Chapter 2
Morphology Computation Algorithms: Generalities

In this chapter, we consider features common to all algorithms for morphology computation. Such algorithms can be classified based on different criteria, that are in general mutually orthogonal (see Sect. 2.1):

1. *dimension of the input scalar field:*
   some methods are dimension-specific (in practice, either for 2D or for 3D scalar fields), while other methods are dimension-independent.
2. *input format:*
   methods may operate on simplicial models or on regular models; some methods also require specific properties for the input scalar field.
3. *output information:*
   some methods compute (a discrete approximation of) the ascending or descending Morse complex; other methods compute (a discrete approximation of) the Morse-Smale complex.
4. *format of the output information:*
   some methods provide a complete combinatorial description of the output complex; many methods simply provide a classification of the 0-cells (vertices), or of the d-cells (triangles or pixels for $d = 2$, tetrahedra or voxels for $d = 3$).
5. *algorithmic approach:*
   different approaches come from different reference theories: Banchoff's piecewise linear Morse theory, watershed transform in the discrete case, Forman's discrete Morse theory.

The basic component of morphology computation algorithms is the identification of the critical points of the field. Section 2.2 is devoted to a treatment of the techniques for computing critical points. Another general issue with morphology computation algorithms is how to deal with the domain boundary and with regions with same elevation (plateaus). We discuss how to deal with such issues in Sects. 2.3 and 2.4, respectively.

L. Čomić et al., *Morphological Modeling of Terrains and Volume Data*,
SpringerBriefs in Computer Science, DOI 10.1007/978-1-4939-2149-2_2

For more details on classifications of morphology computation algorithms, and on computation of critical points, see [5, 8].

2.1 Classification of Morphology Computation Algorithms

In the following, we examine the various criteria which can be used for classifying morphology computation algorithms.

2.1.1 Input Dimension, Format and Properties

Algorithms may have been designed for 2D scalar fields, 3D scalar fields, or they may be dimension-independent. At the same time, they may have been designed for scalar fields represented through regular grids or through simplicial models. Some methods are even independent of the input format, in the sense that they can be applied to both formats, with little change.

Algorithms working on *regular grids* may assume that field values are given at the d-cells (pixels for $d = 2$, voxels for $d = 3$), or at the 0-cells (vertices) of the grid. In the latter case, interpolating functions are used within higher-dimensional cells. Algorithms for *simplicial models* (triangle meshes in 2D, tetrahedral meshes in 3D) always assume that field values are given at the vertices, and linear interpolants are used within higher-dimensional simplices.

In several algorithms, the computation focuses on cells of a certain dimension and is based on navigation among them through adjacency relations. If the computation focuses on 0-cells (vertices), this corresponds to considering the *primal graph* of the input model (see Sect. 1.1.4). If the computation is focused on d-cells, this corresponds to considering the *dual graph* of the input model.

Algorithms for stepped regular models consider the dual graph. Here, the nodes are pixels in 2D, and the arcs are defined according to some connectivity model (4- or 8-connectivity in 2D, see Sect. 1.2 and Fig. 1.5). Algorithms for regular models with interpolating functions work on the primal graph.

Algorithms for simplicial models work on either the primal or the dual graph, depending on the approach (for example, boundary-based methods use the primal graph, and region-growing methods use the dual graph, see Sect. 2.1.3).

Some methods for simplicial complexes require that the input scalar field is general, i.e., all pairs of adjacent vertices have distinct function values. Intuitively, this means that flat edges are not allowed.

2.1.2 Output Information and Its Format

Algorithms may produce a (descending or ascending) Morse complex, or a Morse-Smale complex. In both cases, the output is a cell complex, and thus it consists of

cells of all dimensions, and of combinatorial relations of adjacency and incidence between pairs of cells. In practice, such output cell complex can be represented in the form of:

- Just the d-cells of the output complex, represented through a classification of the d-cells (triangles or pixels in 2D, tetrahedra or voxels in 3D) of the input scalar field model, where each d-cell is labeled as belonging to a specific d-cell of the Morse or Morse-Smale complex.
- Just the d-cells of the output complex, represented through a classification of the vertices of the input scalar field model, where each vertex is labeled as belonging to a specific d-cell of the Morse or Morse-Smale complex. A more detailed classification may assign each vertex to an i-dimensional cell of the output complex, where $i = 0, \ldots, d$.
- The skeleton (1-skeleton consisting of points and lines in 2D, 1- and 2-skeletons consisting of points, lines and surfaces in 3D) of the Morse or Morse-Smale complex, with their combinatorial relations. Such skeleton(s) define the boundaries of the d-cells of the output complex.
- A complete representation of the Morse or Morse-Smale complex, i.e., its cells of all three dimensions in 2D or four in 3D, and their combinatorial relations. This corresponds to the Morse incidence graph (see Sect. 1.3), plus geometric information about cells.

Most methods that produce Morse complexes are completely symmetric in the ascending and in the descending case: the ascending Morse complex can be computed as the descending Morse complex by considering the field $-f$ having the opposite function values on the same underlying cell complex. Thus, the two Morse complexes are produced in the same format.

Forman-based algorithms produce the ascending and the descending Morse complexes represented in different formats. Descending cells associated with maxima are expressed as collections of d-cells of the primal complex. Ascending cells associated with minima are expressed in terms of d-cells of the dual complex, i.e., they are collections of vertices of the primal complex.

An algorithm has been presented in [7], which constructs the Morse incidence graph (see Sect. 1.3) in 2D and 3D, starting from a classification of triangles and tetrahedra, respectively.

2.1.3 Algorithmic Approach

Based on the approach they use, the various algorithms can be classified as:

- *Boundary-based* methods, which build the lower-dimensional skeletons of the Morse-Smale complex (boundary lines of 2-cells in 2D, and boundary lines and surfaces of 3-cells in 3D).

- *Region-growing* methods, which build the d-cells (called *regions*) of the descending (ascending) Morse complex, by progressively growing each of them, starting from its seed maximum (minimum).
- *Watershed* methods, originally developed for grey-level images, which compute the ascending Morse complex based on the idea of simulating the diffusion of water.
- *Forman-based* methods, which previously define a Forman gradient from the data.

Boundary-based algorithms build the lower-dimensional skeletons of the Morse-Smale complex. The input can be a regular model or a simplicial model of the scalar field. The 1-skeleton is extracted by computing the critical points and then tracing the integral lines (or their approximations) starting from saddle points and converging to minima and maxima. For 3D scalar fields, the 2-skeleton is also built. Boundary-based algorithms can generate the skeleton(s) of the descending (or ascending) Morse complex by tracing only integral lines from saddles to minima (or maxima).

Region-growing algorithms compute an approximation of the descending (ascending) Morse complex by growing the d-cells, called *regions*, corresponding to the maxima (minima), of the scalar field. Such regions are built as collections of d-cells (triangles in 2D, tetrahedra in 3D) of the scalar field model, by classifying triangles, or tetrahedra. An approximation of the Morse-Smale complex can be obtained as the intersection of the descending and ascending Morse complexes. The input is in general a simplicial model of the scalar field. However, some watershed methods, working on regular models, actually operate in a region-growing fashion.

Watershed algorithms typically work on regular grids considered as stepped models. They generate the ascending Morse complex by labeling each d-cell γ (pixel or voxel) with the minimum p, such that γ belongs to the ascending d-cell of p. They can generate the descending Morse complex by considering scalar field $-f$. Watershed methods based on simulated immersion may label some d-cells as belonging to the skeleton of the ascending Morse complex (these are the so-called watershed cells). They are also able to identify the specific element of the skeleton, even if they do not do that. Watershed algorithms can work on simplicial models as well, by considering the primal graph. In this case, they produce a vertex classification. A vertex classification is produced also by the 3D region-growing algorithm in [15].

Forman-based algorithms construct (either directly, or by first defining a Forman function F) a Forman gradient vector field V and its critical cells starting from a scalar field f. The Forman-based algorithm in [14] can also be classified as boundary-based, as it approximates the 1-skeleton of the Morse-Smale complex by gradient lines connecting the critical cells of F.

Boundary-based and region-growing algorithms are reviewed in Chap. 3. *Watershed* and *Forman-based* approaches are reviewed in Chaps. 4 and 5, respectively.

2.2 Detection of Critical Points

Most algorithms perform the identification of critical points in the input scalar field as a pre-processing step. Some algorithms, specifically boundary-based ones, find critical points of all types (maxima, minima, saddles). Not all algorithms for 3D morphology computation distinguish between 1- and 2-saddles (e.g., [13]), and not all algorithms recognize multiple saddles composed of 1- and 2-saddles (e.g., [6, 28]).

Other approaches find critical points of just one type. For instance, region-growing algorithms find just minima when computing the ascending Morse complex, and maxima when computing the descending Morse complex. They do not explicitly extract saddle points.

Some algorithms do not compute critical points in advance, but they recognize them during the computation. This happens, for instance, in the watershed approach by simulated immersion.

In methods which do not detect saddle points, saddles can be found as a post-processing step, by computing the overlay of the descending and ascending complexes. Let us consider the 2D case. If we have a Morse-Smale function, the 1-cells of the descending complex must intersect transversally the 1-cells of the ascending complex, and the saddles are the intersection points. With non-Morse-Smale functions, the intersection of the 1-cells of the descending complex with those of the ascending complex can include an edge or a chain of edges, and, thus, it is not feasible to detect saddles exactly in this way.

2.2.1 Detecting Critical Points in a Simplicial Model

In this section, we analyze how critical points can be computed on a 2D or 3D simplicial model. Since a simplicial model uses linear interpolants, critical points can only occur at the vertices of the underlying triangle or tetrahedral mesh Σ.

First of all, we define the upper/lower link, and the upper/lower star of a vertex v within a simplicial complex Σ. We recall that the link and star have been defined in Sect. 1.1.2.

- The *upper star* of v is the subset of the star of v containing those simplices σ such that all vertices of σ different from v have a higher function value than $f(v)$.
- The *upper link* of v is the subset of the link of v containing those simplices σ such that all vertices of σ have a higher function value than $f(v)$.
- The *lower star* and the *lower link* of v are defined in a completely symmetric way.

We denote the upper link and the lower link as $Lk^+(v)$ and $Lk^-(v)$, respectively.

A first, and widely used, way to identify critical points uses the definition of critical points on piecewise-linear functions due to Banchoff [4,20] and discussed in

Sect. 1.5. Details for the 2D and 3D cases can be found in the works by Edelsbrunner et al. [11] and by Takahashi et al. [28], respectively. The following procedure is used to classify a vertex p:

1. Take the *link* $Lk(p)$ of vertex p, and decompose it into:
   - simplices (vertices and edges in 2D; vertices, edges and triangles in 3D) belonging to the upper link $Lk^+(p)$ of p (see Fig. 2.1a for the 2D case).
   - simplices belonging to the lower link $Lk^-(p)$ of p (see Fig. 2.1b for the 2D case).

   Note that, in the 2D case, other edges, which belong neither to the upper nor to the lower link (see Fig. 2.1c) are the mixed edges defined in Sect. 1.5.

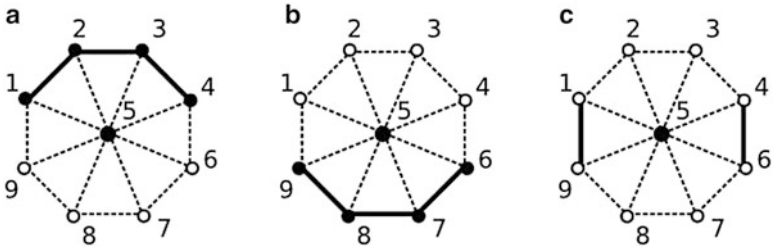

Fig. 2.1 *Bold lines* and *full dots* indicate edges and vertices composing (**a**) the lower link of vertex 5, (**b**) its upper link, and (**c**) the set of its mixed edges of the vertex with field value equal to 5

2. Find the connected components of the upper link of p and of the lower link of p, and count them. Let Δ_+ be the number of components of $Lk^+(p)$, and Δ_- be the number of components of $Lk^-(p)$.
3. Now, classify p:
   - if $\Delta_+ = 0$ (i.e., the upper link is empty), then p is a *maximum*;
   - if $\Delta_- = 0$ (i.e., the lower link is empty), then p is a *minimum*;
   - if $\Delta_+ = \Delta_- = 1$, then p is *regular*;
   - otherwise, p is a *saddle*.

For a 2D complex, in the last case, i.e., when p is a saddle, we always have $\Delta_+ = \Delta_-$, and the multiplicity of the saddle can be computed as $k = (\Delta_+ - 1) = (\Delta_- - 1)$. That is, p is a simple saddle ($k = 1$) if both the upper and the lower link of p have two connected components. It is a multiple saddle otherwise.

For a 3D complex, the type of saddle (i.e., 1-saddle or 2-saddle) can be detected. If $\Delta_+ = 1$ and $\Delta_- = 2$, then p is a simple 1-saddle; if $\Delta_+ = 2$ and $\Delta_- = 1$, then p is a simple 2-saddle. In order to identify multiple saddles, Takahashi et al. [28] count the multiplicity of multiple saddles as follows. If $\Delta_+ = k + 1$, then p is a 1-saddle with multiplicity k. If $\Delta_- = k + 1$, then p is a 2-saddle with multiplicity k.

In 2D, the time complexity of vertex classification, for a triangulation Σ with m vertices, is in $O(m)$. The link of a vertex is a radially sorted sequence of edges

and vertices. The connected components of the upper and lower links are found by scanning the radially sorted list and noting the sign changes. Summing up over all m vertices examined, the total number of neighbor pairs is equal to twice the number of edges of Σ, which is in $O(m)$. In 3D, for a tetrahedralization Σ with m vertices, the time complexity is in $O(m^2)$. The link of p is homeomorphic to a spherical surface. Finding the connected components of the upper and lower links requires a graph traversal process, with a linear time complexity in the number of edges of Σ, which can be in $O(m^2)$ in the worst case.

The classical approach to the extraction of critical points, described above, counts the connected components of the upper and lower links of a vertex. However, other approaches have also been proposed.

In Bajaj et al. [3] the classification of critical points in 2D is performed based on vectors normal at the triangles incident in the vertex. Each triangle t has a unit normal vector $\vec{n_t}$ which is normal to the plane of the triangle and points upwards, i.e., $\vec{n_t} = (a_t, b_t, c_t)$ with $c_t > 0$. Let us consider the convex hull of all points (a_t, b_t) corresponding to the triangles in the star of a vertex p. Vertex p is regular or critical depending on whether the convex hull does not contain or contains the origin $(0, 0)$. Thus, according to [3], a point is critical if the normal space of the adjacent triangles includes vector $(0, 0, 1)$.

A more detailed method for 3D scalar fields has been proposed by Edelsbrunner et al. [10]. Their approach is based on (the reduced Betti numbers of) the lower link of a vertex p.

The link $Lk(p)$ of p is a discrete analogue of a sphere around p. Point p is classified based on the reduced Betti numbers $\tilde{\beta}_{-1}$, $\tilde{\beta}_0$, $\tilde{\beta}_1$, and $\tilde{\beta}_2$ of its lower link $Lk^-(p)$. Informally, the Betti numbers β_0, β_1, and β_2 of a simplicial complex Σ indicate the number of connected components, the number of through holes (tunnels), and the number of voids of the domain of Σ. The reduced Betti numbers $\tilde{\beta}_{-1}$, $\tilde{\beta}_0$, $\tilde{\beta}_1$, and $\tilde{\beta}_2$ are the same as Betti numbers, except that $\tilde{\beta}_0 = \beta_0 - 1$ for non-empty complexes, and $\tilde{\beta}_{-1} = 1$ for empty complexes. The same classification of critical points can be obtained by using Betti numbers, but without the symmetry in the classification of minima and maxima.

The classification of a point p is performed as follows:

- if all reduced Betti numbers of the lower link of p are zero, then p is *regular*;
- if $\tilde{\beta}_{-1} = 1$ ($Lk^-(p)$ is empty), and all other reduced Betti numbers are zero, then p is a *minimum*;
- if $\tilde{\beta}_2 = 1$ ($Lk^-(p)$ is equal to $Lk(p)$), and all other reduced Betti numbers are zero, then p is a *maximum*;
- if $\tilde{\beta}_0 = 1$ ($Lk^-(p)$ has two connected components), and all other reduced Betti numbers are zero, then p is a *simple 1-saddle*;
- if $\tilde{\beta}_1 = 1$ ($Lk^-(p)$ is a cylinder), and all other reduced Betti numbers are zero, then p is a *simple 2-saddle*;
- otherwise (i.e., more than one reduced Betti number is different from 0, and/or some of them is larger than 1), then p is a *multiple saddle*.

A multiple saddle p satisfies $\tilde{\beta}_{-1} = \tilde{\beta}_2 = 0$ and $\tilde{\beta}_0 + \tilde{\beta}_1 \geq 2$. Point p is classified as a multiple saddle, composed of $\tilde{\beta}_0$ 1-saddles, and $\tilde{\beta}_1$ 2-saddles. It has been shown that p can be unfolded into $\tilde{\beta}_0$ simple 1-saddles, and $\tilde{\beta}_1$ simple 2-saddles.

2.2.2 *Detecting Critical Points in a Regular Grid*

For a regular grid, a key issue is the type of interpolation technique used. If the grid is considered as a stepped model, then we can use a characterization of critical points based on counting the connected components of the upper and lower link, as described in Sect. 2.2.1 for simplicial complexes. For example, in the case of a square grid, the field value at a pixel p is compared to the field values of its neighbors, defined based either on the 4- or 8-connectivity model. We have a fixed number of neighbors and, thus, a fixed set of cases. Note that, in case of a 2D grid with 4-connectivity, all saddles will be simple. These approaches [2, 13, 21, 22, 29, 32] are rooted in digital geometry and have been extensively used in image processing [16].

If the regular grid is considered to have field values located at its vertices, then another approach, that we call an *analytic approach*, is used. There is no attempt here at simulating the concept of critical point in the discrete case, but the general idea is that of fitting an interpolating function on the vertices of the grid (at which the field values are known) [2, 24, 25, 30, 31, 33]. The various algorithms differ in the function they use. The selected function usually preserves critical points located at grid vertices, but it may introduce new critical points located inside higher-dimensional cells. Moreover, it is not always globally continuous. Since the interpolating function has a known equation, its critical points can be found analytically. Sometimes they are computed through numerical methods.

The method proposed in [24, 25] uses a bilinear C^0-differentiable interpolating function over a 2D grid, which does not introduce additional minima or maxima, while it may introduce additional saddles inside cells. A grid point p is classified by considering only the elevation of its 4-adjacent neighbors, while a 2-cell, which contains a saddle, is detected by considering the elevation of its four vertices.

In [32], a 3D grid is considered, and a tri-linear interpolating function in each cubic cell is used. In this case also, there may be saddles inside the cubic cells and on their boundaries. The algorithm does not distinguish between 1-saddles and 2-saddles.

Other approaches for 2D or 3D grids [25] use bi-quadratic functions, which provide a globally discontinuous approximation, but guarantee that critical points are constrained to lie at grid vertices. A grid point p is classified based on the characteristics of the functions inside the d-cells incident in p (e.g., in 2D, the function inside a 2-cell can be elliptic, parabolic, or hyperbolic, and a fixed set of cases may occur for p, which are formalized through specific rules).

2.3 Handling the Domain Boundary

A special case in the detection and classification of critical points is represented by the boundary of the domain of the scalar field. Boundary vertices do not have a complete link (homeomorphic to a circle in 2D and to a spherical surface in 3D), but they have an incomplete one (homeomorphic to a segment in 2D and to a disc in 3D). A similar problem arises with boundary pixels and voxels in regular stepped models.

Some methods assume that the domain of the scalar field is a manifold without boundary (e.g., [10]), and, thus, there are no boundary points. However, in most real cases, the domain of a scalar field has a boundary, and thus we must deal with this special case.

The boundary-based methods by Takahashi et al. in 2D [27] and in 3D [28] introduce a virtual minimum (or maximum) which is considered to be adjacent to all boundary vertices of the given simplicial model. This solution, however, causes a non-symmetric treatment of minima and maxima lying on the boundary, depending on the type (minimum or maximum) of the virtual extremum introduced.

Another solution consists of mirroring the function values of adjacent grid points across the boundary (as, for instance, in [13]).

2.4 Presence of Plateaus

Real data often present connected components of vertices (or d-cells in a stepped regular model) all having the same field value. Having in mind the case of a 2D scalar field representing a terrain, these configurations are called *plateaus*, and an edge connecting two vertices with equal field value is called a *flat edge*.

Different solutions are used:

- the notion of a critical point is replaced with that of a critical area, and algorithms use ad-hoc solutions to deal with plateaus (e.g., watershed algorithms consider *regional minima* and other plateaus [18, 19, 23]);
- data are perturbed in a preprocessing step, in order to eliminate flat edges [9, 12].

The first solution implies that connected components of vertices with equal field value (and, thus, of higher-dimensional cells with constant interpolating function) must be identified, classified, and consistently handled. For instance, watershed through simulated immersion [26] can easily identify and treat plateaus during the flooding process. Region-growing methods can identify plateaus which are minima and maxima, and grow regions from them. For plateaus that are not minima or maxima, it is necessary to artificially define one or more entering and/or exiting point [1].

The first solution is very difficult to implement for boundary-based methods, because they should follow lines of steepest slope, which is intrinsically not defined

in a plateau. Even in approaches that can be easily adapted to deal with plateaus, the conventions used to process them are somehow arbitrary (especially for plateaus that are not minima or maxima), and may lead to quite different results (as shown in Sect. 7.4.2).

The second solution (i.e., data perturbation) can be achieved by adding random noise to field values, but this introduces many new critical points, which lead to spurious regions in the computed Morse complexes. Thus, a post-processing step is then required to merge such spurious regions.

Another way to perturb data consists of introducing a conventional order among two points with equal function value, in order to decide that one of them is "above" the other one. For instance, a lexicographic order on the spatial coordinates of points can be used [12]. This has the drawback that a regional minimum (or maximum) may give rise to more minima (or maxima), depending on the relative position of points composing it. Again, a post-processing stage is required to merge regions associated with different minima (or maxima) belonging to the same plateau.

Recently, an algorithm for eliminating flat edges from a 2D scalar field in a morphologically consistent way has been proposed by Magillo et al. [17], which represents a valid alternative solution as it does not introduce new critical points. Experiments presented in [17] show that the similarity between the results of different approaches on the same data increases after using such a method as a preprocessing step.

The method is described for terrains (height fields) although it applies to 2D scalar fields in general. It is based on the observation that changing the elevation of a vertex slightly (i.e., of a smaller amount than the minimum elevation difference between v and its adjacent vertices) is sufficient to eliminate flat edges incident in v (by giving a slope to them), while it does not change the uphill or downhill orientation of other edges incident in v.

Plateaus are progressively reduced and eventually eliminated by iteratively changing the elevation of a vertex v lying on the boundary of a plateau. A vertex v is a candidate for this task if either all flat edges incident in v are consecutive around v (the elimination of v does not change the topology of the plateau), or has exactly two incident flat edges (the elimination of v from the plateau either removes a hole or splits the plateau into two). These situations are illustrated in Fig. 2.2.

The highest priority is given to moves which do not change the topology of plateaus, and make v a regular vertex. Such moves are sufficient to eliminate plateaus without holes which are maxima, minima, or regular, or act as simple saddles. Other plateaus need the application of moves where v becomes a saddle, or the topology of the plateau changes. The priority scheme gives preference to cases in which v becomes a saddle with low multiplicity and/or the topology of plateau does not change.

A plateau which was a maximum or minimum gives rise to a set of regular vertices plus a single maximum or minimum vertex, plus as many simple saddles as holes in the original plateau. A regular plateau gives rise to a set or regular vertices. In other cases, a plateau gives rise to a set or regular vertices plus a number of saddle vertices, whose total multiplicity depends on the total number of terrain portions

around the original plateau having an elevation above/below the plateau itself, in the original configuration.

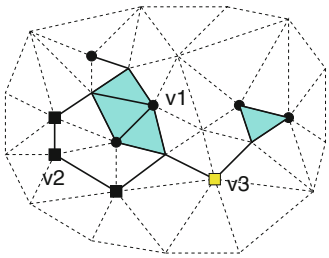

Fig. 2.2 A plateau (formed by *bold edges* and *shaded triangles*) and candidate vertices for elimination. *Black circles* denote vertices, like v_1, which do not change the topology of the plateau. *Black squares* denote vertices, like v_2, which remove a hole from the plateau. *White squares* denote vertices, like v_3, which split the plateau

References

1. L. Arge, J. Chase, P. Halpin, L. Toma, D. Urban, J.S. Vitter, and R. Wickremesinghe. Flow computation on massive grid terrains. *Geoinformatica*, 7(4):283–313, 2003.
2. C. L. Bajaj, V. Pascucci, and D. R. Shikore. Visualization of scalar topology for structural enhancement. In *Proc. IEEE Visualization'98*, pages 51–58. IEEE Computer Society, 1998.
3. C. L. Bajaj and D. R. Shikore. Topology preserving data simplification with error bounds. *Computers and Graphics*, 22(1):3–12, 1998.
4. T. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *American Mathematical Monthly*, 77(5):475–485, 1970.
5. S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys*, 40(4):Article 12, 2008.
6. Y.-J. Chiang, T. Lenz ans X. Lua, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry: Theory and Applications*, 30(2):165–195, 2005.
7. L. Čomić, L. De Floriani, and F. Iuricich. Building morphological representations for 2D and 3D scalar fields. In E. Puppo, A. Brogni, and L. De Floriani, editors, *Eurographics Italian Chapter Conference*, pages 103–110. Eurographics, 2010.
8. L. Čomić, L. De Floriani, and L. Papaleo. Morse-Smale decompositions for modeling terrain knowledge. In *Proc. International Conference on Spatial Information Theory (COSIT)*, volume 3693 of *Lecture Notes in Computer Science*, pages 426–444. Springer, 2005.
9. H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, England, 2001.
10. H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. 19th ACM Symposium on Computational Geometry*, pages 361–370, 2003.
11. H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th ACM Symposium on Computational Geometry*, pages 70–79, 2001.

12. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
13. T. Gerstner and R. Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proc. IEEE Visualization'00*, pages 259–266, 2000.
14. 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*, 14(6):1619–1626, Nov-Dec 2008.
15. A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, Nov-Dec 2007.
16. R. Klette and A. Rosenfeld. *Digital Geometry - Geometric Methods for Digital Picture Analysis*. Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Francisco, 2004.
17. P. Magillo, L. De Floriani, and F. Iuricich. Morphologically-aware elimination of flat edges from a tin. In *Proc. 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013)*, November 5-8 2013.
18. A. Mangan and R. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.
19. F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.
20. X. Ni, M. Garland, and J. C. Hart. Fair Morse functions for extracting the topological structure of a surface mesh. In *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH*, pages 613–622, 2004.
21. L. Papaleo. *Surface Reconstruction: Online Mosaicing and Modelling with Uncertainty*. PhD thesis, University of Genova – Department of Computer Science, 2004.
22. T. K. Peucker and D. H. Douglas. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4:375–387, 1975.
23. J. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms, and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2000.
24. B. Schneider. Extraction of hierarchical surface networks from bilinear surface patches. *Geographical Analysis*, 37(2):244–263, 2005.
25. B. Schneider and J. Wood. Construction of metric surface networks from raster-based DEMs. In S. Rana, editor, *Topological Data Structures for Surfaces*, pages 53–70. John Wiley & Sons Ltd, 2004.
26. P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag, Berlin and New York, 2004.
27. S. Takahashi, T. Ikeda, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographic elevation data. In *Computer Graphics Forum*, volume 14, pages 181–192, 1995.
28. S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004.
29. J. Toriwaki and T. Fukumura. Extraction of structural information from gray pictures. *Computer Graphics and Image Processing*, 7:30–51, 1978.
30. L. T. Watson, T. J. Laffey, and R. M. Haralick. Topographic classification of digital image intensity surfaces using generalized splines and the discrete cosine transformation. *Computer Vision, Graphics, and Image Processing*, 29:143–167, 1985.
31. G. Weber and G. Scheuermann. Automating transfer function design based on topology analysis. In G. Brunnett, B. Hamann, H. Müller, and L. Linsen, editors, *Geometric Modeling for Scientific Visualization*, Mathematics and Visualization. Springer Verlag, Heidelberg, 2004.
32. G. H. Weber, G. Scheuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In *Proc. IEEE Visualization'02*, pages 171–178. IEEE Computer Society, 2002.
33. G. H. Weber, G. Scheuermann, and B. Hamann. Detecting critical regions in scalar fields. In G.-P. Bonneau, S. Hahmann, and C. D. Hansen, editors, *Proc. Data Visualization Symposium*, pages 85–94. ACM Press, New York, 2003.

Springer

http://www.springer.com/978-1-4939-2148-5

Morphological Modeling of Terrains and Volume Data

Čomić, L.; De Floriani, L.; Magillo, P.; Iuricich, F.

2014, XI, 116 p. 58 illus., Softcover

ISBN: 978-1-4939-2148-5