I/O-Efficient Algorithms for Computing Contour Maps on Terrains

Pankaj K. Agarwal[*]
CS Department
Duke University
Durham, NC
pankaj@cs.duke.edu

Lars Arge[†]
MADALGO[‡]
CS Department
University of Aarhus
Aarhus, Denmark
large@daimi.au.dk

Thomas Mølhave[§]
MADALGO
CS Department
University of Aarhus
Aarhus, Denmark
thomasm@daimi.au.dk

Bardia Sadri[¶]
CS Department
University of Toronto
Toronto, ON, Canada
sadri@cs.toronto.edu

Abstract

A terrain $\mathbf{M}$ is the graph of a continuous bivariate function. We assume that $\mathbf{M}$ is represented as a triangulated surface with N vertices. A *contour* (or *isoline*) of $\mathbf{M}$ is a connected component of a level set of $\mathbf{M}$. Generically, each contour is a closed polygonal curve; at "critical" levels these curves may touch each other or collapse to points. We present I/O-efficient algorithms for the following two problems related to computing contours of $\mathbf{M}$:

(i) Given a sequence $\ell_1 < \cdots < \ell_s$ of real numbers, we present an I/O-optimal algorithm that reports all contours of $\mathbf{M}$ at heights $\ell_1, \ldots, \ell_s$ using $O(\textsc{sort}(N) + T/B)$ I/Os, where T is the total number of edges in the output contours, B is the "block size," and $\textsc{sort}(N)$ is the number of I/Os needed to sort N elements. The algorithm uses $O(N/B)$ disk blocks. Each contour is generated individually with its composing segments sorted in clockwise or counterclockwise order. Moreover, our algorithm generates information on how the contours are nested.

(ii) We can preprocess $\mathbf{M}$, using $O(\textsc{sort}(N))$ I/Os, into a linear-size data structure so that all contours at a given height can be reported using $O(\log_B N + T/B)$ I/Os, where T is the output size. Each contour is generated individually with its composing segments sorted in clockwise or counterclockwise order.

[*]Supported by NSF under grants CNS-05-40347, CFF-06-35000, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and W911NF-07-1-0376, by an NIH grant 1P50-GM-08183-01, by a DOE grant OEG-P200A070505, and by a grant from the U.S.–Israel Binational Science Foundation. Supported by NSF under grants EIA-01-31905, CCR-02-04118, and DEB-04-25465, by an ARO grant W911NF-04-1-0278, and by a grant from the U.S.–Israel Binational Science Foundation.

[†]Supported in part by the US Army Research Office through grant W911NF-04-01-0278, by an Ole Roemer Scholarship from the Danish National Science Research Council, a NABIIT grant from the Danish Strategic Research Council, and by the Danish National Research Foundation.

[‡]Center for Massive Data Algorithmics — a Center of the Danish National Research Foundation.

[§]Supported in part by an Ole Roemer Scholarship from the Danish National Science Research Council, a NABIIT grant from the Danish Strategic Research Council, and by the Danish National Research Foundation.

[¶]Work performed primarily when this author was at Duke University and supported by NSF under grants CNS-05-40347, CFF-06-35000, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and W911NF-07-1-0376, by an NIH grant 1P50-GM-08183-01, by a DOE grant OEG-P200A070505, and by a grant from the U.S.–Israel Binational Science Foundation. Supported by NSF under grants EIA-01-31905, CCR-02-04118, and DEB-04-25465, by an ARO grant W911NF-04-1-0278, and by a grant from the U.S.–Israel Binational Science Foundation.

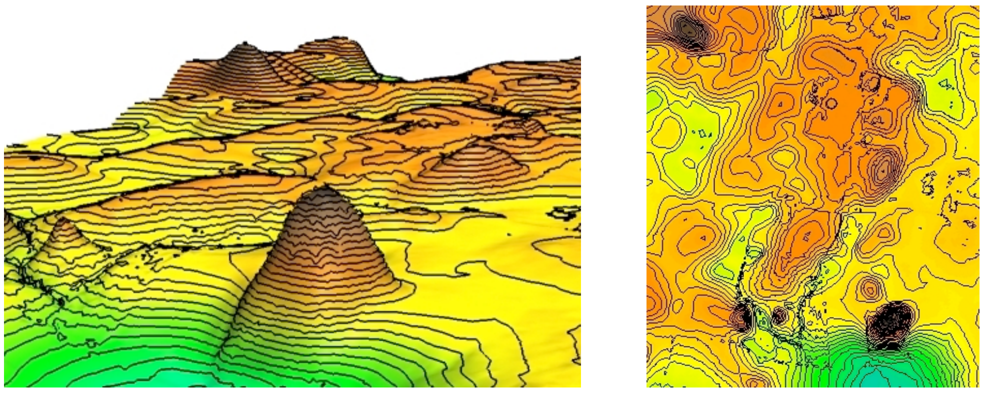
(a) (b)

Figure 1: Examples of equidistant contours of a terrain. (a) Rendered on a perspective view of the terrain in 3d. (b) Projected onto the 2d plane.

1 Introduction

Motivated by a wide range of applications, there is extensive work in many research communities on modeling, analyzing, and visualizing terrain data. A three-dimensional (digital elevation) model of a terrain is often represented as a two-dimensional uniform grid, with a height associated with every grid point, or a triangulated xy-monotone surface; the latter is also known as triangulated irregular network (TIN). A *contour* (or *isoline*) of a terrain $\mathbf{M}$ is a connected component of a level set of $\mathbf{M}$. *Contour maps* (aka *topographic maps*), consisting of contour lines at regular height intervals, are widely used to visualize a terrain and to compute certain topographic information of a map; this representation goes back to at least the eighteenth century [19]. In this paper we propose efficient algorithms for computing contour maps as well as computing contours at a given level.

With the recent advances in mapping techniques and equipment, such as the laser based LIDAR technology, hundreds of millions of points on a terrain, at sub-meter resolution with very high accuracy (∼10-15 cm), can be acquired in a short period of time. The terrain models generated from these data sets are too large to fit in main memory and thus reside on disks. Transfer of data between disk and main memory is often the bottleneck in the efficiency of algorithms for processing these massive terrain models (see e.g. [12]). We are therefore interested in developing efficient algorithms in the two-level I/O-model [3]. In this model, the machine consists of a main memory of size M and an infinite-size disk. A block of B consecutive elements can be transferred between main memory and disk in one *I/O operation* (or simply *I/O*). Computation can only take place on elements in main memory, and the complexity of an algorithm is measured in terms of the number of I/Os it performs. Over the last two decades, I/O-efficient algorithms and data structures have been developed for several fundamental problems, including sorting, graph problems, geometric problems, and terrain modeling and analysis problems. See the recent surveys [4, 24] for a comprehensive review of I/O-efficient algorithms. Here we mention that sorting N elements takes $\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$ I/Os, and we denote this quantity by $\textsc{sort}(N)$.

Related work. A natural way of computing a contour K of a terrain $\mathbf{M}$ is simply to start at one triangle of $\mathbf{M}$ intersecting the contour and then tracing out K by walking through $\mathbf{M}$ until we reach the starting point. If we have a starting point for each contour of a level set of $\mathbf{M}$, for a given level ℓ, we can compute all contours of that level set in time linear on the size of the output in the internal-memory model. The so-called *contour tree* [9] encodes a "seed" for each contour of K. Many efficient internal-memory algorithms are known for computing a contour tree; see e.g. [9]. Hence, one can efficiently construct a contour map of K. This approach of tracing a contour has been extended to higher dimensions as well, e.g., the well-known marching-cube algorithm for computing iso-surfaces [16].

An $O(\textsc{sort}(N))$ algorithm in the I/O-model was recently proposed by Agarwal et al. [2] for constructing a contour tree of $\mathbf{M}$, so one can quickly compute a starting point for each contour. However, it is not clear how to trace a contour efficiently in the I/O-model, since a naive implementation requires $O(T)$ instead of $O(T/B)$ I/Os, to trace a contour of size T. Even using a provably optimal scheme for blocking a planar (bounded degree) graph, so that any path can be traversed I/O-efficiently [1, 17], one can only hope for an $O(T/\log_2 B)$ I/O solution. Nevertheless, I/O-efficient algorithms have been developed for computing contours on a terrain. Chiang and Silva [11] designed a linear-size data structure for storing a TIN terrain $\mathbf{M}$ on disk such that all T edges in the contours at a query level ℓ can be reported in $O(\log_B N + T/B)$ I/Os, but their algorithm does not sort the edges along each contour. Agarwal et al. [1] designed a data structure with the same bounds so that each contour at level ℓ can be reported individually, with its edges sorted in either clockwise or counterclockwise order. However, while the space and query bounds of these structures are optimal, preprocessing them takes $O(N \log_B N)$ I/Os. This bound is more than a factor of B away from the desired $O(\textsc{sort}(N))$ bound. Thus using this structure one can at best hope for an $O(N \log_B N + T/B)$ I/O algorithm to compute a contour map; here T is the total size of all the output contours. We refer the reader to the tutorial [18] and references therein for a review of practical algorithms for contour and iso-surface extraction problems, and to [12, 15] for a sample of I/O-efficient algorithms for problems arising in terrain modeling and analysis.

Our results. Let $\mathbf{M}$ be a terrain represented as a triangulated surface (TIN) with N vertices. For a contour K of $\mathbf{M}$, let $F(K)$ denote the set of triangles intersecting K. We prove (in Section 3) that there exists a total ordering '$\lhd$' on the triangles of $\mathbf{M}$ that has the following two crucial properties:

(C1) For any contour K, if we visit the triangles of $F(K)$ in $\lhd$ order, we visit them along K in either clockwise or counter clockwise order.

(C2) For any two contours K_1 and K_2 on the same level set of $\mathbf{M}$, $F(K_1)$ and $F(K_2)$ are not interleaved in $\lhd$ ordering, i.e., suppose the first triangle of $F(K_1)$ in $\lhd$ appears before that of $F(K_2)$, then either all of the triangles in $F(K_1)$ appear before $F(K_2)$ in $\lhd$, or all triangles of $F(K_2)$ appear between two consecutive triangles of $F(K_1)$ in $\lhd$.

We call such an ordering a *level-ordering* of the triangles of $\mathbf{M}$. We show that $\lhd$ can be computed using $O(\textsc{sort}(N))$ I/Os. Next, we present two algorithms that rely on this ordering.

Computing a contour map. Given as input a sorted list $\ell_1 < \cdots < \ell_s$ of levels in $\mathbb{R}$, we present an algorithm (Section 4) that reports all contours of a terrain $\mathbf{M}$ at levels $\ell_1, \ldots, \ell_s$ using $O(\textsc{sort}(N)+T/B)$ I/Os and $O(N/B)$ blocks of space, where T is the total number of edges in the output contours. Each contour is generated individually with its edges sorted in clockwise or counterclockwise order. Moreover, our algorithm reports how the contours are nested; see Section 4 for details.

Answering a contour query. We can preprocess $\mathbf{M}$, using $O(\textsc{sort}(N))$ I/Os, into a linear-size data structure so that all contours at a given level can be reported using $O(\log_B N + T/B)$ I/Os, where T is the output size. Each contour is generated individually with its edges sorted in clockwise or counterclockwise order (Section 4.5).

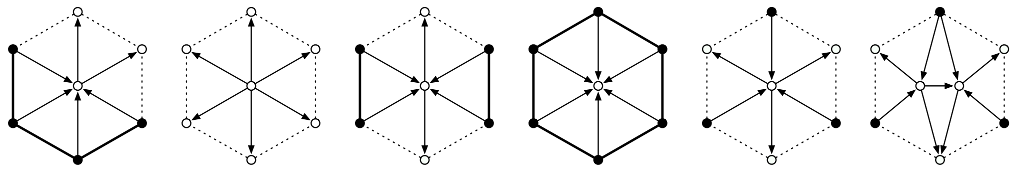

Figure 2: Link of a vertex; lower link is depicted by filled circles and bold edges. The type of a vertex is determined by its lower link.

2 Preliminaries

Let $\mathbb{M} = (V, E, F)$ be a triangulation of $\mathbb{R}^2$, with vertex, edge, and face (triangle) sets V, E, and F, respectively. We assume that V contains a vertex v_∞, set at infinity, and that each edge $\{u, v_\infty\}$ is a ray emanating from u. The triangles in $\mathbb{M}$ incident to v_∞ are unbounded. Let $h : \mathbb{R}^2 \to \mathbb{R}$ be a continuous *height function* with the property that the restriction of h to each triangle of $\mathbb{M}$ is a linear map. Given $\mathbb{M}$ and h, the "graph" of h is a *terrain* $\mathbf{M} = (\mathbb{M}, h)$ which describes an xy-monotone triangulated surface in $\mathbb{R}^3$ whose triangulation is induced by $\mathbb{M}$. That is, vertices, edges, and faces of $\mathbf{M}$ are in one-to-one correspondence with those of $\mathbb{M}$. With a slight abuse of notation, in what follows we write V, E, and F, to respectively refer to the sets of vertices, edges, and triangles of both the terrain $\mathbf{M}$ and its underlying plane triangulation $\mathbb{M}$.

For convenience we assume that $h(u) \neq h(v)$ for all vertices $u \neq v$, and that $h(v_\infty) = -\infty$. Within each bounded triangle $f \in F$, h is uniquely determined as the linear interpolation of the height of the vertices of f. This is not the case for an unbounded face f since interpolation using $h(v_\infty) = -\infty$ is undefined; in which case to determine h on f an extra parameter, such as the height of a point in f, is needed.

For a given terrain $\mathbf{M}$ and a *level* $\ell \in \mathbb{R}$, the *ℓ-level set* of $\mathbf{M}$, denoted by $\mathbb{M}_\ell$, is defined as $h^{-1}(\ell) = \{x \in \mathbb{R}^2 \mid h(x) = \ell\}$. Equivalently, $\mathbb{M}_\ell$ is the vertical projection of $\mathbf{M} \cap z_\ell$ on the xy-plane, where z_ℓ is the horizontal plane $z = \ell$. The *closed ℓ-sublevel* and *ℓ-superlevel sets* of $\mathbf{M}$ are defined respectively as $\mathbb{M}_{\leq \ell} = h^{-1}((-\infty, \ell])$ and $\mathbb{M}_{\geq \ell} = h^{-1}([\ell, +\infty))$, and the *open ℓ-sublevel* and *ℓ-superlevel sets* $\mathbb{M}_{<\ell}$ and $\mathbb{M}_{>\ell}$ are $\mathbb{M}_{\leq \ell} \setminus \mathbb{M}_\ell$ and $\mathbb{M}_{\geq \ell} \setminus \mathbb{M}_\ell$, respectively. For any $R \subseteq \mathbb{R}^2$, let $\mathbf{M}(R)$ denote the subset of the surface $\mathbf{M}$ whose vertical projection into the xy-plane is R, i.e. $\mathbf{M}(R) = \{(x, y, z) \in \mathbf{M} : (x, y) \in R\}$. We shall also use the shorthand notations of $\mathbf{M}_\ell$, $\mathbf{M}_{<\ell}$, etc, for $\mathbf{M}(\mathbb{M}_\ell)$, $\mathbf{M}(\mathbb{M}_{<\ell})$, etc, respectively.

In much of what follows we need to compare the heights of two neighboring vertices of a terrain $\mathbb{M}$. To simplify the exposition we "orient" each edge of $\mathbb{M}$ toward its *higher* endpoint, and treat $\mathbb{M}$ as a directed triangulation in which a directed edge (u, v) indicates that $h(u) < h(v)$.

The *dual* graph $\mathbb{M}^* = (F^*, E^*, V^*)$ of the triangulation $\mathbb{M}$ is defined as the planar graph that has a vertex $f^* \in F^*$ for each face $f \in F$, called the *dual* of f. For any directed edge $e \in E$, there is a directed *dual edge* $e^* = (f_1^*, f_2^*) \in E^*$ where f_1 and f_2 are the faces to the left and to the right of e respectively. The graph $\mathbb{M}^*$ is naturally embedded in the plane as follows: the vertex f^* is placed inside the face f and e^* is is drawn as a curve that crosses e but no other edges of $\mathbb{M}$. A vertex $v \in V$ leads to a dual face v^* in $\mathbb{M}^*$ that is bounded by the duals of the edges incident to v. The dual of $\mathbb{M}^*$ is $\mathbb{M}$ itself. For a given subset V_0 of V, we use the notation V_0^* to refer to the set of duals to the vertices in V_0, i.e., $V_0^* = \{v^* : v \in V_0\}$. A similar notation is also used for subsets of F or E.

Links and critical points. For a vertex v of $\mathbb{M}$, the *link* of v, denoted by $\mathrm{Lk}(v)$, is the cycle in $\mathbb{M}$ consisting of the vertices adjacent to v, as joined by the edges from the triangles incident upon v.

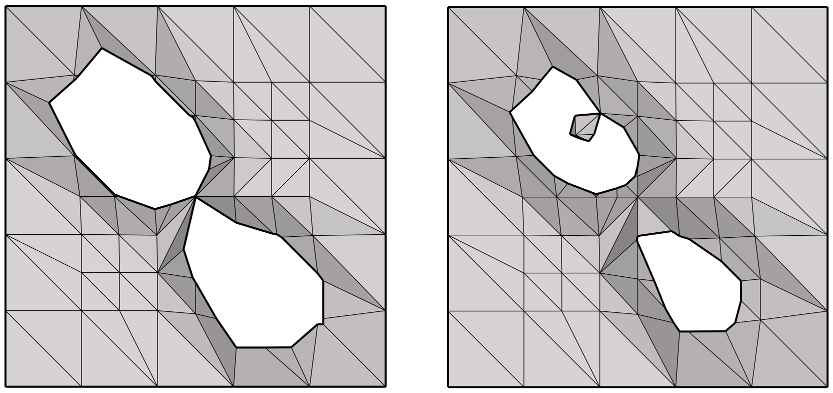

Figure 3: Examples of sublevel sets of negative (left) and positive (right) saddle points.

The *lower link* of v, $\mathrm{Lk}^-(v)$, is the subgraph of $\mathrm{Lk}(v)$ induced by vertices lower (of smaller height) than v. The *upper link* of v, $\mathrm{Lk}^+(v)$ is defined analogously; see Figure 2.

If a level parameter ℓ varies continuously along the real line, the topology of $\mathbb{M}_{\leq \ell}$ changes only at a discrete set $\{\ell_1, \ldots, \ell_m\}$ of *critical levels* of h, where each ℓ_i is $h(v_i)$ for some vertex $v_i \in V$. $v_1, \ldots, v_m$ are *critical vertices* of $\mathbb{M}$. A non-critical level of h is also called *regular*. Vertices with regular heights are *regular vertices*. By our assumption that the height of every vertex is distinct, there is only one critical vertex at each critical level.

There are three types of critical vertices: *minima*, *saddles*, and *maxima*. The type of a vertex v can be determined from the topology of $\mathrm{Lk}^-(v)$: v is minimum, regular, saddle, or maximum if $\mathrm{Lk}^-(v)$ is empty, a path, two or more paths, or a cycle, respectively. We assume that all saddles are *simple*, meaning that the lower link of each saddle consists of precisely two paths. Multifold saddles can be split symbolically into simple saddles; see Figure 2. Equivalently, a vertex can be classified based on the clockwise ordering of its incoming and outgoing edges: a minimum has no incoming edges, and a maximum has no outgoing edges. For other vertices v, we count the number of times incident edges switch between incoming to outgoing as we scan them around v in clockwise order. This number is always even. Two switches indicate that v is regular while four or more switches take place if v is a saddle.

A saddle vertex v is further classified into two types. At $\ell = h(v)$ the topology of $\mathbb{M}_{\leq \ell}$ differs from that of $\mathbb{M}_{< \ell}$ in one of two possible ways: either two connected components of $\mathbb{M}_{< \ell}$ join at v to become the same connected component in $\mathbb{M}_{\leq \ell}$, or the boundary of the same connected component of $\mathbb{M}_{< \ell}$ "pinches" at v introducing one more "hole" in $\mathbb{M}_{\leq \ell}$. Saddles of the former type are *negative* saddles and those of the latter type are *positive* saddles; see Figure 3. It is well-known that the number of minima (resp. maxima) is one more than the number of negative (resp. positive) saddles, and therefore

$$\#\text{saddles} = \#\text{minima} + \#\text{maxima} - 2. \tag{1}$$

This classification of saddles is related to persistent homology and a more general statement is proved in [13].

Contours. A *contour* of a terrain $\mathbf{M}$ is a connected component of a level set of $\mathbf{M}$. Each contour K at a regular level is a simple closed curve and partitions $\mathbb{R}^2 \setminus K$ into two open sets: a bounded one called *inside* of K and denoted by K^{i}, and an unbounded one called *outside* of K and denoted by K^{o}. This is violated at critical levels at which a contour may shrink into a point (an extremum), or may consists of two simple closed curves whose intersection is the critical point (a saddle). When the level parameter ℓ scans the open interval between two consecutive critical values of h

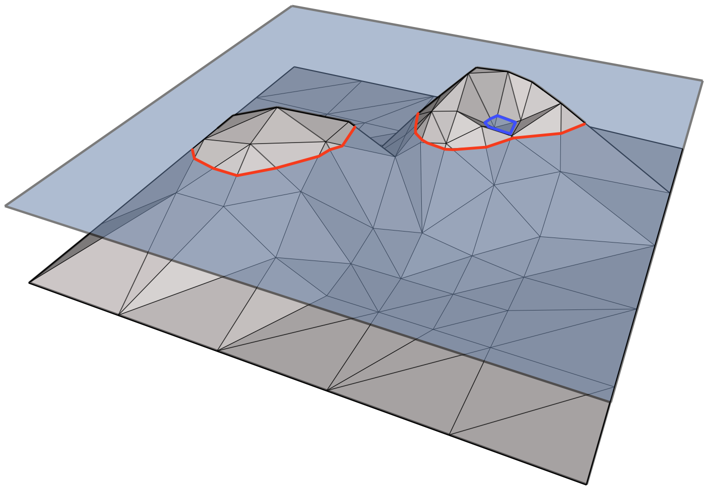

Figure 4: Red and blue contours in a level-set of a terrain.

the contours of $\mathbb{M}_\ell$ change continuously and the topology of $\mathbb{M}_\ell$ remains unchanged. However, at a critical level the contours that contain the corresponding critical point undergo topological changes. Let K_1 and K_2 be two contours at levels ℓ_1 and ℓ_2 respectively with $\ell_1 < \ell_2$. We regard K_1 and K_2 as "the same" if K_1 continuously deforms into K_2 (without any topological changes), as z_ℓ sweeps $\mathbb{M}$ in the interval $[\ell_1, \ell_2]$.

Following [1], we call a contour K in $\mathbb{M}_\ell$ *blue* if, "locally", $\mathbb{M}_{<\ell}$ lies in K^{i}, and *red* otherwise; see Figure 4. Every blue contour is born as a single point at a minimum. Conversely, a blue contour is born at every minimum except at v_∞. Because of being placed at infinity, a red contour is born at v_∞. Likewise, a red contours "dies" by shrinking into a single point at a maximum, and conversely, some red contour dies at every maximum. Two contours, with at least one of them being blue, merge into the same contour at a negative saddle. The resulting contour is red if one of the merging contours is red, and blue otherwise. A contour splits into two contours at a positive saddle. A red contour splits into two red contours while a blue contour splits into one red and one blue contour.

Two contours K_i and K_j of a level set $\mathbb{M}_\ell$ are called *neighbors* if no other contour K of $\mathbb{M}_\ell$ separates them, i.e., one of K_i and K_j is contained in K^{i} and the other in K^{o}. If K_i is neighbor to K_j and $K_i \subset K_j^{\mathrm{i}}$, then K_i is called a *child* of K_j. If $K_i \subset K_j^{\mathrm{o}}$ and $K_j \subset K_i^{\mathrm{o}}$ then K_i is called a *siblings* of K_j. It can be verified that all children of a red (resp. blue) contour are blue (resp. red) contours while all siblings of a red (resp. blue) contour are red (resp. blue) contours.

We conclude this section by making a key observation, which is crucial for our main result. Each regular contour of $\mathbf{M}$ corresponds to a cycle in $\mathbb{M}^*$: let K be a contour in an arbitrary level set $\mathbb{M}_\ell$, and let $F(K)$ (resp. $E(K)$) denote the set of faces (resp. edges) of $\mathbb{M}$ that intersect K. If K is a red (resp. blue) contour, all the edges in $E(K)$ are oriented toward K^{i} (resp. K^{o}). Consequently, the vertices in $F^*(K)$ are linked by the edges in $E^*(K)$ into a cycle in $\mathbb{M}^*$. We refer to this cycle as the *representing cycle of* K. We use $C(K, \mathbb{M})$ to denote the circular sequence of triangles dual to the representing cycle of K in $\mathbb{M}$. The sequence in $C(K, \mathbb{M})$ is oriented clockwise (resp. counterclockwise) if K is blue (resp. red); see Figure 5.

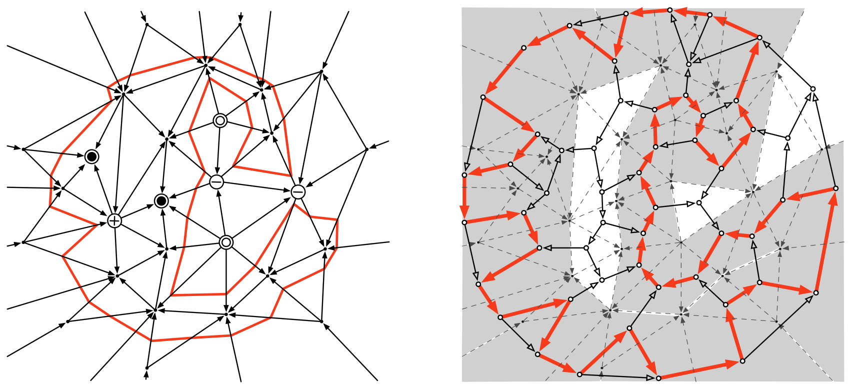
(a) (b)

Figure 5: (a) Orientation of the edges in the plane triangulation $\mathbb{M}$ of a terrain. Critical points and a contour K in a regular level set are shown. (b) the dual $\mathbb{M}^*$ of $\mathbb{M}$. The representing cycle of K in $\mathbb{M}^*$ is shown with bold edges. Triangles in $C(K, \mathbb{M})$ are shaded.

3 Level-ordering of Triangles

In this section we present our main result, i.e. the existence of a level-ordering on triangles of any terrain $\mathbf{M}$, i.e., an ordering that satisfies conditions (C1) and (C2). We begin by proving the existence of a level-ordering for terrains that do not have saddle vertices. We call such terrains *basic*. Next we prove certain structural properties of terrains and show that any arbitrary terrain can be transformed into a basic terrain through a *surgery* that effectively "preserves" the contours of the original terrain. We then argue that a level-ordering on the transformed terrain corresponds to a level-ordering on the original one.

3.1 Basic terrain

Let $\mathbf{M}$ be a basic terrain. The above discussion and (1) imply that $\mathbf{M}$ has one (global) minimum, $\check{v}$, which coincides with v_∞, and one (global) maximum, $\hat{v}$, and that every level set consists of a single red contour. At $\hat{v}$ this contour collapses into a single point.

Lemma 3.1 *Let $P \subset E$ be a directed (monotone) path in $\mathbb{M}$ from $\check{v}$ to $\hat{v}$. Then every cycle of $\mathbb{M}^*$ contains exactly one edge from P^*. In particular, the graph $\mathbb{M}^* \setminus P^*$ obtained from deleting the edges in P^* from $\mathbb{M}^*$ is acyclic.*

Proof. We claim that $\hat{v}$ is reachable in $\mathbb{M}$ from every vertex $v \in V$. Recall that $\hat{v}$ is the only local maximum in $\mathbb{M}$ and that every other vertex has at least one outgoing edge. If one starts at v and follows an arbitrary outgoing edge at each step, the height of the vertex at which we arrive is greater than that of the previous one. This process can only stop at $\hat{v}$. By a similar argument, every vertex $v \in V$ is reachable from $\check{v}$.

Consider an arbitrary cycle C^* in $\mathbb{M}^*$. In the plane drawing of $\mathbb{M}^*$, C^* is a Jordan curve. Let $V_0 \subset V$ be the set of vertices that are contained in the inside of C^* (equivalently, $V_0^* \subset V^*$ is the set of faces of $\mathbb{M}^*$ whose union is bounded by C^*). Let $C \subset E$ be the set of edges in $\mathbb{M}$ dual to those in C^*. Since C^* is a cycle, the edges in C are either all oriented from V_0 to $V \setminus V_0$ or all from $V \setminus V_0$ to V_0.

The former case cannot happen because $v_\infty \notin V_0$ and every vertex in V is reachable from v_∞. If all edges of C are oriented from $V \setminus V_0$ to V_0, then $\hat{v} \in V_0$ because otherwise $\hat{v}$ cannot be reachable from the vertices of V_0. Since $\hat{v} \in V_0$ and $v_\infty \in V \setminus V_0$, $|P \cap C| \geq 1$. There is no edge directed from V_0 to $V \setminus V_0$, so once P reaches a vertex of V_0 it cannot leave V_0, implying that $|P \cap C| = 1$. Thus, every cycle of $\mathbb{M}^*$ is destroyed by the removal of the edges in P^*, implying that $\mathbb{M}^* \setminus P^*$ is acyclic. ■

Let P be the path from $\check{v}$ to $\hat{v}$ as defined in Lemma 3.1. The graph $\mathbb{M}^* \setminus P^*$ has all of the vertices of $\mathbb{M}^*$. Thus every face f of $\mathbb{M}$ is represented by f^* in $\mathbb{M}^* \setminus P^*$. Let $\prec$ be the a binary relation on F (triangles in $\mathbb{M}$) defined as $f_1 \prec f_2$ if $(f_1^*, f_2^*) \in E^* \setminus P^*$. Since by Lemma 3.1 $\mathbb{M}^* \setminus P^*$ is acyclic, $\prec$ is a partial order on F. We call $\prec$ the *adjacency partial order* induced by the acyclic graph $\mathbb{M}^* \setminus P^*$. A linear extension of $\prec$ is any total order $\lhd$ on F that is consistent with $\prec$, i.e. $f_1 \prec f_2$ implies $f_1 \lhd f_2$. Such a linear extension can be obtained by topological sorting of $\mathbb{M}^* \setminus P^*$. By definition, the existence of a directed path from f_i^* to f_j^* in $\mathbb{M}^* \setminus P^*$ implies that $f_i \lhd f_j$. Thus condition (C1) of the definition of level ordering holds for $\lhd$. Since in a basic terrain each level consists of only one contour, condition (C2) holds trivially. This results the following statement.

Corollary 3.2 *Let* $\mathbf{M}$ *be a basic terrain, and let* P *be a directed (monotone) path from* $\check{v}$ *to* $\hat{v}$ *in* $\mathbb{M}$*. Let* $\lhd$ *be a linear extension of the adjacency partial order induced by* $\mathbb{M}^* \setminus P^*$*. Then* $\lhd$ *is a level-ordering of the triangles of* $\mathbf{M}$*.*

3.2 Red and blue cut-trees

Consider now a non-basic terrain with saddle vertices. We first introduce the notions of *ascending (red)* and *descending (blue) cut-trees* of $\mathbb{M}$ as subgraphs of the triangulation $\mathbb{M}$, which we later use to turn $\mathbf{M}$ into a basic terrain $\tilde{\mathbf{M}}$. Contours of each level set of $\mathbf{M}$ will then be encoded in a corresponding level set of $\tilde{\mathbf{M}}$ which consists of a single contour.

A *descending* (resp. *ascending*) path on $\mathbb{M}$ from a vertex $v \in V$ is a path $v_0, v_1, \ldots, v_r$ where $v_0 = v$ and $h(v_i) < h(v_{i-1})$ (resp. $h(v_i) > h(v_{i-1})$) for $i = 1, \ldots, r$. For each negative saddle v, let $P_1(v) = u_0, u_1, \ldots, u_r$ and $P_2(v) = w_0, \ldots, w_s$ be two descending paths from v such that u_r and w_s are both minima and u_1 and w_1 belong to different connected components of $\mathrm{Lk}^-(v)$. Furthermore assume that for any two negative saddles u and w, if $P_i(u) = u_0, \ldots, u_r$ and $P_j(w) = w_0, \ldots, w_s$, for some $i, j \in \{1, 2\}$, and $u_k = w_l$ for some $1 \leq k \leq r$ and $1 \leq l \leq s$, then $u_{k+1} = w_{l+1}$; in other words, descending paths from different vertices can join but then cannot diverge. Such a set of paths always exist: one can assign such paths to negative saddles in the increasing order of their heights. At any negative saddle u, we follow a descending paths through each of the two connected components of $\mathrm{Lk}^-(u)$ until it either reaches a minimum or joins a path already assigned to a lower negative saddle. Let $P(u) = P_1(u) \cup P_2(u)$ for any negative saddle u. Since $P_1(u) \setminus \{u\}$ and $P_2(u) \setminus \{u\}$ are contained in different connected components of $\mathbb{M}_{<h(u)}$, the underlying undirected graph of $P(u)$ is a simple path. For a positive saddle u, $P_1(u)$ and $P_2(u)$ are defined similarly using ascending paths that start at different connected components of $\mathrm{Lk}^+(u)$ and end in maxima.

We define the *descending (blue) cut-tree* $\check{\mathbb{T}} = (\check{V}, \check{E})$ of $\mathbf{M}$ to be the union of the paths $P(u)$ over all negative saddles u. Similarly, we define the *ascending (red) cut-tree* $\hat{\mathbb{T}} = (\hat{V}, \hat{E})$ of $\mathbf{M}$ to be the union of all the paths $P(u)$ over all positive saddles u. It is, of course, not clear that $\check{\mathbb{T}}$ and $\hat{\mathbb{T}}$ are trees but this and some of their other properties are proven below.

Remark. The definitions of red and blue cut-trees are closely related to the notions of *split* and *join trees* in the context of *contour trees* [21]. Intuitively the contour tree is the result of contracting each contour of $\mathbf{M}$ into a single point [1] and is topologically a connected collection of simple curves

[1]Taking the terrain $\mathbf{M}$ as a topological space with the usual topology of $\mathbb{R}^2$ and defining an equivalence relation $\sim$ between points on $\mathbf{M}$ as $x \sim y$ if and only if x and y are on the same contour (connected component of some level set), the contour tree $\mathbf{M}_\sim$ of $\mathbf{M}$ is the quotient space of $\mathbf{M}$ modulo $\sim$.

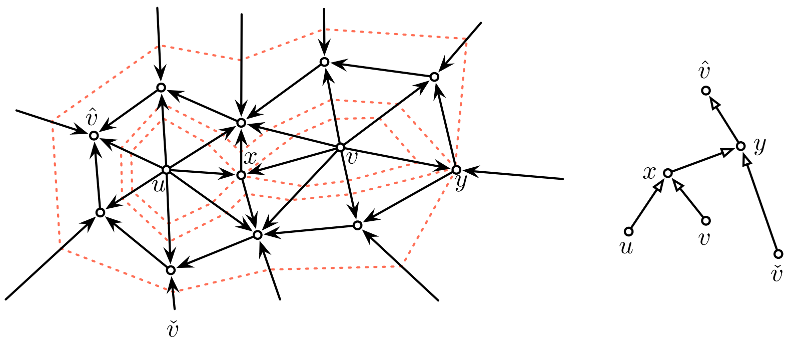

Figure 6: A terrain for which the join tree cannot be embedded as a subgraph of the underlying triangulation in such a way that the edges are realized by ascending paths. Level sets of saddles are depicted in dotted lines. The contour tree of the terrain on the left is shown on the right. Notice that there is no directed (ascending) path from x to y on the terrain.

whose endpoints correspond to the critical points of $\mathbf{M}$. These curves can only intersect at their endpoints and thus realize the edges of a graph on a set of vertices that correspond to the critical points of $\mathbf{M}$. It can be shown that this graph is always connected and acyclic — hence the name contour tree.[2] Each point on a curve realizing an edge represents a contour at some height. The heights of points along any edge of the contour tree vary monotonically from one end to the other. The contour tree is often described as the union of two of its subtrees, namely the merge and split trees. The join tree is the minimal connected subtree of the contour tree that contains minima and negative saddles and the split tree is defined analogously using maxima and positive saddles.

The similarity between the notions of blue (resp. red) cut-tree and join (resp. split) tree naturally poses the question of whether our cut-trees can be replaced by their contour tree counterparts. We emphasize here that our cut-trees are subgraphs of the triangulation $\mathbb{M}$ and this plays a crucial role in our algorithms. It is possible to draw the contour tree on the terrain in such a way that the vertices coincide with their corresponding critical points and edges are realized by monotonically ascending curves on the terrain. It is easy to observe that if one can realize each edge of the join or split tree as a monotonically ascending path in $\mathbb{M}$ then it is indeed possible to simply use the merge or split trees in place of our cut-trees. However, this is not always possible as the terrain depicted in Figure 6 demonstrates.

Lemma 3.3 *The underlying undirected graph of a blue (resp. red) cut-tree $\check{\mathbb{T}}$ (resp. $\hat{\mathbb{T}}$) has no cycles.*

Proof. We prove the claims for blue cut-tree. The argument for red cut-tree can be made symmetrically. Let $u_1, \ldots, u_r$ be the list of all negative saddles of $\mathbb{M}$ in the increasing order of their heights. Let $\check{\mathbb{T}}_0$ be the empty graph and for each $i = 1, \ldots, r$, let $\check{\mathbb{T}}_i = \bigcup_{j=1}^{i} P(u_j)$; $\check{\mathbb{T}}_{i-1}$ is a subgraph of $\check{\mathbb{T}}_i$, and $\check{\mathbb{T}}_r = \check{\mathbb{T}}$. We prove by induction on i that the underlying undirected graph of each $\check{\mathbb{T}}_i$ is a forest. This trivially holds for $\check{\mathbb{T}}_0$. Assume now that the underlying undirected graph of $\check{\mathbb{T}}_i$ is a forest. By construction, adding $P(u_{i+1})$ connects two distinct connected components of $\check{\mathbb{T}}_i$, one contained in each of the two distinct connected components of $\mathbb{M}_{<h(u_{i+1})}$ that join at u_{i+1}.

Moreover, once each of $P_1(u_{i+1})$ or $P_2(u_{i+1})$ reaches a vertex of $\check{\mathbb{T}}_i$, it continues by following a

[2]Indeed such graphs, known generally as *Reeb graphs* [22], can be obtained from arbitrary continuous real valued functions defined on more general topological spaces such as arbitrary manifolds. Contour trees are Reeb graphs of terrains as determined by their height functions.

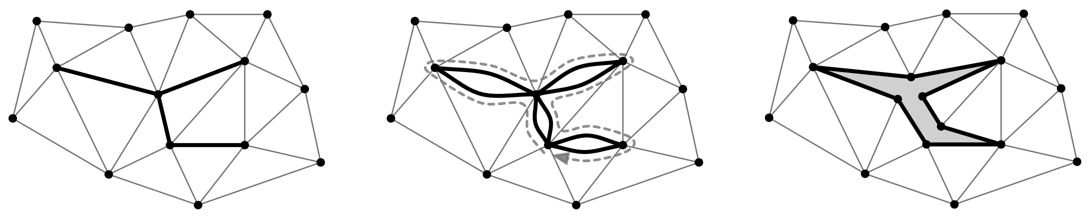
Figure 7: Cutting a triangulation along a tree.

path contained in $\check{\mathbb{T}}_i$, and therefore does not introduce a cycle within the corresponding connected component of $\check{\mathbb{T}}_i$. ■

For a set $U \subseteq \mathbb{R}^2$, let $\check{\mathbb{T}}(U)$ (resp. $\hat{\mathbb{T}}(U)$) be the union of the paths $P(u)$ for all negative (resp. positive) saddles $u \in U$. In particular, $\check{\mathbb{T}} = \check{\mathbb{T}}(\mathbb{R}^2)$ and $\hat{\mathbb{T}} = \hat{\mathbb{T}}(\mathbb{R}^2)$.

Lemma 3.4 *For a blue (resp. red) contour K, the underlying undirected graph $\check{\mathbb{T}}(K^i)$ (resp. $\check{\mathbb{T}}(K^o)$) connects all of the minima in K^i (resp. K^o). A symmetric statement holds regarding $\hat{\mathbb{T}}$ and maxima by switching "red" and "blue".*

Proof. We prove the lemma for $\check{\mathbb{T}}$ and blue contours. The other cases are similar. Let K be a blue contour in $\mathbb{M}_\lambda$ for some $\lambda \in \mathbb{R}$. We show that for each $\ell \in \mathbb{R}$, the minima in each connected component of $U_\ell = \mathbb{M}_{<\ell} \cap K^{\mathrm{i}}$ are connected by $\check{\mathbb{T}}(U_\ell)$. The statement of the lemma then follows by taking ℓ to be larger than the height of all vertices in K^{i} and from the fact that in that case U_ℓ consists of a single component.

To prove the lemma we sweep ℓ from $-\infty$ toward $+\infty$ and verify the claim for U_ℓ. Every time ℓ reaches the height of a minimum in K^{i}, a new connected component is added to U_ℓ. The lemma holds for this new component since it originally has only a single minimum which is vacuously connected by $\check{\mathbb{T}}(U_\ell)$ to every other minimum in that component. The validity of the claim as ℓ continues to raise can only be altered when ℓ reaches the height of a negative saddle u in K^{i} at which two connected components U_1 and U_2 of $U_{<\ell}$, where $\ell = h(u)$, join at u. At this time the path $P(u)$ is added to $\check{\mathbb{T}}(U_\ell)$. The crucial observation here is that because K is a blue contour, no descending path started at a vertex $u \in K^{\mathrm{i}}$ can reach K^{o}. Thus the endpoints of $P(u)$ have to be minima in K^{i}. In other words $P(u)$, which reaches a minimum in U_1 and another in U_2, connects $\check{\mathbb{T}}(U_1)$ and $\check{\mathbb{T}}(U_2)$ as desired. ■

Corollary 3.5 *The underlying undirected graphs of $\check{\mathbb{T}}$ and $\hat{\mathbb{T}}$ are trees. Moreover, all minima are vertices of $\check{\mathbb{T}}$ and all maxima are vertices of $\hat{\mathbb{T}}$.*

We conclude this discussion by mentioning a property of $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$ that follows from their constructions.

Lemma 3.6 *Let u be a vertex of $\hat{\mathbb{T}}$ (resp. $\check{\mathbb{T}}$). If u is a positive (resp. negative) saddle, then u has two outgoing (resp. incoming) edges in $\hat{\mathbb{T}}$ (resp. $\check{\mathbb{T}}$) — one to each connected component of the upper (resp. lower) link of u in $\mathbb{M}$. If u is a regular vertex or a negative saddle, then u has one outgoing (resp. incoming) edge. Finally, if u is a maximum (resp. minimum), then it has no outgoing (resp. incoming) edges.*

3.3 Surgery on terrain

Let $\hat{\mathbb{T}}$ be a red cut-tree for $\mathbf{M}$. Consider the following combinatorial operation on $\mathbb{M}$. First we duplicate every edge e of $\hat{\mathbb{T}}$, thus creating a face f_e that is bounded by the two copies of e. We then

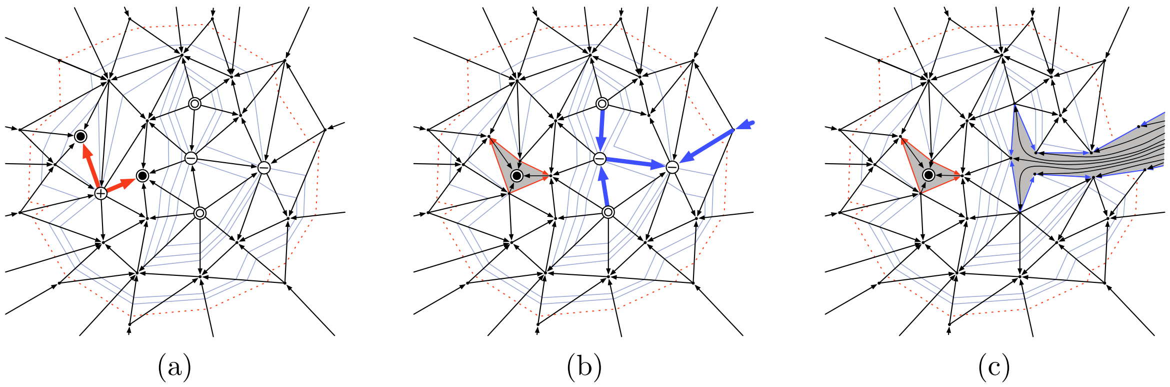
Figure 8: (a) A red (ascending) cut-tree marked $\hat{\mathbb{T}}$ marked on the terrain $\mathbb{M}$ of Figure 5. (b) Construction of the graph $\mathbb{M}_0$: the terrain is cut along $\hat{\mathbb{T}}$ and a new maximum $\hat{v}$ is inserted in the opened face. On the right, a blue cut-tree of $\mathbb{M}_0$ is marked. (c) Construction of the graph $\tilde{\mathbb{M}}$: the terrain is cut open on the red cut-tree and a new maximum is inserted.

perform an Eulerian tour on the subgraph of $\mathbb{M}$ induced by the copies of the edges of $\hat{\mathbb{T}}$ in which at each vertex the next edge of the tour is the first unvisited edge of the subgraph in clockwise order, relative to the previous edge of the tour. We then combine all of the faces f_e into a single face $\hat{f}$ that is bounded by this Eulerian tour by making as many copies of each vertex as its degree in $\hat{\mathbb{T}}$ (or equivalently the number of times the tour has passed through it) and connecting non-tree edges incident on u to appropriate copies of u; see Figure 7. Geometrically, the above modification of the terrain triangulation can be interpreted as "puncturing" the plane along the edges of $\hat{\mathbb{T}}$ and introducing a new face $\hat{f}$ bounded by the $2|\hat{E}|$ edges in the Euler tour.

We then subdivide $\hat{f}$ by placing a new vertex $\hat{v}$ inside it and connecting $\hat{v}$ via incoming edges $(u, \hat{v})$ to every vertex u on the boundary of $\hat{f}$. The result is a triangulation $\mathbb{M}_0 = (V_0, E_0, F_0)$; see Figures 8 (a) and (b). The newly added triangles are all incident to $\hat{v}$, and we refer to them as $\hat{v}$-triangles. The edge e opposite to $\hat{v}$ in a $\hat{v}$-triangle f (which is a copy of a $\hat{\mathbb{T}}$ edge) is called the *base* of f and f is said to be *based* at e. One can modify the plane drawing of $\mathbb{M}$ into a (singular) plane drawing of $\mathbb{M}_0$, that has faces of zero area and edges that bend and overlap, by jamming all the new faces and edges in the (zero-area) hole that results from cutting the plane along $\hat{\mathbb{T}}$.

$\mathbb{M}_0$ can be regarded as the triangulation of a terrain $\mathbf{M}_0$: Fáry's theorem [14] can be used to straight-line embed $\mathbb{M}_0$ while preserving all its faces and the height function of $\mathbf{M}$ induces a height function on triangles of $\mathbb{M}_0$ that are also in $\mathbb{M}$. The height of $\hat{v}$ is then chosen to be higher than the heights of all vertices of $\mathbf{M}$ and is used to linearly interpolate a height function on $\hat{v}$-triangles.

Lemma 3.7 *$\mathbf{M}_0$ has no positive saddles and exactly one maximum, namely $\hat{v}$. The minima of $\mathbf{M}_0$ are precisely those of $\mathbf{M}$. Each negative saddle of $\mathbf{M}_0$ is a copy of a negative saddle of $\mathbf{M}$, and only one copy of each negative saddle of $\mathbf{M}$ is a negative saddle of $\mathbf{M}_0$.*

Proof. For a vertex $u \notin \hat{\mathbb{T}}$, $\mathrm{Lk}(u)$ is the same in $\mathbb{M}$ and $\mathbb{M}_0$ modulo taking copies of $\hat{\mathbb{T}}$ vertices as identical. In particular, minima of $\mathbf{M}$ stay minima in $\mathbf{M}_0$. Thus it suffices to consider $\hat{v}$ and copies of $\hat{\mathbb{T}}$ vertices. Clearly, $\hat{v}$ is a maximum. Let u be a vertex of $\hat{\mathbb{T}}$, and let u' be a copy of u in $\mathbb{M}_0$. Let e_1' and e_2' be copies of $\hat{\mathbb{T}}$ edges that enter and leave u', respectively, in the Eulerian tour of $\hat{\mathbb{T}}$. Both of these edges remain incident to u' in $\mathbb{M}_0$. Let v_1' and v_2' be other endpoints of e_1' and e_2' in $\mathbb{M}_0$, respectively. Let v_i and e_i, $i = 1, 2$, be the vertex and edge in $\mathbb{M}$ corresponding to v_i' and e_i', respectively. $\mathrm{Lk}(u')$ consists of a path $\pi(u')$ from v_1' to v_2' followed by $\hat{v}$. Moreover, e_1' and e_2' are the only edges incident on u' that are copies of $\hat{\mathbb{T}}$ edges, and $\pi(u')$ is also a path in $\mathrm{Lk}(u)$ in $\mathbb{M}$, modulo taking copies of $\hat{\mathbb{T}}$ vertices as identical.

First, u' cannot be a maximum because u' is adjacent to $\hat{v}$. It cannot be a minimum either because then $\pi(u') \subseteq \mathrm{Lk}^+(u)$ and e_1 and e_2 are outgoing edges from u in $\hat{\mathbb{T}}$ connected to some component of $\mathrm{Lk}^+(u)$, which contradicts Lemma 3.6. Next, if $\mathrm{Lk}^+(u')$ is not connected, then its component U not containing $\hat{v}$ does not contain v_1' and v_2' either and thus u lies in the interior of the path $\pi(u')$. Then U is also a connected component of $\mathrm{Lk}^+(u)$ in $\mathbb{M}$. Unless u is a negative saddle, by Lemma 3.6, there is an outgoing edge in $\hat{\mathbb{T}}$ from u to a vertex in U, contradicting the fact that e_1' and e_2' are the only edges adjacent to u' that are copies of $\hat{\mathbb{T}}$ edges. Hence, unless u is a negative saddle, $\mathrm{Lk}^+(u')$ is connected and u' is a regular vertex in $\mathbb{M}_0$.

Finally, suppose u is a negative saddle, with two components U_1 and U_2 in $\mathrm{Lk}^+(u)$. By Lemma 3.6, u has exactly one outgoing edge e in $\hat{\mathbb{T}}$. Without loss of generality assume that e is connected to U_1. Then U_2 will appear as a connected component of the upper link of exactly one copy u' of u, namely if $U \subseteq \pi(u')$, and u' will be a negative saddle in $\mathbb{M}_0$. The upper link of all other copies of u will be connected — consisting of $\hat{v}$ and possibly a portion of U_1. Consequently, one copy of every negative saddle of $\mathbf{M}$ becomes a negative saddle in $\mathbf{M}_0$ and other copies become regular vertices. This completes the proof of the lemma. ■

Next we perform a similar surgery on $\mathbf{M}_0$ only using a blue cut-tree $\check{\mathbb{T}}$ of $\mathbf{M}_0$. As above, the idea is to slice the plane along $\check{\mathbb{T}}$ and insert a new vertex $\check{v}$ in the resulting face and connect $\check{v}$ to every copy u of a vertex in $\check{\mathbb{T}}$ by an outgoing edge $(\check{v}, u)$. We call the resulting triangulation $\tilde{\mathbb{M}} = (\tilde{V}, \tilde{E}, \tilde{F})$. A slight technicality arises in this case as a result of the fact that v_∞ is a minimum of $\mathbb{M}_0$ which by Corollary 3.5 is a vertex of $\check{\mathbb{T}}$. As it will become clear later, we only need to treat $\check{v}$ symbolically below v_∞ by connecting them by an edge oriented toward v_∞. We conclude, using the same argument as in Lemma 3.7, the following:

Lemma 3.8 *$\tilde{\mathbb{M}}$ does not have saddle vertices.*

Lemma 3.9 *If (f_1^*, f_2^*) is an edge of $\mathbb{M}^*$, then there is a path from f_1^* to f_2^* in $\tilde{\mathbb{M}}^*$.*

Proof. If f_1 and f_2 are adjacent in $\tilde{\mathbb{M}}$ then (f_1^*, f_2^*) is an edge in $\tilde{\mathbb{M}}^*$. Thus we only need to consider the case in which an edge e shared by f_1 and f_2 in $\mathbb{M}$ is an edge of $\hat{\mathbb{T}}$ or $\check{\mathbb{T}}$ (or both). Suppose e is an edge of $\hat{\mathbb{T}}$. In constructing $\mathbb{M}_0$, e is duplicated to create two edges e_1 and e_2, respectively, incident to f_1 and f_2. Let ϕ_1 and ϕ_2 respectively be the $\hat{v}$-triangles based at e_1 and e_2. By construction, f_1 is to the left and ϕ_1 to the right of e_1 and therefore (f_1^*, ϕ_1^*) is an edge in $\tilde{\mathbb{M}}^*$. Similarly (ϕ_2^*, f_2^*) are edges in $\tilde{\mathbb{M}}^*$. Consider the subgraph of $\tilde{\mathbb{M}}^*$ induced by $\hat{v}$-triangles. Since all the edges incident to $\hat{v}$ are incoming, their duals make a cycle in $\tilde{\mathbb{M}}^*$ which includes ϕ_1^* and ϕ_2^*. Since there is a path from ϕ_1^* to ϕ_2^* on this cycle and there are edges from f_1^* to ϕ_1^* and from ϕ_2^* to f_2^* in $\tilde{\mathbb{M}}^*$, we get a path from f_1^* to f_2^*. It is easy to observe that the same argument extends to neighboring $\mathbb{M}$ triangles that are separated by the edges of $\check{\mathbb{T}}$ or both $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$. ■

3.4 Encoding of contours in the resulting basic terrain

Although we argued above that $\tilde{\mathbb{M}}$ can be realized and therefore treated as the triangulation of a terrain $\tilde{\mathbf{M}}$, to relate the level sets of $\tilde{\mathbf{M}}$ to those of $\mathbf{M}$, in the rest of this section we use a degenerate realization of $\tilde{\mathbb{M}}$ as a surface in $\mathbb{R}^3$ that differs from what a straight-line embedding of $\tilde{\mathbb{M}}$ results. This substantially simplifies the arguments that follow. We shall realize the $\hat{v}$- and $\check{v}$-triangles as vertical *curtains*: An *upward* (resp. *downward*) *extending* curtain based at a segment pq in $\mathbb{R}^3$ is the convex hull of two infinite rays shot in positive (resp. negative) direction of the z-axis from the points p and q respectively. A curtain can be regarded as a vertical (orthogonal to the xy-plane) triangle that has a vertex at infinity.

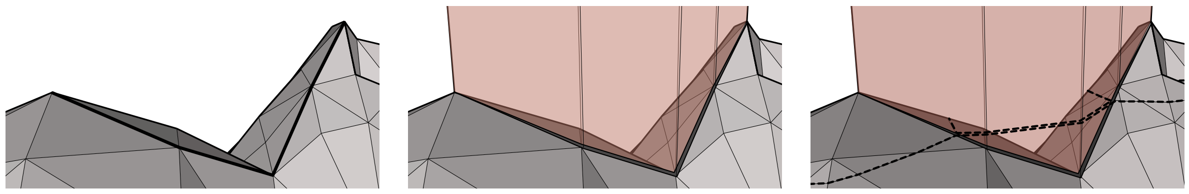

Figure 9: Left: The red cut-tree $\hat{\mathbb{T}}$ of a terrain is drawn using heavier segments on the terrain $\mathbf{M}$. Middle: The terrain is sliced along $\hat{\mathbb{T}}$ and $\hat{v}$-triangles are represented by upward extending curtains. Note that each edge e of $\hat{\mathbb{T}}$ results two overlapping curtains one based at each of the two (overlapping) copies of e that results from cutting the terrain along $\hat{\mathbb{T}}$. Right: A contour of the resulting terrain (dashed) overlapping itself on $\hat{v}$-triangles.

We realize $\tilde{\mathbb{M}}$ by preserving the geometry of every triangles that existed in $\mathbf{M}$ and representing $\hat{v}$-triangles (resp. $\check{v}$-triangles) as upward (resp. downward) extending curtains based at segments corresponding to the edges of $\hat{\mathbb{T}}$ (resp. $\check{\mathbb{T}}$) on $\mathbf{M}$; see Figure 9. Note that in this realization of $\tilde{\mathbb{M}}$, the two copies of each $\hat{\mathbb{T}}$ or $\check{\mathbb{T}}$ edge overlap as do the segments that represent them on $\tilde{\mathbf{M}}$ and therefore the curtains tha realize their corresponding $\hat{v}$- or $\check{v}$-triangles also overlap. Although in this sense $\tilde{\mathbf{M}}$ is not the graph of a bivariate function, it can still be regarded as a (self-overlapping) piece-wise linear surface in $\mathbb{R}^3$ and a level set $\tilde{\mathbb{M}}_\ell$ of it can be defined as the projection into the xy-plane of the set $\tilde{\mathbf{M}}_\ell = \tilde{\mathbf{M}} \cap z_\ell$. Let $\hat{\mathbf{T}}$ and $\check{\mathbf{T}}$ respectively be shorthands for $\mathbf{M}(\hat{\mathbb{T}})$ and $\mathbf{M}(\check{\mathbb{T}})$. Note that $\hat{\mathbf{T}}$ and $\check{\mathbf{T}}$ are also contained in $\tilde{\mathbf{M}}$, although under the topology of this surface they are (self-overlapping) closed curves that correspond to Eulerian traversals of $\hat{\mathbf{T}}$ and $\check{\mathbf{T}}$ on $\mathbf{M}$.

Since every triangle of $\mathbb{M}$ is also in $\tilde{\mathbb{M}}$ and is geometrically realized by the same triangle in both $\mathbf{M}$ and $\tilde{\mathbf{M}}$, $\mathbb{M}_\ell \subseteq \tilde{\mathbb{M}}_\ell$ for all $\ell \in \mathbb{R}$ and $\tilde{\mathbb{M}}_\ell \setminus \mathbb{M}_\ell \subset \hat{\mathbb{T}} \cup \check{\mathbb{T}}$ (with some abuse of notation we write $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$ to refer the red and blue cut-trees as subgraphs $\mathbb{M}$ as well as their drawings as subsets of $\mathbb{R}^2$). In other words $\tilde{\mathbb{M}}_\ell$ consists of the contours of $\mathbb{M}_\ell$ together with fragments of the red and blue cut-trees.

Lemma 3.10 *Let K_0 be a blue (resp. red) contour in a level set $\mathbb{M}_\ell$ and let $K_1, \dots, K_r$ be its children. Let R be the interior of $K_0^i \setminus (K_1^i \cup \dots \cup K_r^i)$. Then $\tilde{\mathbb{M}}_\ell \cap R = \hat{\mathbb{T}} \cap R$ (resp. $\check{\mathbb{T}} \cap R$).*

Proof. We prove the lemma for the case where K_0 is blue. The proof for the case where it is red is symmetric. Let Q and $\tilde{Q}$ respectively be shorthands for $\mathbf{M}(R)$ and $\tilde{\mathbf{M}}(R)$. By definition, $Q \subseteq \tilde{Q}$. Since K_0 is a blue contour of $\mathbf{M}$, Q is entirely below the plane z_ℓ. Since Q and $\tilde{Q}$ differ only in curtains based at $\hat{\mathbf{T}}$ or $\check{\mathbf{T}}$ segments, $z_\ell \cap \tilde{Q}$ is contained in such curtains. On the other hand any curtain whose intersection with $\tilde{Q}$ intersects z_ℓ has to be extending upward from some segment of $\hat{\mathbf{T}}$ that intersects Q. Thus $\tilde{\mathbb{M}}_\ell \cap R \subseteq \hat{\mathbb{T}}$. Conversely, any segment of $\hat{\mathbf{T}}$ that intersects Q is the base of an upward extending curtain which intersects z_ℓ. Thus $\hat{\mathbb{T}} \cap R \subseteq \tilde{\mathbb{M}}_\ell$. ■

Let us fix a regular level ℓ of h and and let $K_1, \dots, K_t$ be the contours in $\mathbb{M}_\ell$. For simplicity, we virtually add an infinitely large contour K_0 that bounds the entire plane. Let $\mathcal{K}_\ell = \{K_0, K_1, \dots, K_t\}$. Consider the set $\mathcal{R}_\ell = \{R_0, R_1, \dots, R_t\}$ of the connected component of $\mathbb{R}^2 \setminus \mathbb{M}_\ell$. The boundary of each R_i, $i \geq 0$, consists of a set $B(R_i) = \{K_{i_0}, K_{i_1}, \dots, K_{i_{r(i)}}\} \subseteq \mathcal{K}_\ell$ of $r(i)$ contours in which $K_{i_1}, \dots, K_{i_{r(i)}}$ are the children of K_{i_0} in the arrangement of contours in $\mathcal{K}_\ell$.

For any $R \in \mathcal{R}_\ell$ we construct an undirected graph $G_\ell(R) = (V_R, E_R)$ as follows. By Lemma 3.10, $\tilde{\mathbb{M}}_\ell \cap R$ is contained in exactly one of $\hat{\mathbb{T}}$ or $\check{\mathbb{T}}$; let $\mathbb{T}$ denote that tree. The vertex set V_R of $G_\ell(R)$ consists of all the vertices of $\mathbb{T}$ that are contained in R together with one *auxiliary vertex* v_K associated with every contour $K \in B(R)$. The edge set E_R of $G_\ell(R)$ consists of an edge $\{u, v\}$ corresponding to each edges (u, v) of $\mathbb{T}$ whose endpoints u and v are both in R together with an edge

$\{v, v_K\}$ for any edge of $\mathbb{T}$ that crosses a contour $K \in B(R)$ and its endpoint in R is v. Equivalently, $G_\ell(R)$ is obtained from the subgraph of $\mathbb{T}$ that is induced by those edges of $\mathbb{T}$ that intersect R, by identifying all vertices that are contained in each component $R' \neq R$ of $\mathcal{R}_\ell$ that is separated from R by a contour $K \in B(R)$ into a single vertex v_K.

Lemma 3.11 *For any $R \in \mathcal{R}_\ell$ the graph $G_\ell(R)$ as defined above is a tree.*

Proof. Let $K_0, K_1, \ldots, K_r$ be the contours bounding R and let K_0 be the parent of $K_1, \ldots, K_r$. We prove the lemma for the case where K_0 is blue. The proof for the case where it is red is symmetric. We prove that the existence of a cycle in $G_\ell(R)$ implies the existence of a cycle in the underlying undirected graph of $\hat{\mathbb{T}}$ which contradicts Corollary 3.5. Consider any contour K_j, $j \geq 1$ and let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two edges of $\hat{\mathbb{T}}$ that intersect K_j. Since K_j is red, $v_1, v_2 \in K_j^{\mathrm{i}}$. Since e_1 is an edge of $\hat{\mathbb{T}}$, v_1 is followed in $\hat{\mathbb{T}}$ by an ascending path that ends at a maximum. Since no ascending path can leave K_j^{i}, $\hat{\mathbb{T}}$ reaches a maximum v_1' in K_j^{i} through v_1. Similarly, $\hat{\mathbb{T}}$ reaches a maximum v_2' in K_j^{i} through v_2. Lemma 3.4 implies that v_1' and v_2' are connected by a path in the underlying undirected graph of $\hat{\mathbb{T}}$ that is contained in K_j^{i}. In other words, any two branches of $\hat{\mathbb{T}}$ that enter K_j^{i} meet in K_j^{i}. Thus if we contract every edge of $\hat{\mathbb{T}}$ whose endpoints are both outside R, we precisely get the graph $G_\ell(R)$. The proof of the lemma follows from the fact that the result of contracting a tree edge is a tree. ■

We next combine the graphs $G_\ell(R), R \in \mathcal{R}_\ell$ into a graph G_ℓ by identifying, for any two components R_1 and R_2 that share a contour K in their boundaries, the two auxiliary vertices v_K associated to K in $G_\ell(R_1)$ and $G_\ell(R_2)$. The acyclic structure of the hierarchy of red and blue contours together with Lemma 3.11 result the following statement.

Corollary 3.12 *The graph G_ℓ is a tree.*

Let P be a $\check{v}$-$\hat{v}$ path in $\tilde{\mathbb{M}}$. Since $\tilde{\mathbf{M}}$ is a basic terrain (does not have saddle vertices), by Lemma 3.1 $\tilde{\mathbb{M}}^* \setminus P^*$ is acyclic. Let $\prec$ be the adjacency partial order on $\tilde{F}$ induced by $\tilde{\mathbb{M}}^* \setminus P^*$. Since $F \subset \tilde{F}$, $\prec$ is also a partial order when restricted to F.

Lemma 3.13 *Let $\lhd$ be a linear extension of $\prec$ on F. If K and K' are two contours of a level set $\mathbb{M}_\ell$ and $f_1, f_2 \in F(K)$ and $f_1', f_2' \in F(K')$ are such that $f_1 \lhd f_1' \lhd f_2$, then $f_1 \lhd f_2' \lhd f_2$.*

Proof. Since $\tilde{\mathbf{M}}$ is a basic terrain, $\tilde{\mathbb{M}}_\ell$ consists of a single contour. Let $C^* = C(K, \mathbb{M}^*)$ be the representing cycle of $\tilde{\mathbb{M}}_\ell$ in $\tilde{\mathbb{M}}^*$. If $f_1, f_2 \in F(K)$ for some contour K in $\mathbb{M}_\ell$ and the edge common to f_1 and f_2 does not belong to either of $\hat{\mathbb{T}}$ or $\check{\mathbb{T}}$, then (f_1^*, f_2^*) is an edge in C^*. By Lemma 3.1, C^* has exactly one edge in P^*. Thus $C^* \setminus P^*$ is a path Q^* that is exactly one edge short of C^*.

Let G_ℓ be the tree of Corollary 3.12. We map the path Q^* in $\tilde{\mathbb{M}}_\ell$ into a walk W in G_ℓ as follows: Each vertex f^* of Q^* is mapped to a pair (u, v) where either u and v are neighboring vertices in G_ℓ or $u = v$. Specifically, if $f \in \mathbb{M}_\ell(K)$, then f^* is mapped into (v_K, v_K) where v_K is the vertex of G_ℓ that represents K. Otherwise, f^* is dual to of some $\check{v}$- or $\hat{v}$-triangle f in $\tilde{\mathbb{M}}$ that intersects $\tilde{\mathbb{M}}_\ell$. Let $e = (u, v)$ be the edge in $\mathbb{M}$ at (a copy of) which f is based in $\tilde{\mathbb{M}}$. If the edge e does not intersect $\mathbb{M}_\ell$, it is contained in $G_\ell(R)$ (as an undirected edge) for precisely one $R \in \mathcal{R}_\ell$, in which case we map f^* to (u, v). On the other hand, if e intersects $\mathbb{M}_\ell$ at a contour K, then by Lemma 3.10 there will be precisely one edge $\{v_K, v\}$ in G_ℓ where v is an endpoint of e, in which case we map f^* into (v, v_K). It can be verified that the set of pairs in the image of Q^* under this mapping is indeed a walk W in G_ℓ. Since $\tilde{\mathbb{M}}_\ell$ goes thorough any segment s at most twice, each edge of G_ℓ appears at most twice in W.

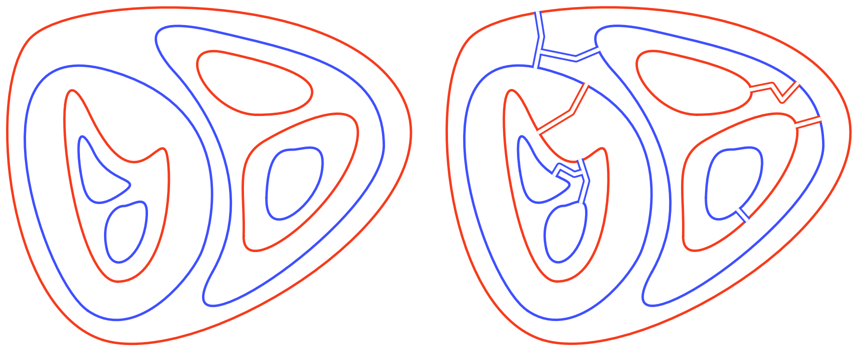

Figure 10: Contours of $\mathbb{M}_\ell$ (left) versus those of $\tilde{\mathbb{M}}_\ell$ (right).

For $f_1 \lhd f'_1 \lhd f_2$ to hold, Q^* must visit f_1^*, ${f'_1}^*$ and f_2^* in this order. Assume without loss of generality that $f'_1 \lhd f'_2$. In order for $f_1 \lhd f'_2 \lhd f_2$ not to hold, one must have $f_2 \lhd f'_2$ which means Q^* must visit ${f'_2}^*$ after f_2^*. But this corresponds to going from K to K', then back to K and then again to K'. Since each of K and K' are represented by a vertex in G_ℓ this would mean that W goes thorugh v_K, $v_{K'}$, and again v_K in this order. Corollary 3.12 implies then that W has to traverse some edge of G_ℓ at least three times, a contradiction. ■

Lemmas 3.9 and 3.13 respectively prove that the total order $\lhd$ has properties (C1) and (C2) of a level-ordering .

Theorem 3.14 *For any terrain* $\mathbf{M}$ *with triangulation* $\mathbb{M}$*, there is exists a level level-ordering of the triangles of* $\mathbb{M}$*.*

4 Contour Algorithms

In this section we describe I/O-efficient algorithms for computing contour maps as well as an I/O-efficient data structure for answering contour queries.

4.1 Level-ordering of terrain triangles

We describe an I/O-efficient algorithm for computing, given a terrain $\mathbf{M}$, the triangulation $\tilde{\mathbb{M}}$ of the simplified terrain $\tilde{\mathbf{M}}$, and a monotone path P from $\check{v}$ to $\hat{v}$ in $\tilde{\mathbb{M}}$. We can then compute a level-ordering of the triangles of $\tilde{\mathbb{M}}$ in $O(\textsc{sort}(N))$ I/Os using an existing I/O-efficient topological sorting algorithm for planar DAGs [8]. This induces a level-ordering on the triangles of $\mathbb{M}$.

Computing the red cut-tree. The first step in computing $\tilde{\mathbb{M}}$ is to compute a red (ascending) cut-tree $\hat{\mathbb{T}}$ of $\mathbb{M}$. The I/O-efficient topological persistence algorithm of Agarwal et al. [2] can determine the type (regular, minimum, negative saddle, ...) of every vertex of $\mathbb{M}$ in $O(\textsc{sort}(N))$ I/Os. Moreover, for every vertex $v \in \mathbb{M}$, it can also compute, within the same I/O bound, a vertex from each connected component of $\mathrm{Lk}^+(v)$. Since each saddle of $\mathbb{M}$ is assumed to be simple, $\mathrm{Lk}^+(v)$ has at most two connected components.

To compute $\hat{\mathbb{T}}$, we apply the *time-forward processing* technique [10] using a priority queue Q: we scan the vertices of $\mathbb{M}$ in the increasing order of their heights. We store a subset of vertices in Q, namely the upper endpoints of the edges of $\hat{\mathbb{T}}$ whose lower endpoints have been scanned. The priority of a vertex v in Q is its height $h(v)$. Suppose we are scanning a vertex v of $\mathbb{M}$ and u is the lowest priority vertex in Q. If $h(v) < h(u)$ and v is not a positive saddle, we move to a new vertex in $\mathbb{M}$. Otherwise, i.e. if $h(u) = h(v)$ or v is a positive saddle, we choose a vertex w from each connected component of $\mathrm{Lk}^+(v)$, which we have already computed in the preprocessing step. We add the edge (v, w) to $\hat{\mathbb{T}}$ and add w to Q. Since each operation on Q can be performed in $O\left(\frac{1}{B}\log_{M/B} N/B\right)$ I/Os, $\hat{\mathbb{T}}$ can be computed in $O(\textsc{sort}(N))$ I/Os.

Adding the blue cut-tree. The second step in computing $\tilde{\mathbb{M}}$ is to compute a blue cut-tree $\check{\mathbb{T}}$ of $\mathbb{M}_0$. However, we can compute $\check{\mathbb{T}}$ directly on $\mathbb{M}$ if we ensure that $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$ do not cross each other, even though they can share edges. This property can be ensured by choosing the ascending and descending edges, in $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$, respectively, out of each vertex v, more carefully. Specifically, we use the following rule:

1. On an ascending path, the edge following (u, v) is (v, w) where (v, w) is the first outgoing edge out of v after (u, v) in clockwise order, and
2. On a descending path, the edge following (v, u) is (w, v) where (w, v) is the first incoming edge of v after (v, u) in counterclockwise order.

It can be verified that $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$ do not cross. One can therefore compute $\check{\mathbb{T}}$ precisely in the same way as $\hat{\mathbb{T}}$ directly on $\mathbb{M}$.

Computing a monotone $\check{v}$-$\hat{v}$ path P. While computing $\check{\mathbb{T}}$ we also compute a descending path starting at the lowest *positive* saddle v_1 of $\mathbb{M}$ as though v_1 were another negative saddle. This path P, which ends at a $\check{\mathbb{T}}$ vertex v_0, together with $(\check{v}, v_0)$ and $(v_1, \hat{v})$ serves as a monotone path in $\tilde{\mathbb{M}}$ connecting $\check{v}$ to $\hat{v}$.

Generating $\tilde{\mathbb{M}}^* \setminus P^*$. The topological sorting algorithm of Arge et al. [8] takes as input a planar directed acyclic graph, represented as a list of vertices along with the list of edges incident upon each vertex in circular order. Given $\mathbb{M}$, $\hat{\mathbb{T}}$, $\check{\mathbb{T}}$, and P, we need to compute such a representation of $\tilde{\mathbb{M}}^* \setminus P^*$. Since each face in $\tilde{\mathbb{M}}$ is a triangle, $\tilde{\mathbb{M}}^*$ is 3-regular. It is easy to compute the circular order of edges incident upon a vertex of $\tilde{\mathbf{M}}^*$ whose dual triangle is neither a $\hat{v}$- or $\check{v}$-triangle, nor adjacent to a copy of a $\hat{\mathbb{T}}$ or $\check{\mathbb{T}}$ edge. The main task is then to compute these the $\hat{v}$- and $\check{v}$-triangles. This can be accomplished by computing the Eulerian tours of $\hat{\mathbb{T}}$ and $\check{\mathbb{T}}$, which takes $O(\textsc{sort}(N))$ I/Os [8]. Putting everything together, we obtain the main result of this paper.

Theorem 4.1 *Given a terrain* $\mathbf{M}$ *with triangulation* $\mathbb{M}$*, a level-ordering of the triangles of* $\mathbb{M}$ *can be computed in* $O(\textsc{sort}(N))$ *I/Os, where* N *is the number of vertices of* $\mathbb{M}$.

4.2 Contour maps of basic terrains

Let $L = \{\ell_1, \ldots, \ell_s\}$ be a set of input levels with $\ell_1 < \cdots < \ell_s$. Given a basic terrain $\mathbb{M}$, the goal is to compute the contour map of $\mathbb{M}$ for levels in L. Since $\mathbb{M}$ is simple, each $\mathbb{M}_{\ell_i}$ consists of a single contour. Generating the segments of $\mathbb{M}_{\ell_i}$ in clockwise or counterclockwise order is equivalent to listing the triangles of $\mathbb{M}$ that the contour $\mathbb{M}_{\ell_i}$ intersects in that order, i.e. reporting $C(\mathbb{M}_\ell, \mathbb{M})$.

Our algorithm uses a *buffer tree* $\mathcal{B}$ to store the triangles of $\mathbb{M}$ that intersect a level set. The buffer tree [5] is a variant of B-tree, which propagates updates from the root to the leaves in a lazy manner, using buffers attached to the internal nodes of the tree. As a result, a sequence of N updates (inserts and deletes) can be performed in amortized $O(\textsc{sort}(N))$ I/Os. Moreover, one can perform a *flush* operation on a buffer tree that results in the writing of all its stored elements on the disk in sorted order. Flushing a tree with T elements takes $O(T/B)$ I/Os.

It is more intuitive to describe the algorithm as a plane sweep of $\mathbf{M}$ in $\mathbb{R}^3$. At the first step, the algorithm computes a level-ordering $\lhd$ of the terrain triangles using Corollary 3.2. Then starting at $\ell = -\infty$, the algorithm sweeps a horizontal plane z_ℓ in the positive z-direction. A *target level* ℓ_* is initially set to ℓ_1. At any time the algorithm maintains the list of triangles of $\mathbf{M}$ that intersect the sweeping plane in a buffer tree $\mathcal{B}$ ordered by $\lhd$. Whenever the sweep plane encounters the bottom-most vertex of a triangle f, we insert f into $\mathcal{B}$; f is deleted again from $\mathcal{B}$ when the plane reaches the top-most vertex of f. When the sweep plane z_ℓ reaches the target level ℓ_* (or rather the lowest vertex of height ℓ_* or more when the sweeping is implemented discretely), we flush the buffer tree. The generated list of triangles (vertices of $\mathbb{M}^*$) are precisely the set of triangles in $\mathbb{M}$ that intersect z_{ℓ_*} ordered by $\lhd$. Corollary 3.2 implies that the output is exactly $C(\mathbb{M}_{\ell_*}, \mathbb{M})$. The algorithm then raises the target level ℓ_* to the next level in L and continues.

Level-ordering of the terrain triangles takes $O(\text{SORT}(N))$ I/Os (Theorem 4.1). Preprocessing for the sweep algorithm consists of sorting the vertices in their increasing order of heights which can also be done in $O(\text{SORT}(N))$ I/Os. During the sweep each update on the buffer tree takes $O(\frac{1}{B}\log_{M/B} N/B)$ amortized I/Os [5]. Thus all the $O(N)$ updates can be performed in $O(\text{SORT}(N))$ I/Os in total. Each flushing operation takes $O(T'/B)$ I/Os, where T' is the number of triangles in $\mathcal{B}$. If a triangle is in $\mathcal{B}$ but has been deleted, it is not in $\mathcal{B}$ after the flushing operation, so a "spurious" triangle is flushed only once.

Hence, the total number of I/Os is $O(\text{SORT}(N) + T/B)$, where T is the output size. Finally, in addition to storage used for the terrain the algorithm uses $O(N/B)$ blocks to store the buffer tree and thus uses $O(N/B)$ blocks in total.

4.3 Generalization to arbitrary terrains

Given a general terrain $\mathbf{M}$ with saddles, one can still compute by Theorem 4.1 a level-ordering of the triangles of $\mathbb{M}$ in $O(\text{SORT}(N))$ I/Os. If one runs the algorithm of the previous section on $\tilde{\mathbb{M}}$ the output generated for each input level ℓ_i is $C(\tilde{\mathbb{M}}_{\ell_i}, \tilde{\mathbb{M}})$. Running the algorithm on $\mathbb{M}$ is equivalent to running it on $\tilde{\mathbb{M}}$ but ignoring all $\hat{v}$ and $\check{v}$-triangles by omitting their insertions into the buffer tree. Consequently, the produced output for level ℓ_i is the same sequence of triangles only with $\hat{v}$ and $\check{v}$-triangles omitted. By Theorem 3.14 this is a subsequence $R = \langle f_1, \dots, f_k \rangle$ of $C(\tilde{\mathbb{M}}_{\ell_i}, \tilde{\mathbb{M}}))$ in which $C(K, \mathbb{M})$ of each contour K in $\mathbb{M}_{\ell_i}$ appears as a subsequence R_K. Thus all one needs to do is to extract the subsequence R_K and write it separately in the same order as it appears in R. Property (C2) of a level-ordering allows this to be done in $O(k/B)$ I/Os: if in scanning the sequence R from left to right some elements of R_K are later followed by elements of $R_{K'}$, then the appearance of another element of R_K, indicates that no more elements from $R_{K'}$ remain.

We thus scan the sequence R from left to right and *push* the scanned triangles into a stack S_F. Every time the last element of a contour is pushed into the stack, the triangles of that contour make a suffix of the list of elements stored in S_F. At such a point, we *pop* all the elements corresponding to the completed contour and write them to the disk. To find out when a contour is completed and how many elements on the top of stack belong to it, we keep a second stack S_E of edges. For any triangle $f \in F(\mathbb{M}_{\ell_i})$, two of the edges of f intersect $\mathbb{M}_{\ell_i}$. With respect to the orientation of these edges, f is to the right of one of them and to the left of the other one which we respectively call the *left* and *right* edges of f at level ℓ_i. If $e^* = (f_j^*, f_{j+1}^*)$ is an edge of the representing cycle of a contour in $\mathbb{M}_{\ell_i}$, then e is the right edge of f_j and left edge f_{j+1} at level ℓ_i. We therefore check when scanning a triangle f_j whether its left edge is the same as the right edge of the triangle on top of S_F and insert f_j into S_F if this is the case. Otherwise, we compare the left edge of f_j with the edge on top of S_E. If they are not the same, we are visiting a new contour and we insert the left edge of f_j into S_E and f_j into S_F. Otherwise, f_j is the last triangle of its contour. Therefore we write it to the disk and successively pop and write to disk enough triangles from S_F until the left edge of a popped triangle matches the right edge of f_j. We also pop this edge from S_E. In this algorithm each scanned triangle is pushed to the stack once and popped once. In a standard I/O-efficient stack implementation this costs $O(k/B)$ I/Os.

Theorem 4.2 *Given any terrain* $\mathbf{M}$ *with* N *vertices and a list* $L = \{\ell_1, \ldots, \ell_s\}$ *of levels with* $\ell_1 < \cdots < \ell_s$*, one can compute using* $O(\textsc{sort}(N) + T/B)$ *I/Os the contour map of* $\mathbf{M}$ *for levels in* L*, where* T *is the total number of produced segments.*

4.4 Extracting nesting of contours

In addition to reporting each contour individually, a number of applications call for computing how various contours are nested within each other. We produce this information by returning the parent of each contour in a computed contour map.

The parent-child relationship between individual contours in a contour map of a terrain $\mathbf{M}$ can be read from the contour tree of $\mathbf{M}$. Each contour in a contour map corresponds to a point on some edge of the contour tree. Two contours K and K' in the map are neighbors (either siblings or parent and child) if and only if their corresponding points on the contour tree can be connected by a path that does not pass through a point corresponding to a third contour $K'' \neq K, K'$ in the given contour map. Each edge of the contour tree can be colored red or blue according to the color of the contours it represents (all contours represented by the points on the same edge of the contour tree have the same color). By the assumption that all saddles are simple, internal nodes of the contour tree which correspond to saddles all have degree three. It can be verified that at a joining (negative) saddle only two color combinations on the edges incident to the saddles are possible. The same holds for a splitting (positive) saddle. In each case, using the edges colors, and using the corresponding patterns in which contours of various colors can merge or split, one can uniquely determine one of the edges incident to the saddle that carries contours that are parents to those carried by the other two. We "orient" this edge at each saddle away from and the other two toward the saddle. The resulting orientation on the contour-tree is equivalent to orienting each blue edge toward its higher end and each red edge toward its lower end. With this orientation of the contour tree a contour K will be the parent of a contour K', if the path between points representing K and K' on the contour tree follows the orientation of the edges of the tree.

The algorithm of Arge et al. [2] for computing the contour tree can return the color of each edge on the generated tree. Thus if a contour tree of the terrain is available, by oriented the edges of the tree to point toward the neighboring edge and one can determine the edge of the contour tree on which each contour in a computed contour map is represented, one can determine the parent-child relationship between individual contours in $O(\textsc{sort}(N) + T/B)$ I/Os, where T is the number of contours in the given contour map, through a pre-order traversal of the oriented contour tree.

To facilitate finding of the edge the contour tree that carries a contour in the given map, instead of the the contour tree, we compute the *augmented contour-tree* [2]. The augmented contour tree of a terrain replaces each edge of the contour tree with a monotonically ascending path whose vertices are the vertices of the terrain. Every regular vertex of the terrain appears precisely once in one of these paths. All the properties of contour tree are also valid for the augmented contour tree. We store in each vertex u of the terrain a pointer to each of the (at most two if u is a saddle and one if u is regular) edges in the augmented contour tree that have u as the lower endpoint. To locate the edge in the augmented contour tree corresponding to a computed contour K, we scan the list of triangles that intersect K and determine the vertices u and u' of these triangles respectively highest below and lowest above the level of K. Since there are no vertices between u and u', shifting K down or up between $h(u)$ and $h(u')$ does not change the set of triangles it intersects and therefore its homology class. Consequently, uu' is an edge of the augmented contour tree. All that is needed then is to located the pointer to the contour tree edge stored at u that matches uu'. Since the augmented contour tree of a terrain of size N can be computed in $O(\textsc{sort}(N))$ I/Os [2], we summarize the above discussion as follows:

Theorem 4.3 *For a contour map consisting of T contours of a terrain with N triangles, where each contour is given as a list of triangles that intersect it together with its level, one can report for each contour in the map a pointer to its parent in $O(\textsc{sort}(N) + T/B)$ I/Os.*

4.5 Answering contour queries

The sweep algorithm described in the previous section can easily be modified to construct a linear space data structure that given a query level ℓ can report the contours in the level set $\mathbb{M}_\ell$ I/O-efficiently. Unlike the previously known structure for this problem [1], our structure can be constructed in $O(\textsc{sort}(N))$ I/Os. To obtain the structure we simply replace the buffer tree $\mathcal{B}$ with a partially persistent B-tree [6, 23]. To build the structure, we sweep $\mathbf{M}$ by a horizontal plane in the same way as we did in the algorithm of Section 4.2, inserting the triangles when the sweep plane reaches their bottom-most vertex, without checking for them to intersect any target levels, and deleting them when the sweep plane passes their top-most vertex. There will also be no need to flush the tree.

Since $O(N)$ updates can be performed on a persistent B-tree in $\textsc{sort}(N)$ I/Os [20, 7], the sweeping of the terrain require $O(\textsc{sort}(N))$ I/Os. A persistent B-tree allows us to query any previous version of the structure and in particular produce the list of the elements stored in the tree in $O(\log_B N + T/B)$ I/Os when T is the number of reported elements. Therefore we can obtain $\mathbb{M}_\ell$ in the same bound, simply by querying the structure for the triangles it contained when the sweep-plane was at height ℓ and then utilize Theorem 3.14 and the contour extraction algorithm discussed above to extract individual contours of $\mathbb{M}_\ell$.

Theorem 4.4 *Given a terrain $\mathbf{M}$ with N vertices, one can construct in $O(\textsc{sort}(N))$ I/Os a linear size data structure, such that given a query level ℓ, one can report contours of $\mathbb{M}_\ell$ in $O(\log_B(N) + T/B))$ I/Os where T is the size of the query output. Each contour is reported individually, and the edges of each contour are sorted in clockwise order.*

5 Conclusions

We defined level-ordering of terrain triangles and proved that every terrain has a level ordering that can be computed I/O-efficiently. Based on this, we provided algorithms that compute contours of a given terrain within similar I/O bounds. An immediate question is whether this approach can be generalized to triangulated surfaces of arbitrary genus with arbitrary piecewise linear functions defined on them. Notice that the problem is valid even if the input surface is not embedded in $\mathbb{R}^3$. Another interesting open problem for surfaces that are embedded in $\mathbb{R}^3$ is computing of level sets or answering contour queries for a height function defined by a variable z direction: is it possible to preprocess a given triangulated surface embedded in the three space so that for any given direction, the contours of the height function for that direction can be computed I/O-efficiently?

References

[1] P. K. Agarwal, L. Arge, T. M. Murali, K. R. Varadarajan, and J. S. Vitter, I/O-efficient algorithms for contour-line extraction and planar graph blocking, *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, 1998, pp. 117–126.

[2] P. K. Agarwal, L. Arge, and K. Yi, I/O-efficient batched union-find and its applications to terrain analysis, *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, 2006, pp. 167–176.

[3] A. Aggarwal and J. S. Vitter, The input/output complexity of sorting and related problems, *Commun. ACM*, 31 (1988), 1116–1127.

[4] L. Arge, External memory data structures, in: *Handbook of Massive Data Sets* (J. Abello, P. M. Pardalos, and M. G. C. Resende, eds.), Kluwer Academic Publishers, 2002, pp. 313–358.

[5] L. Arge, The buffer tree: A technique for designing batched external data structures, *Algorithmica*, 37 (2003), 1–24.

[6] L. Arge, A. Danner, and S.-H. Teh, I/O-efficient point location using persistent B-trees, *Proc. Workshop on Algorithm Engineering and Experimentation*, 2003.

[7] L. Arge, K. H. Hinrichs, J. Vahrenhold, and J. S. Vitter, Efficient bulk operations on dynamic R-trees, *Algorithmica*, 33 (2002), 104–128.

[8] L. Arge, L. Toma, and N. Zeh, I/O-efficient topological sorting of planar dags, *Proc. 15th Annu. ACM Sympos. Parallel Algorithms and Architectures*, 2003, pp. 85–93.

[9] H. Carr, J. Snoeyink, and U. Axen, Computing contour trees in all dimensions, *Computational Geometry: Theory and Applications*, 24 (2003), 75–94.

[10] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter, External-memory graph algorithms, *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, 1995, pp. 139–149.

[11] Y.-J. Chiang and C. T. Silva, I/O optimal isosurface extraction, *Proc. IEEE Visualization*, 1997, pp. 293–300.

[12] A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitasova, TerraStream: From elevation data to watershed hierarchies, *Proc. ACM Sympos. on Advances in Geographic Information Systems*, 2007, 212–219.

[13] H. Edelsbrunner, D. Letscher, and A. Zomorodian, Topological persistence and simplification, *Discrete Comput. Geom.*, 28 (2002), pp. 511–533.

[14] I. Fáry, On straight lines representation of planar graphs, *Acta Sci. Math. Szeged*, 11 (1948), 229–233.

[15] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink, Streaming computation of Delaunay triangulations, *Proc. of SIGGRAPH*, 2006, pp. 1049–1056.

[16] W. Lorensen and H. Cline, Marching cubes: a high resolution 3d surface construction algorithm, *Comput. Graph.*, 21 (1987), 163–170.

[17] M. H. Nodine, M. T. Goodrich, and J. S. Vitter, Blocking for external graph searching, *Algorithmica*, 16 (1996), 181–214.

[18] C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom, Out-of-core algorithms for scientific visualization and computer graphics, *Visualization'02*, 2002. Course Notes for Tutorial 4.

[19] R. A. Skelton, Cartography, *History of Technology*, 6 (1958), 612–614.

[20] J. van den Bercken, B. Seeger, and P. Widmayer, A generic approach to bulk loading multidimensional index structures, *Proc. International Conference on Very Large Databases*, 1997, pp. 406–415.

[21] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore, Contour trees and small seed sets for isosurface traversal, *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 1997, pp. 212–219.

[22] G. Reeb, Sur les points singuliers dune forme de Pfaff complèment intégrable ou dune fonction numérique. *Comptes Rendus de LAcadémie des Sciences*, Paris 222 (1946), 847-849.

[23] P. J. Varman and R. M. Verma, An efficient multiversion access structure, *IEEE Transactions on Knowledge and Data Engineering*, 9 (1997), 391–409.

[24] J. S. Vitter, External memory algorithms and data structures: Dealing with MASSIVE data, *ACM Computing Surveys*, 33 (2001), 209–271.