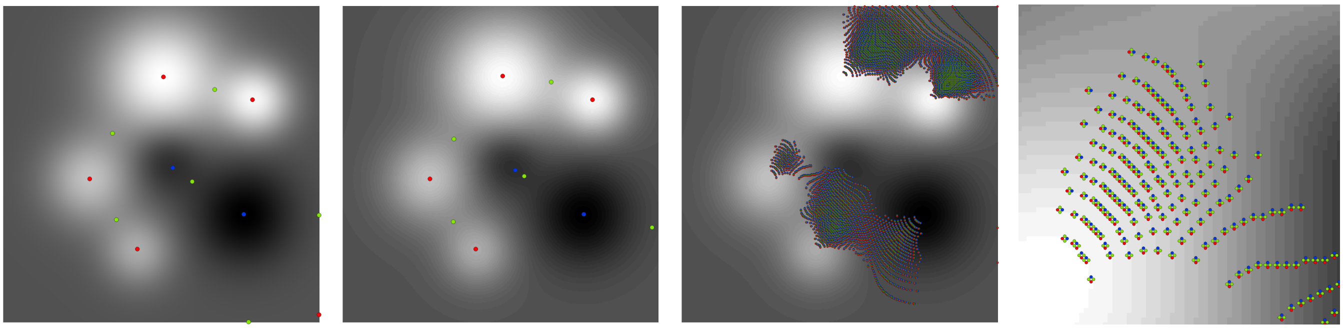
Volume xx (20yy), Number z, pp. 1–10

Disambiguating Flat Spots in Digital Elevation Models

L. Rocca[1] and E. Puppo[2]

[1]IGAG - CNR, Milan, Italy [2]DIBRIS - University of Genoa, Italy



Original data | Our method (quantized data) | Ramps (quantized data) | Detail of the ramps image

Figure 1: *Results on the GaussHills synthetic dataset. Left: the original dataset encoded with double values in the range* $[-1907, 4078]$ *has no flat spots and contains five maxima (red bullets), two minima (blue bullets), and six saddles (green bullets). Middle-left: the dataset quantized at vertical resolution 100 consists of 96 flat spots, covering 100% of the surface; our algorithm detects the minimal number of critical values compatible with its structure, and places them at or near their original locations; two critical points at the boundary (lower right corner) are deleted by quantization. Middle-right: on the same quantized dataset, an algorithm that disambiguates flat spots by building ramps based on the coordinates of points in the grid – as customary in most TDA systems – generates an enormous number of spurious critical points. Right: a zoom reveals that the spurious points come in clusters at boundaries between adjacent flats; even if spurious critical points were removed with persistence analysis, the survivors would be generally located incorrectly (e.g., the spot at the lower left corner should contain a maximum in the middle, while it will be necessarily placed at its boundary).*

Abstract

We consider Digital Elevation Models (DEMs) encoded as regular grids of discrete elevation data samples. When the terrain's slope is low relative to the data's vertical resolution, the DEM may contain flat spots*: connected areas where all points share the same elevation. Flat spots can hinder certain analyses, such as topological characterization or drainage network computations. We discuss the application of Morse-Smale theory to grids and the disambiguation of flat spots. Specifically, we show how to characterize the topology of flat spots and symbolically perturb their elevation data to make the DEM compatible with Morse-Smale theory while preserving its topological properties. Our approach applies equivalently to three different surface models derived from the DEM grid: the step model, the bilinear model, and a piecewise-linear model based on the quincunx lattice.*

1. Introduction

Topological data analysis (TDA) finds application in many fields, such as computer graphics [WG09], scientific visualization [Tie17], geographical information science [DWW17, RJP17, XIDF20], environmental science [VMN*19], genomics [RB19], biomedicine [SL22], fluid dynamics [GSW12, SWTH07], material science [GDN*07], chemistry [OSPGT20], just to mention a few. A recent account of the main techniques in TDA can be found in [DW22].

The characterization of functions in terms of their critical points is fundamental to powerful TDA tools, such as the Morse-Smale complex and persistent homology [EH10, DW22, BDFFP08].

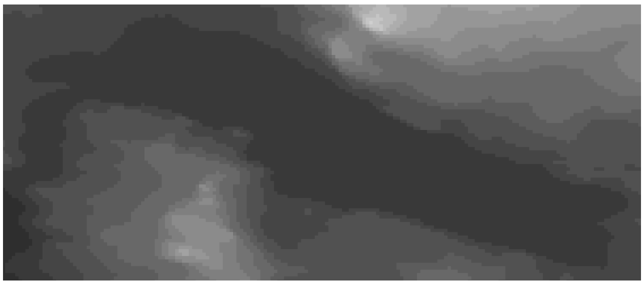


Figure 2: *A lake (dark region) in a DEM constitutes an intrinsic flat spot since all its points are at the same elevation. An emissary river (on the left) consists of a series of flat spots: changes of value occur only where the difference in altitude exceeds data resolution.*

However, these tools often rely on assumptions that may not hold in real-world data. One crucial assumption, frequently violated, is the absence of *flat spots*, regions where the function has a constant value. Flat spots can be inherent in the data or arise due to quantization, especially when numerical precision is low relative to the data's dynamic range. Consider topographic maps: the most accurate, openly distributed global models currently have a horizontal resolution of approximately 30 meters and a vertical resolution of one meter [JAS16, NJ19, NAS00]. A lake on such a map represents an intrinsic flat spot; in contrast, a river flowing through a plain has an inherent slope, but it will be represented as a series of flat spots if it is less steep than the lowest slope that can be expressed in the model, depending on its vertical and horizontal resolution (Fig. 2). In fact, any geographical area with a slope less than $1/30$ will be represented by a group of terraces in the aforementioned models, with a banding effect. Counter-intuitively, a better horizontal resolution paired with the same vertical resolution worsens the problem.

The presence of flat spots significantly complicates the direct application of mathematical principles to real-world data, leading to intricate algorithms and potential ambiguities, not just in TDA, but also in other fields, e.g., hydrological analysis [NGS*08]. Depending on its banks' configuration, a lake can be categorized as either: a sink, when all surrounding areas are at a higher elevation; a regular area, when a single stream flows out; or even a saddle area, when multiple streams emanate from different boundary zones. In contrast, a flat spot representing a river segment should always be classified as a regular area, with its boundary divided into an upper end, a lower end, and two side banks.

Several authors have attempted to mitigate the impact of flat spots by adding random perturbations to the data. However, this approach introduces topological noise in the form of numerous spurious critical points. Other methods, such as those employing ramps [RWS11], may also lead to spurious critical points and misclassify flat spots (see Fig. 1). Magillo et al. [MDFI13] proposed a systematic approach to remove flat spots from Triangulated Irregular Networks (TINs) while preserving correct topology and avoiding spurious critical points. Nonetheless, their classification of flat spots is limited, and their algorithm is relatively complex, requiring the discrimination of various cases. Inspired by their work, we address the analysis of flat spots within Digital Elevation Models and, more specifically, grid data. Our contributions are as follows:

1. We consider three continuous models for functions sampled on a grid: the *step model*, the *bilinear model*, and the *quincunx linear model*. We demonstrate that the classification of critical points is equivalent for these models and that, unlike the standard simplicial model for TINs, they cannot contain multiple saddles.
2. We introduce an algorithm for classifying the simplest topology of flat spots.
3. We introduce an algorithm for the symbolic disambiguation of flat spots, aligning with this classification. Specifically, we establish an ordering of neighboring data elevations within a flat spot, leading to the desired singularities.

We evaluate our algorithms using both synthetic and real-world datasets. To assess the impact of data resolution on our disambiguation algorithm, we analyze various versions of the same datasets captured at different vertical resolutions within the dynamic range. Our goal is to demonstrate that the results obtained from lower-resolution data align with those from higher-resolution data, which typically contain fewer or no flat spots.

2. A continuous Morse model for grids

Morse theory and the Morse-Smale complex were originally defined for smooth functions. We briefly summarize the basic concepts, referring to [EH10, Mat02] for a more thorough formal treatment of this subject. Next, we consider the application of such theory to sampled data: we discuss the existing approaches and propose a continuous model, which applies to the most common representations of functions from data on a grid.

2.1. Background

Smooth setting. A smooth function $f : \Omega \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ is a Morse function if its critical points are isolated, or equivalently, if its Hessian is non-zero at critical points. For a minimum $p \in \Omega$, the unstable submanifold is the set of points in Ω that lie on integral curves of f originating from p. This region is bounded by separatrices, which are integral curves connecting maxima and saddles. The unstable submanifolds partition Ω. Similarly, the stable submanifold of a maximum q is the set of points on integral curves converging to q, bounded by separatrices connecting minima and saddles. The stable submanifolds also partition Ω. If these two partitions intersect transversally, their overlay is called a Morse-Smale complex, which can be represented by a planar graph with edges corresponding to separatrices connecting saddles to maxima or minima.

Models for sampled data. Computing Morse-Smale complexes for sampled data necessitates adapting the concept to discrete settings. Two primary approaches exist:

1. Piecewise-Linear Approximation: In this approach, the smooth function is approximated by a piecewise-linear signal interpolating values at the vertices of a simplicial complex discretizing the domain [EH10]. It can be applied to TINs only.
2. Discrete Morse Theory: This approach discretizes the domain with a CW complex, analyzes the incidence graph of cells, and

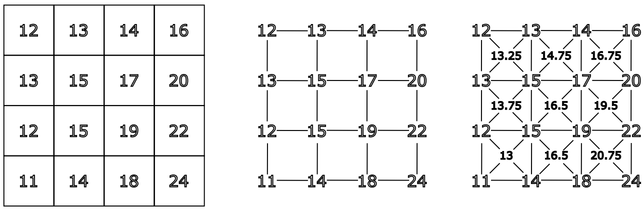


Figure 3: *The step model is a regular grid where each cell (pixel) is assigned a constant value (left). The bilinear model is defined on the dual grid, with data assigned to vertices and the function extended by bilinear interpolation within cells (center). The quincunx linear model is defined on the triangulation obtained by adding a vertex inside each cell of the dual grid and splitting the cell into four triangles; the value assigned to the quincunx vertex depends on the values at the corners of the cell (right).*

assigns discrete gradients to its arcs [For98]. It can be applied to either TINs or grids.

The piecewise-linear model offers a closer connection to the smooth setting, leading to more accurate critical point and separatrix locations. However, it introduces artifacts like multiple saddles (not present in the smooth setting) and ambiguities in tracing separatrices, requiring careful handling in algorithms. Applying this model to regular grid data involves defining a simplicial complex with grid vertices, and different choices can yield different results.

Discrete Morse Theory, grounded in a formal framework, has gained popularity. It provides a cleaner, unambiguous Morse-Smale complex once a discrete gradient is defined. Since it operates on CW complexes, it's directly applicable to regular grids, too. However, defining a discrete gradient field consistent with the input data remains challenging. As discussed in [GBP12, RGH*12], the standard algorithm [RWS11] may produce a Morse-Smale diagram that doesn't converge to the continuous solution with refinement. Recent work [TAW24] highlights the drawbacks of this approach, including potential inaccuracies in geometry and topology.

2.2. Models for the grid

We consider three different extensions of grid data to a continuous domain (see Fig. 3):

1. *Step Model*: Grid data represent cells of a regular square grid, and the function value is constant within each cell, similar to pixels in a digital image. This model assumes each datum approximates the underlying function in its neighborhood, resulting in a discontinuous approximation.
2. *Bilinear Model*: Grid data correspond to vertices of a regular square grid, and the function value over each cell is interpolated bilinearly from its four corners. This yields a continuous, piecewise-smooth function, but it lacks smoothness at the edges and vertices of the tessellation.
3. *Quincunx Linear Model*: Starting with the bilinear model's tessellation, each square cell is subdivided into four triangles by adding a vertex at its center. The value assigned to this vertex will be discussed later. This model results in a piecewise-linear function, as in [EH10].

We demonstrate the equivalence of these models regarding critical point classification and the absence of multiple saddles. Detailed computations for the Morse-Smale complex on these models will be presented in a separate study.

We identify each point p in the input grid with a pair of integer coordinates (i, j) and denote its value as $g(p)$ or $g(i, j)$. Two points are considered *4-adjacent* if they differ by one unit in one coordinate while remaining equal in the other. They are *8-adjacent* if they differ by one unit in both coordinates. For now, we assume the input grid has no *flat edges*, meaning no pair of 4-adjacent points shares the same value.

We consider the bilinear model first. It is straightforward to verify that a bilinear cell cannot contain a minimum or a maximum in its interior. Furthermore, it contains a simple saddle if and only if the smallest (respectively, largest) two values of g at its corners lie at opposite endpoints of diagonals. Moreover, critical points cannot exist in the interior of the edges, where the function consists of linear ramps. However, vertices of the grid can be regular points or critical points of all types. For a vertex p and its four adjacent patches, although function g is not smooth at p, its directional derivative at p is well-defined for any direction v and varies continuously as v rotates about p. The type of point p can be characterized by counting the changes of sign of the directional derivative during its rotation about p. Equivalently, we can count the sign changes between $g(p)$ and the values of g at its 4-adjacent vertices, rotating about p. This sign can change at most once in each cell incident at p, leading to a total of zero, two, or four sign changes:

- If there are no sign changes, p is either a minimum or a maximum, depending on the sign of the directional derivative.
- If there are two sign changes, p is regular.
- If there are four sign changes, p is a simple saddle.

Note that, the 8-adjacent vertices of p are irrelevant as the directional derivatives of g at p do not depend on them. In summary, for the bilinear model, critical points are located at vertices and inside cells; cells can only contain saddles; and all saddles are simple (e.g., there are no monkey saddles).

Concerning the step model, we note that it is defined on the dual lattice of the bilinear model. For clarity, we refer to the elements of this dual lattice as *pixels* (containing data values) and *corners* (vertices); the boundaries between 4-adjacent pixels remain *edges*. To characterize critical points in the step model, it suffices to consider dual rules. A corner x can be either regular or a simple saddle. It is a saddle if and only if there are four sign changes in the difference between the values of g at pairs of cells incident at x, taken in cyclic order about x. A pixel can be either regular or any type of critical point, depending on the number of sign changes in the difference between its value and the values of its 4-adjacent pixels, taken in cyclic order about it.

Finally, we define the quincunx model as follows. For a quincunx vertex q within a regular cell c in the bilinear model, assign the value:

$$g(q) = \frac{1}{4} \sum_{i=1}^{4} g(c_i)$$

where $g(c_i)$ are the values of g at the four corners of c. For a quincunx vertex q within a saddle cell c in the bilinear model, assign the value: $g(q) = g(c_s)$ where $g(c_s)$ is the value of g at the saddle point of the bilinear patch within c. While this assignment might introduce slight misalignments of saddles compared to the bilinear model, this approach is crucial for maintaining correspondence between critical points.

To determine critical points in the quincunx model, we apply the piecewise-linear model rules [EH10]. All critical points in this model occur at vertices. The definitions above show that each quincunx vertex q is classified as the cell c containing q in the bilinear model. To show that the type of a grid vertex p remains consistent between the bilinear and quincunx models, we need only demonstrate that quincunx vertices do not introduce additional sign changes around p. Let r and s be two consecutive 4-neighbors of p, and let q be the quincunx vertex inside the cell c with p, r, s on its boundary. If $g(p) - g(r)$ and $g(p) - g(s)$ have opposite signs, then the sign of $g(p) - g(q)$ is irrelevant for determining the number of sign changes rotating about p from r to q to s. Conversely, let $g(p) < g(r)$, $g(p) < g(s)$, and let t be the fourth corner of cell c. If q is regular, we must have $g(p) < g(t)$, hence $g(p) < g(q)$ because $g(q)$ is the average of the four corner values; if q is a saddle, we necessarily have $g(p) < g(q)$. The case $g(p) > g(r)$ and $g(p) > g(s)$ is symmetric.

Another common approach to construct a piecewise-linear simplicial model from a grid involves splitting each square along a diagonal. While this method avoids introducing additional vertices, it is susceptible to bias due to the chosen diagonal direction and may yield inaccurate results, as noted in [TAW24]. Furthermore, it can contain multiple saddles. These limitations are why we opted for the quincunx model instead.

We will describe our algorithms on the step model. However, they depend only on the 4- and 8-relations of each vertex and work just as they are on all three models.

3. Classification of flat spots

We now accommodate input grids that may contain flat spots. A flat spot is identified as a 4-connected set of pixels with identical values. Consider the chains of edges that separate the pixels of a flat spot from the other pixels. The flat spot is the region of the domain enclosed by these chains and encompassing all the flat vertices. Refer to Figure 4 for an illustration. It's important to note that distinct flat spots can be adjacent but remain separate entities.

We now determine the minimum number of critical points that a flat spot must contain, along with their types, to ensure compatibility with the overall topology of the dataset.

We label each edge on the boundary of a flat spot with a $+$ or a $-$ sign, depending on whether its incident pixel from the outer region has a value higher or lower than the value within the flat spot, respectively. Given a chain γ_i forming a boundary of the flat spot, we count the number k_i of changes of sign along γ_i; this value is necessarily even. Let us define $h_i = k_i/2 + 1$ and

$$H = \sum_{i=1}^{b} h_i,$$

12	18	14	16	17	15	14	16	15	16	18	19
13	19	20	20	20	19	18	19	20	20	17	20
14	20	19	18	20	20	20	20	20	20	19	17
17	19	18	19	20	20	20	24	22	20	20	18
19	20	14	16	20	20	23	22	21	20	20	16
23	21	19	20	17	20	20	20	20	20	17	19
24	22	21	22	19	18	20	20	20	15	19	22
25	24	23	24	20	19	18	19	17	16	18	21

Figure 4: *A flat spot in the step model is a 4-connnected set of pixels with the same value (blue area), which is bounded by one or more chains of edges (red lines).*

where b is the number of boundaries of the flat spot. We then have the following rules:

A. If $k_i = 0$ for all $i = 1, \ldots, b$ and all signs at all boundaries are $+$, then the flat spot must contain a minimum and $b-1$ saddles.
B. If $k_i = 0$ for all $i = 1, \ldots, b$ and all signs at all boundaries are $-$, then the flat spot must contain a maximum and $b-1$ saddles.
C. Otherwise, the flat spot must contain $H-2$ saddles; in particular, if $H = 2$ then the flat spot is a regular region.

Note that $H \geq b$, thus if $H < 2$ we are necessarily in one of the first two cases. Note also that, in the first two cases, if the flat spot has just one boundary then it contains only a minimum or a maximum.

To prove the rules enumerated above, it is convenient to extend the domain to a topological sphere by adding a dummy region, 4-connected to all pixels on the boundary of the initial domain. The value assigned to the dummy region, whether the largest or smallest compared to other pixels, is irrelevant to our subsequent analysis. Let M, m, and s represent the total number of maxima, minima, and saddles, respectively, in this extended dataset. While we omit a complete proof for brevity, a sketch of the proof stems from the following observations:

1. The Poincaré-Hopf theorem implies that $M + m - s = 2$, and there must be at least one minimum and one maximum.
2. Integral lines that leave the flat spot towards an external region must necessarily terminate at a critical point within that external region. This is because an integral line that descends (or ascends) upon leaving the flat spot cannot re-enter it.
3. Due to Observation 2, if an external region is bounded by chain γ_i, it follows that the minimal set of critical points in that region is composed by either one maximum and $h_i - 1$ minima, or one minimum and $h_i - 1$ maxima, without any saddles. This can be demonstrated by considering the bundles of integral lines that extend into the region from the segments of γ_i between sign changes. Given that each integral line must reach a maximum or minimum and the lines cannot intersect, a total of h_i critical points that aren't saddles are necessary and sufficient to accommodate all the bundles of integral lines.
4. The rules above follow from the need of satisfying Observation 1 after determining the minimum set of critical points in all ex-

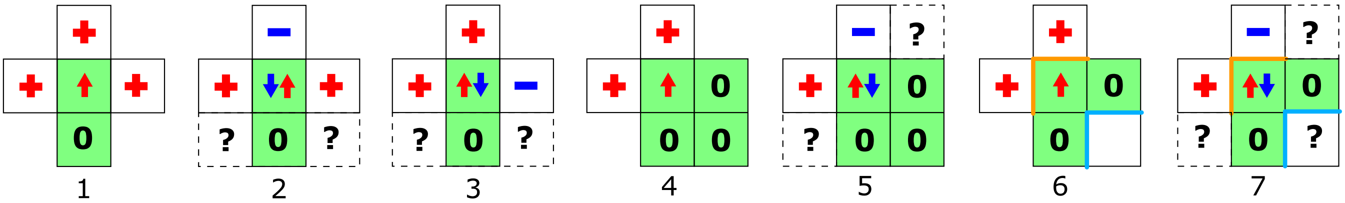


Figure 5: *Moves to perturb pixels with related priorities. Green pixels indicate unprocessed areas within the flat spot; the central pixel is currently processed. Signs $+$, $-$, and 0 denote values higher, lower, or equal to the central pixel, respectively. Arrows represent the direction of perturbation (raise/lower). Question marks highlight 8-neighbors that must be analyzed to set the proper direction of perturbation to minimize the number of generated saddles. Bold edges outline the boundary loops of the unprocessed area, with different colors for distinct boundaries. Priorities are as follows (lowest label first): 1) Three non-flat neighbors with the same sign: Avoid creating a maximum or minimum. 2) Three non-flat neighbors with alternating signs: prefer move that creates fewer saddles (possibly zero). 3) Two consecutive non-flat neighbors with the same sign and a third with the opposite sign: prefer move that creates fewer saddles. 4) Two consecutive non-flat neighbors with the same sign and the 8-neighbor between the flat neighbors is also flat: avoid creating a maximum or minimum. 5) Two consecutive non-flat neighbors with alternating signs and the 8-neighbor between the flat neighbors is also flat: prefer move that creates fewer saddles. 6) Same as 4, but with a non-flat 8-neighbor between the flat 4-neighbors; the orange and blue loops must be distinct before the operation and merged after it. 7) Same as 5, but with a non-flat 8-neighbor between the flat 4-neighbors; the orange and blue loops must be distinct before the operation and merged after it.*

ternal regions with Observation 3. In particular, Rules 1 and 2 descend from the need of having at least one maximum and one minimum, which must belong to the flat spot if no outer region contains it; while Rule 3 follows from satisfying Poincaré-Hopf with a number of saddles that compensates for the number of maxima and minima belonging to the external regions.

Our algorithm for classifying flat spots is very simple. We begin by identifying all groups of 4-connected pixels that define flat spots through a grid traversal. For each group, we extract the edges along its boundaries by examining the 4-adjacencies of its vertices. Next, we process this set of edges by starting with one of them and following the chain it belongs to. Each edge leads to a corner, and there is exactly one other edge in the chain exiting that corner. We continue this traversal until returning to the initial edge. As we traverse a chain, we collect labels and count sign changes. This process is repeated until all edges have been processed, ensuring that all boundaries are traversed. Finally, we apply the previously described rules to classify the flat spot.

4. Disambiguation of flat spots

To disambiguate flat spots, we assign symbolic displacements to the pixels of each flat spot. Consider the dual edges of the lattice, which connect the centers of pixels. We assign an orientation to each dual edge, from its lower to its higher endpoint. While dual edges that connect pixels with different values have predefined orientations, the orientations of edges connecting pixels within a flat spot depend on how such pixels are perturbed. Increasing [decreasing] the value at a pixel of the flat spot makes it higher [lower] than its unperturbed flat neighbors.

Any set of displacements that maintain relationships with pixels outside the flat spot is valid for disambiguation. However, arbitrary perturbations can introduce an excessive number of critical points within the flat spot. We minimize the number of introduced critical points, matching the classification from the previous section.

Our algorithm processes flat spots sequentially, preserving their relationships with surrounding data. For a given flat spot, we process its pixels one by one, starting at the boundary and moving inward until all pixels are processed. The algorithm's rationale is to consume unprocessed pixels while minimizing the introduction of critical points. To this aim, we employ a queue containing candidate pixels on the boundary of the unprocessed set. We extract pixels from the queue and apply symbolic perturbations, thus orientating the dual edges that connect them to other unprocessed pixels within the flat spot. The unprocessed set shrinks by one pixel at each move until just one unperturbed pixel is left.

A single perturbation consists of (symbolically) increasing or decreasing the elevation of a pixel with respect to its neighbors in the unprocessed set while maintaining the same relations with its neighbors in the processed/outer set. An arbitrary perturbation might generate critical points at the processed pixel, its 4-neighbors, or its corners. We allow only a given set of legal moves, guaranteeing that the number of critical points in the unprocessed set decreases according to any critical point newly generated by a perturbation move. The analysis to select legal moves stems from rules A, B, and C given in Sec. 3 and is omitted here for brevity. In particular, for cases A and B, the maximum/minimum is always generated by the last move while intermediate moves may possibly generate only saddles. The legal moves are illustrated in Fig. 5 and explained in the following:

- Moves 1, 2, 3 erode thin protrusions of the flat spot, where *thin* refers to pixels of the flat spot that have only one neighbor in the unprocessed region. The direction of Move 1 is obliged to avoid creating a maximum or minimum. The perturbation applied in Moves 2 and 3 is selected by analyzing the content of the 8-adjacent pixels marked with a **?**, in such a way that a minimum number of saddles (possibly zero) is generated.
- Moves 4 and 5 erode solid parts of the flat spot, where *solid* refers to those parts that contain at least a block of 2×2 pixels. These moves refer to pixels that have two 4-neighbors, plus an

8-neighbor forming a 2×2 block with them, all belonging to the unprocessed region. Move 4 is obliged, similarly to Move 1. The perturbation for Move 5 is decided similarly to Moves 2 and 3.
- Move 6 and 7 break thin chains of pixels, thus merging two distinct boundaries (marked by the orange and blue lines) into one boundary. Again, Move 6 is obliged, while Move 7 is decided by analyzing the content of the 8-neighbors marked with a **?**. Note that, if the orange and blue lines belong to the same boundary, we consider the move not legal, thus avoiding disconnecting the unprocessed subset.

The topological validity of the result is warranted no matter the order in which the pixels are processed. In the simplest case, moves could be applied in any order and a simple FIFO queue could be used. However, the order of moves influences the location of the critical points: we obtained the best results by prioritizing the moves according to the labels in Fig. 5 (lowest label first).

The pseudo-code is illustrated in Algorithm 1. The algorithm takes in input a *grid of states* S and a set of flat spots F. Grid S contains, for each pixel p in the input grid, its state with respect to its 4-connected neighbors: for each neighbor, the state can be $+$, $-$ or 0, depending on the neighbor having a value higher, lower or same as p. Set F contains the flat spots in the input dataset, where each flat spot f consists of a collection of pixels. The algorithm disambiguates all zeros in S changing each of them into a $+$ or a $-$ and returns the updated grid.

For each flat spot f, a priority queue Q is initialized with all pixels of f that belong to one of the configurations depicted in Fig. 5. Each pixel p is pushed into Q together with its priority P and its current state $S[p]$ (lines 3-6). If two pixels have the same priority, the order of push rules (FIFO). Then, elements of the queue are processed until the queue becomes empty (lines 7-26). After each pixel p is popped from the queue (line 8), function `Invalid` tests its state in the queue against its current state in S; if the state has changed, the pixel is discarded (lines 9-10). Moreover, if the pixel has two non-flat neighbors in its state, we must check if the 8-connected pixel between those neighbors has changed. If it's non-flat and on the same boundary, we discard this element and proceed to the next one (lines 13-14). If it was flat and it has become non flat, we we simply push p into Q again with an updated priority (lines 15-17). Otherwise, function `GetPerturbation` returns the new state to be assigned to p. Function `UpdateState` accordingly updates p in S (lines 18-19). Finally, function `UpdateNeighborState` updates the state of the pixels that are 4-adjacent to p in S, and each such neighbor that has a non-null priority after having changed state is pushed into Q (lines 23-26).

Algorithm 1: Disambiguation of flat spots

```
   Input: Grid of states S, Set of flat spots F
   Output: Disambiguated grid S
1  Q ← ∅
2  forall f ∈ F do
3      forall p ∈ f do
4          P ← GetPriority(p,S)
5          if P > 0 then
6              Q.push(p,P,S[p])
7      while Q ≠ ∅ do
8          p ← Q.pop()
9          if Invalid(p,S) then
10             continue
11         if p.Priority() ≥ 4 then
12             P ← GetPriority(p,S)
13             if P = 0 then
14                 continue
15             if P > p.Priority() then
16                 Q.push(p,P,S[p])
17                 continue
18         s ← GetPerturbation(p,S)
19         UpdateState(p,S,s)
20         forall q 4-adjacent to p do
21             if q is not flat then
22                 continue
23             UpdateNeighborState(q,p,S)
24             P ← GetPriority(q,S)
25             if P ≠ 0 then
26                 Q.push(q,P,S[q])
27 return S
```

5. Experimental results

We experimented our algorithms on two synthetic datasets that sample mathematical surfaces and on three real-world DEMs. The five datasets are: **CosCos**, a surface sampled from a sinusoidal function, dampened with a Gaussian centered at the origin; **GaussHills**, a surface sampled from the sum of different Gaussian functions; **Aletsch Glacier**, a section of the swisstopo DEM with a 25m horizontal resolution [swi15], centered around Konkordiaplatz in the Aletsch glacier; **Graian Alps**, a section of the SRTM global 3 arc-seconds (90m) dataset [NAS00], in the version fixed by [dF23], roughly centered on the Mont Blanc Massif; **Finland**, another section of the previous dataset, representing the region around lakes Pyhäjärvi and Kolima north of Central Finland in Scandinavia. For each dataset, we produced several versions quantized at decreasing vertical resolutions. The synthetic datasets were originally encoded in double precision; the vertical resolution of natural DEMs is 1 meter. All datasets were quantized up to 100m, except from Finland: given its lower dynamic range, we chose to quantize up to 16m. For all datasets, in the coarsest version flat areas cover most of the surface ($\sim 99\%$). Table 1 shows statistics about the various datasets and corresponding results. The total count of critical points after classifying the flat spots always satisfies the Poincaré-Hopf formula.

In the synthetic datasets, the distribution of critical points remains substantially stable throughout the various level of quantization, because most of their morphological features are at a scale larger than the data resolution. On the contrary, on real world, noisy datasets that contain many features at a tiny scale, the quantization acts as a filter and the number of critical points tends do decrease drastically at the coarsest vertical resolutions.

Dataset: Grid size, Dynamic range	Dataset: Vertical resolution	Extremalities: Type Min	Extremalities: Type Max	Extremalities: Type Sad	Flat spots: Number	Flat spots: Area (%)	Flat spots: Type Reg	Flat spots: Type 1 Min	Flat spots: Type 1 Max	Flat spots: Type 1 Sad	Flat spots: Type 2 Sad	Flat spots: Type >2 Sad	Flat spots: Type Mixed	Elapsed time (sec): Classify	Elapsed time (sec): Disambiguate
CosCos 256×256 [-2450,2990]	double	20	25	44	-	-	-	-	-	-	-	-	-	-	-
	1	20	25	44	2435	9.28	2399	10	13	13	0	0	0	0.004	0.002
	2	18	25	42	4102	17.10	4048	8	23	23	0	0	0	0.006	0.004
	5	16	25	40	6312	31.70	6250	13	23	26	0	0	0	0.010	0.007
	10	12	25	36	7947	47.98	7879	8	25	35	0	0	0	0.016	0.011
	20	14	25	38	8583	66.94	8509	14	25	35	0	0	0	0.022	0.016
	50	12	25	36	6500	89.74	6441	10	25	16	4	4	0	0.027	0.021
	100	14	24	37	2826	98.58	2779	14	24	6	0	3	0	0.026	0.026
GaussHills 400×400 [-1907,4078]	double	2	5	6	-	-	-	-	-	-	-	-	-	-	-
	1	2	5	6	5614	15.43	5602	2	4	6	0	0	0	0.012	0.008
	2	2	5	6	7305	23.52	7293	1	5	6	0	0	0	0.017	0.012
	5	2	5	6	10969	38.93	10957	2	4	6	0	0	0	0.027	0.021
	10	2	5	6	15175	56.26	15162	2	5	6	0	0	0	0.039	0.029
	20	2	4	5	19518	80.08	19507	2	4	5	0	0	0	0.053	0.041
	50	2	4	5	3801	99.11	3790	2	4	5	0	0	0	0.060	0.054
	100	2	4	5	96	100.00	85	2	4	5	0	0	0	0.045	0.058
Aletsch Glacier 703×697 [813,4268]	1	755	1850	2604	34557	18.12	34243	54	88	170	2	0	0	0.044	0.030
	2	519	1602	2120	46193	30.59	45810	53	122	206	2	0	0	0.076	0.052
	5	244	1150	1393	59604	52.18	59193	35	147	225	1	3	0	0.121	0.088
	10	129	882	1010	65354	73.85	64906	15	201	227	4	1	0	0.163	0.122
	20	104	595	698	38816	91.82	38344	25	222	213	10	2	0	0.189	0.151
	50	22	263	284	4159	99.70	3897	5	141	96	17	3	0	0.157	0.160
	100	20	188	207	406	99.97	238	6	109	30	12	11	0	0.117	0.193
Graian Alps 1200×900 [450,4786]	1	3810	7280	11089	24363	6.10	23350	359	228	424	0	2	0	0.031	0.024
	2	3402	6935	10336	42302	10.27	40688	457	417	737	3	0	0	0.053	0.039
	5	2483	6179	8661	86382	21.29	83959	471	773	1168	8	3	0	0.110	0.078
	10	1716	5244	6959	136079	36.78	132940	460	1108	1562	9	0	0	0.194	0.137
	20	952	3983	4934	180642	60.83	177322	295	1344	1666	13	2	0	0.324	0.229
	50	380	2386	2765	109872	93.27	107177	129	1214	1273	68	11	0	0.494	0.376
	100	215	1507	1721	13195	99.62	11440	72	962	520	127	74	0	0.438	0.412
Finland 2401×1201 [68,232]	1	2547	5084	7630	210232	88.51	201249	1445	3448	3481	376	240	7	1.045	1.014
	2	1218	3712	4929	100946	95.97	94691	753	2800	2201	305	198	2	0.965	1.121
	4	897	3131	4027	38366	98.96	33219	559	2655	1378	293	265	3	0.775	1.153
	8	449	2566	3014	10254	99.86	6819	290	2292	425	165	266	3	0.647	1.295
	16	338	1711	2048	2532	99.99	539	219	1555	90	31	105	7	0.538	1.958

Table 1: *Statistics on the datasets and results. We report the name, the grid size, and the original range of elevations. For each dataset, we built several versions quantized at lower vertical resolutions. We report the total number of critical points, including: those located on non-flat vertices; and those within flat spots, computed using the method described in Sec. 3. The next columns show the number of flat spots in each dataset, their total coverage percentage, and the classification of flat spots. Each flat spot can contain: a single minimum; a single maximum; a number of saddles; or a mix – one minimum or maximum, plus a number of saddles. The last two columns show execution times for our algorithms, classification (Sec. 3) and disambiguation (Sec. 4), running on a Apple M3 Pro with 18GB of RAM. The current implementation is a single-core prototype, without any parallelism or optimization.*

After running the disambiguation algorithm described in Sec. 4, we verified that there is no change in topology. As expected, the number of critical points introduced by the algorithm corresponds to the number computed by the classification algorithm for all flat spots in all datasets. This equivalence holds both globally, on the whole dataset, and locally for each flat area. The only difference is that the critical points are now located inside the former flat areas.

A visual inspection confirms that all maxima/minima and most saddle points inside the flat areas are located in positions that are intuitively correct, near their position in the original dataset. This is easier to spot in the synthetic datasets, as seen in Fig. 1 and Fig. 6. Some artifacts arise close to the boundary, where some saddles are slightly displaced, even though their position is still compatible with the overall terrain morphology. Figures 7 (full dataset), 8 and 9 (zoom-in) show a comparison between features in the original terrain vs critical points computed on the highest quantization level for natural datasets. The position of computed critical points, located inside flat spots that cover wide areas, is compatible with the overall morphology of the terrain and shows good performance in recovering the underlying information that was lost by decreasing the vertical resolution. This holds true on different types of terrain: the Aletsch Glacier and Graian Alps represent steep alpine areas, full of dramatic drops and rises; the Finland dataset is even more demanding, as it depicts low, gently rolling lands with intricate and complex features of low dynamic range and wide spatial distribution, carved a long time ago by the Scandinavian Ice Sheet.

6. Concluding remarks

We have examined three distinct models for digital surfaces derived from grid data: the step model, the bilinear model, and the quincunx linear model. We have established the equivalence of these models in terms of critical point classification and the interpretation of grid data as Morse functions. Subsequently, we have addressed the classification of flat spots, a common occurrence in real-world data, based on the morphology of the surface they represent. Our algorithm determines the necessary number and type of critical points

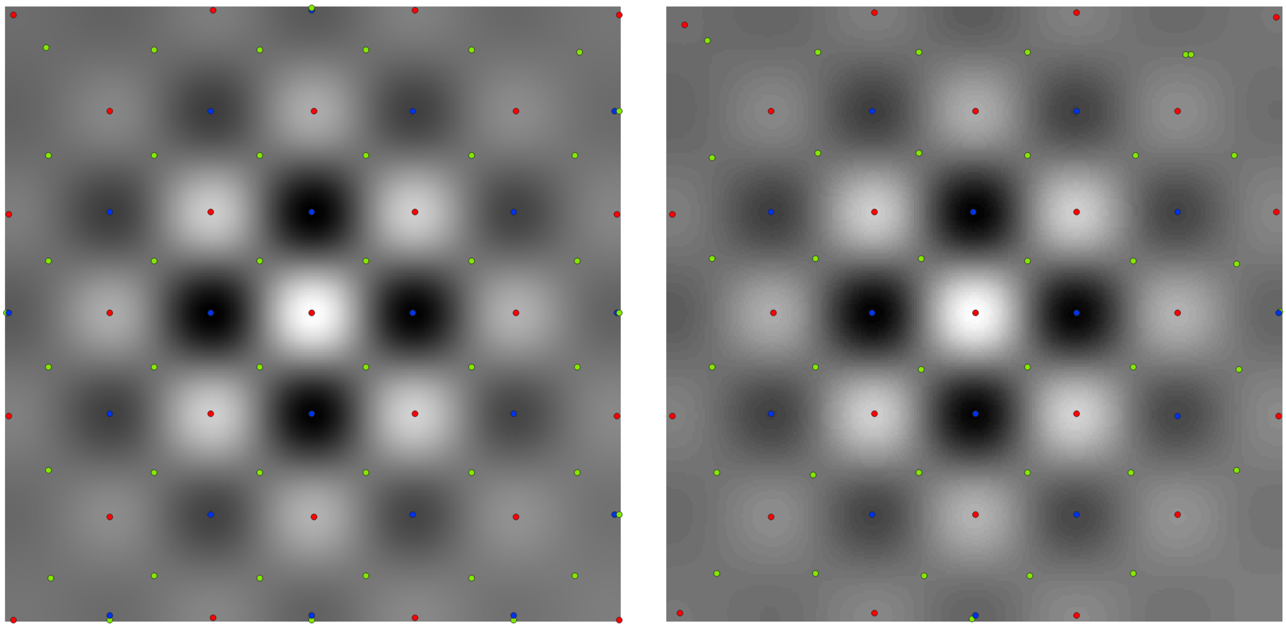


Figure 6: *CosCos dataset. Left: the original dataset and its features, with no flat areas. Right: the dataset at the highest quantization level (100), with recovered critical points. At this level,* 100% *of the DEM is covered by flat spots.*

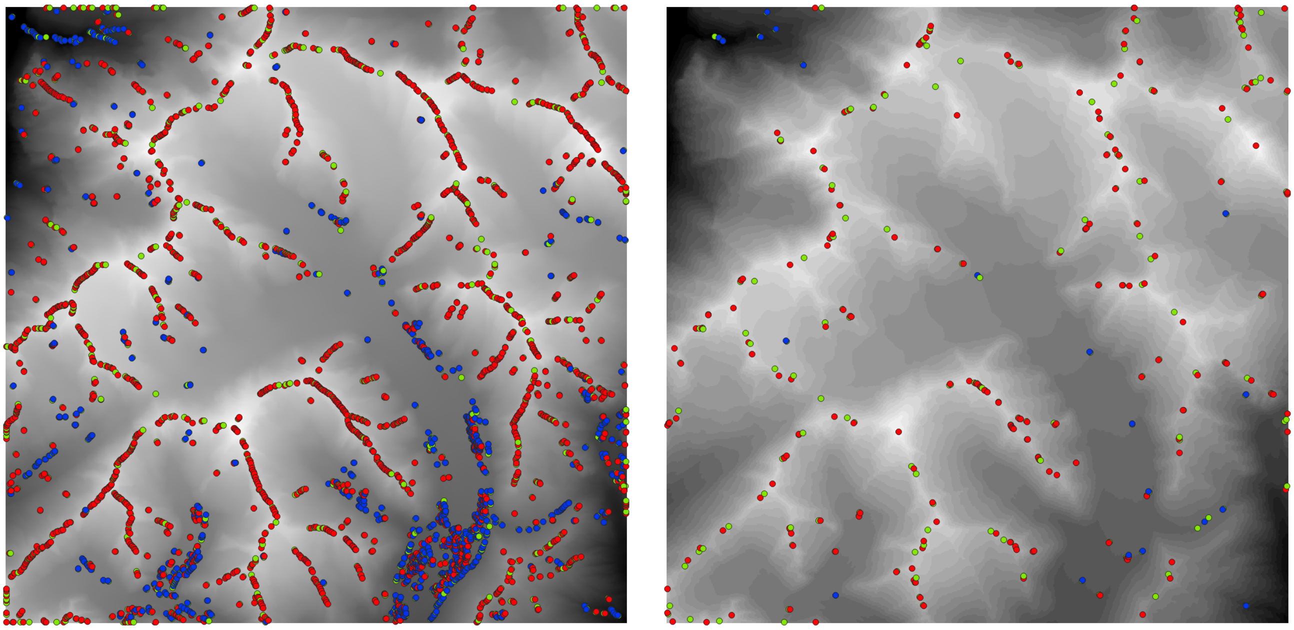


Figure 7: *Aletsch Glacier dataset. Left: the original dataset and its features, with flat areas covering only* 18.10%*) of the surface. Right: the dataset at the highest quantization level (100), with recovered critical points. At this level,* 99.97% *of the DEM is covered by flat spots.*

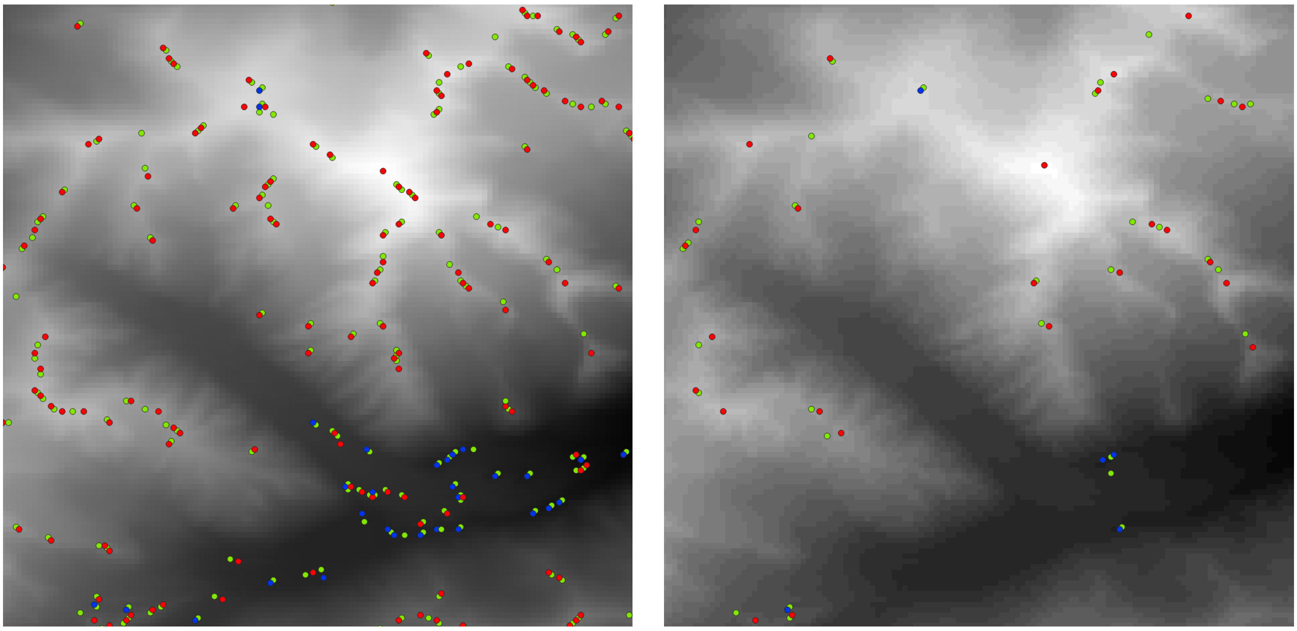


Figure 8: *Graian Alps dataset. Left: the original dataset and its features, with flat areas covering only* 6.10%*) of the surface. Right: the dataset at the highest quantization level (100), with recovered critical points. At this level,* 99.62% *of the DEM is covered by flat spots.*

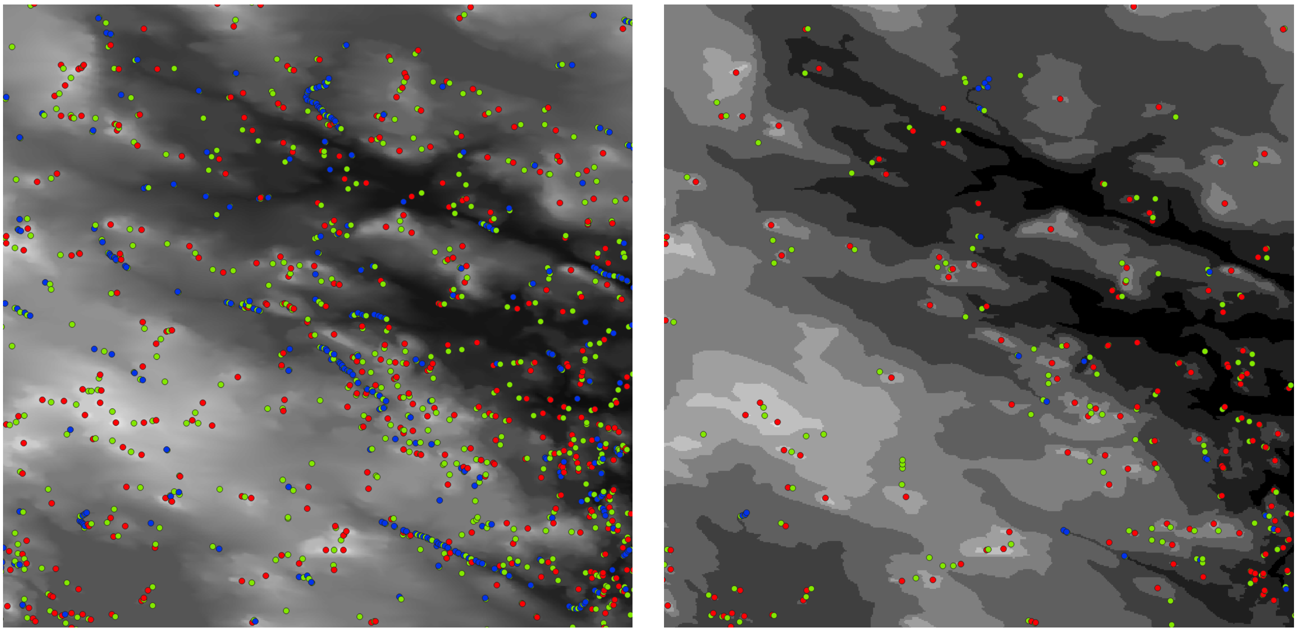


Figure 9: *Finland dataset. Left: the original dataset and its features, with flat areas covering already* 88.51%*) of the surface. Right: the dataset at the highest quantization level (100), with recovered critical points. At this level,* 99.99% *of the DEM is covered by flat spots.*

within a flat spot to maintain the global topology of the underlying function. Finally, we have introduced an algorithm to disambiguate flat spots through a virtual displacement of their points, establishing a strict ordinal relationship among adjacent point elevations. Our algorithm effectively identifies the correct number and type of critical points within each flat spot. We have rigorously tested our algorithms on both synthetic and real datasets at varying levels of quantization, consistently achieving reliable results.

Algorithm 1 will generate different solutions depending on different orderings of valid displacements. We plan to extend our work by investigating whether more complex priorities could give better results, and to perform a quantitative comparison on the quality of recovered critical points. The next logical step will be to compute a new DEM surface by assigning perturbed elevations to the flat points, consistent with their symbolic perturbation. This will provide a super-resolution version of the original DEM that changes only its precision, without affecting its horizontal resolution.

By eliminating flat spots, any grid dataset can be transformed into a discrete Morse function. We propose that the three models we considered, which extend this function to a continuous domain, can be effectively employed to construct a Morse-Smale complex on real data. This approach may potentially mitigate several artifacts encountered with both Piecewise-Linear and discrete Morse models. We will extend our research in this direction in the near future and apply our analysis to real-world application problems.

References

[BDFFP08] BIASOTTI S., DE FLORIANI L., FALCIDIENO B., PAPALEO L.: *Morphological Representations of Scalar Fields*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 185–213. 1

[dF23] DE FERRANTI J.: Viewfinder Panoramas. URL: http://www.viewfinderpanoramas.org/dem3.html, 2023. 6

[DW22] DEY T. K., WANG Y.: *Computational Topology for Data Analysis*. Cambridge University Press, 2022. 1

[DWW17] DEY T. K., WANG J., WANG Y.: Improved road network reconstruction using discrete morse theory. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2017), Association for Computing Machinery. 1

[EH10] EDELSBRUNNER H., HARER J.: *Computational Topology: An Introduction*. American Mathematical Society, 2010. 1, 2, 3, 4

[For98] FORMAN R.: Morse theory for cell complexes. *Advances in Mathematics 134*, 1 (1998), 90–145. 3

[GBP12] GYULASSY A., BREMER P.-T., PASCUCCI V.: Computing morse-smale complexes with accurate geometry. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2014–2022. 3

[GDN*07] GYULASSY A., DUCHAINEAU M., NATARAJAN V., PASCUCCI V., BRINGA E., HIGGINBOTHAM A., HAMANN B.: Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1432–1439. 1

[GSW12] GÜNTHER D., SEIDEL H.-P., WEINKAUF T.: Extraction of dominant extremal structures in volumetric data using separatrix persistence. *Comput. Graph. Forum 31*, 8 (2012), 2554–2566. 1

[JAS16] JASP: Advanced land observing satellite, 2016. URL: https://www.eorc.jaxa.jp/ALOS/en/dataset/aw3d_e.htm. 2

[Mat02] MATSUMOTO Y.: *An Introduction to Morse Theory*, vol. 208 of *Translations of Mathematical Monographs*. American Mathematical Society, 2002. 2

[MDFI13] MAGILLO P., DE FLORIANI L., IURICICH F.: Morphologically-aware elimination of flat edges from a tin. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2013), Association for Computing Machinery, p. 244–253. 2

[NAS00] NASA: Shuttle radar topography mission, 2000. URL: https://www.earthdata.nasa.gov/sensors/srtm. 2, 6

[NGS*08] NARDI F., GRIMALDI S., SANTINI M., PETROSELLI A., UBERTINI L.: Hydrogeomorphic properties of simulated drainage patterns using digital elevation models: the flat area issue / propriétés hydrogéomorphologiques de réseaux de drainage simulés à partir de modèles numériques de terrain: la question des zones planes. *Hydrological Sciences Journal 53*, 6 (2008), 1176–1193. 2

[NJ19] NASA, JASP: Advanced spaceborn thermal emission and reflection radiometer, 2019. URL: https://asterweb.jpl.nasa.gov/gdem.asp. 2

[OSPGT20] OLEJNICZAK M., SEVERO PEREIRA GOMES A., TIERNY J.: A topological data analysis perspective on noncovalent interactions in relativistic calculations. *International Journal of Quantum Chemistry 120*, 8 (2020), e26133. 1

[RB19] RABADAN R., BLUMBERG A. J.: *Topological Data Analysis for Genomics and Evolution: Topology in Biology*. Cambridge University Press, 2019. 1

[RGH*12] REININGHAUS J., GÜNTHER D., HOTZ I., WEINKAUF T., SEIDEL H. P.: Combinatorial gradient fields for 2d images with empirically convergent separatrices, 2012. URL: https://arxiv.org/abs/1208.6523, arXiv:1208.6523. 3

[RJP17] ROCCA L., JENNY B., PUPPO E.: A continuous scale-space method for the automated placement of spot heights on maps. *Computers & Geosciences* (2017). 1

[RWS11] ROBINS V., WOOD P. J., SHEPPARD A. P.: Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 8 (2011), 1646–1658. 2, 3

[SL22] SKAF Y., LAUBENBACHER R.: Topological data analysis in biomedicine: A review. *Journal of Biomedical Informatics 130* (2022), 104082. 1

[swi15] SWISSTOPO: DHM25 Basis Model and Matrix Model. URL: http://www.swisstopo.admin.ch/internet/swisstopo/en/home/products/height/dhm25.html, 2015. 6

[SWTH07] SAHNER J., WEINKAUF T., TEUBER N., HEGE H.-C.: Vortex and strain skeletons in eulerian and lagrangian frames. *IEEE Trans. on Visualization and Computer Graphics 13*, 5 (2007), 980–990. 1

[TAW24] THANH S. L., ANKELE M., WEINKAUF T.: Revisiting accurate geometry for morse-smale complexes, 2024. URL: https://arxiv.org/abs/2409.05532, arXiv:2409.05532. 3, 4

[Tie17] TIERNY J.: *Topological Data Analysis for Scientific Visualization*. Springer, 2017. 1

[VMN*19] VALSANGKAR A. A., MONTEIRO J. M., NARAYANAN V., HOTZ I., NATARAJAN V.: An exploratory framework for cyclone identification and tracking. *IEEE Transactions on Visualization and Computer Graphics 25*, 3 (2019), 1460–1473. 1

[WG09] WEINKAUF T., GÜNTHER D.: Separatrix persistence: extraction of salient edges on surfaces using topological methods. In *Proceedings of the Symposium on Geometry Processing* (Goslar, DEU, 2009), Eurographics Association, p. 1519–1528. 1

[XIDF20] XU X., IURICICH F., DE FLORIANI L.: A persistence-based approach for individual tree mapping. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2020), Association for Computing Machinery, p. 191–194. 1