TTK using ParaView.

<table>
<thead>
<tr>
<th>Image</th>
<th>Time (seconds)</th>
<th>Memory (megabytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>0.158</td>
<td>1.48</td>
</tr>
<tr>
<td>Crater Lake</td>
<td>0.458</td>
<td>6.38</td>
</tr>
<tr>
<td>Cumberland</td>
<td>4.841</td>
<td>148.10</td>
</tr>
<tr>
<td>Death Valley</td>
<td>5.479</td>
<td>132.16</td>
</tr>
<tr>
<td>Mars</td>
<td>2.733</td>
<td>-</td>
</tr>
</tbody>
</table>

## 9 Conclusions

We presented and analyzed algorithms for computing discrete Morse complexes of two-dimensional images. We modeled complexes with an efficient edge-based data structure, e.g., a winged-edge or a quad-edge structure, to guarantee constant-time local operations applied to the complexes.

The presented algorithms work in a way that only paths in a vector field that may lead to paths in a Morse complex are processed. The algorithms show how to deal with merging and branching in order to produce a consistent representation of the model. The proposed simplified complex is easy to manipulate and useful to compute local topological operations. Furthermore, the model used for the complex is suitable for visualization tasks.

Theoretical and experimental results were presented to show the effectiveness of our algorithms. The proposed complex is suitable for the computation of homological persistence numbers [53], removal of topological noise [51] and acquisition of hierarchies of the discrete Morse complex [18]. Computational time results show that our algorithm has a performance comparable to the Topological Toolkit (TTK). However, memory storage still needs to be improved in future work.

## References


