Maintaining Contour Trees of Dynamic Terrains*

Pankaj K. Agarwal[1], Thomas Mølhave[2], Morten Revsbæk[3], Issam Safa[4], Yusu Wang[5], and Jungwoo Yang[3]

1 **Department of Computer Science, Duke University, USA**
2 **Scalable Algorithmics – SCALGO, USA**
3 **MADALGO – Center for Massive Data Algorithmics, Aarhus University, Denmark**
4 **Computational Lithography Group, Intel Corporation, USA**
5 **Department of Computer Science and Engineering, The Ohio State University, USA**

Abstract

We study the problem of maintaining the contour tree $\mathbb{T}$ of a terrain Σ, represented as a triangulated xy-monotone surface, as the heights of its vertices vary continuously with time. We characterize the combinatorial changes in $\mathbb{T}$ and how they relate to topological changes in Σ. We present a kinetic data structure (KDS) for maintaining $\mathbb{T}$ efficiently. It maintains certificates that fail, i.e., an *event* occurs, only when the heights of two adjacent vertices become equal or two saddle vertices appear on the same contour. Assuming that the heights of two vertices of Σ become equal only $O(1)$ times and these instances can be computed in $O(1)$ time, the KDS processes $O(\kappa + n)$ events, where n is the number of vertices in Σ and κ is the number of events at which the combinatorial structure of $\mathbb{T}$ changes, and processes each event in $O(\log n)$ time. The KDS can be extended to maintain an augmented contour tree and a join/split tree.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Contour tree, dynamic terrain, kinetic data structure

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.796

1 Introduction

Let $\mathbb{M}$ be a triangulation of $\mathbb{R}^2$, also known as a *triangulated irregular network* (TIN), and let $h : \mathbb{M} \to \mathbb{R}$ be a continuous function, often called a *height function*, that is linear within each triangle of $\mathbb{M}$. The graph of h, denoted by Σ, is a triangulated xy-monotone surface in $\mathbb{R}^3$ and is called a *terrain*. There has been extensive work in computational geometry, GIS, and spatial databases on the design and analysis of terrain-analysis algorithms such as flood-risk analysis, visibility analysis, and navigation.

Given a height value ℓ, the level set of the height function h is the set of all points in $\mathbb{M}$ whose height values are ℓ. As ℓ varies, the level set continuously deforms and its topology changes at certain heights. Level sets and their topology are often used for the analysis and visualization of height functions. The contour tree of a height function encodes the evolution

* P. A. and T. M. supported by the ARO contract W911NF-13-P-0018; P.A. also supported by NSF under grants CCF-09-40671, CCF-10-12254, and CCF-11-61359, and by Grant 2012/229 from the U.S.–Israel Binational Science Foundation; M. R. and J. Y. supported by Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation; Y. W. supported by NSF under grants CCF-0747082 and CCF-1319406.

31st International Symposium on Computational Geometry (SoCG'15).
Eds.: Lars Arge and János Pach; pp. 796–811

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of the level sets and succinctly represents the topology of all level sets, and it has found applications in a wide array of data analysis and visualization problems [4, 6, 10, 11, 15].

A variety of applications involve height functions that vary with time. In some cases, the height function may vary continuously with time (e.g., temperature, air pressure, etc.), or it may be updated dynamically at discrete time values (e.g., the height function models the elevation of points on the Earth, and the elevation is updated because of new measurements reflecting the changes due to natural processes or human activity, or a user interacting with a GIS may wish to change the elevation at some places to see its impact on the hydrology, mobility, visibility of the surface). Motivated by theses applications we study how the contour tree changes as the height function varies with time. Even if the height function is updated in discrete steps, the only efficient way we know to update the contour tree is to treat such an update as a continuous change from the old value to the new value. We therefore focus on the case when the height function varies continuously with time.

Related work. Van Kreveld et al. [20] gave an $O(n \log n)$-time algorithm for constructing the contour tree of a piecewise-linear height function on $\mathbb{R}^2$, where n is the number of linear pieces in the height function. Their algorithm was extended to $\mathbb{R}^3$ by Tarasov and Vyalyi [19], and to arbitrary dimensions by Carr et al. [9]. Agarwal et al. [3] gave an I/O-efficient algorithm for constructing the contour tree of a terrain representation that does not fit in main memory.

There has been some work on contour trees of time-varying height functions. Sohn and Bajaj [17] and Szymczak [18] compute a mapping between the contour trees of the height function at successive time steps, but they ignore (possibly many) combinatorial changes in the contour tree between the time steps. Furthermore they compute the contour tree at each time step using a static algorithm. Edelsbrunner et al. [12] study how the Reeb graph (generalization of contour tree for height functions over manifolds with non-zero genus) of a smooth function on three-dimensional space evolves over time. They characterize the combinatorial changes in the Reeb graph, and they describe an algorithm for updating the Reeb graph whenever a combinatorial change occurs. Their algorithm, however, works in an off-line setting, i.e., they assume that the height function over all time values is given in advance, and the algorithm takes $O(n)$ time to update the graph at each event.

There is extensive work in computational geometry on maintaining geometric/topological structures over a set of continuously moving or varying objects mostly under the kinetic data structure (KDS) framework introduced by Basch et al. [5]. See [13] for a survey of this line of work.

Our results. We describe the first KDS for maintaining the contour tree of a time-varying piecewise-linear height function over a simple triangulated (zero-genus) 2-manifold. The KDS can be extended to maintain the augmented contour tree and the join/split tree.

Our first result is a detailed characterization of the combinatorial changes in the contour tree, more refined than what is described in [12], and how they relate to topological changes in the height function (Section 3). This refined characterization, together with the use of two auxiliary trees, enables us to develop an efficient algorithm for updating the contour tree at each combinatorial change. The second result (Section 4) is a linear-size KDS for maintaining the contour tree. Assuming that the height of each vertex, as a function of time, is specified by a polynomial of constant degree and that the roots of these polynomials can be computed in $O(1)$ time, the KDS can update the contour tree in $O(\log n)$ time at each event. The KDS processes a total of $O(\kappa + n)$ events, where κ is the number of combinatorial

changes in the contour tree and n is the number of vertices in the triangulation. Finally we adapt our KDS to maintain the augmented contour tree and the join/split tree (Section 5).

2 Preliminaries

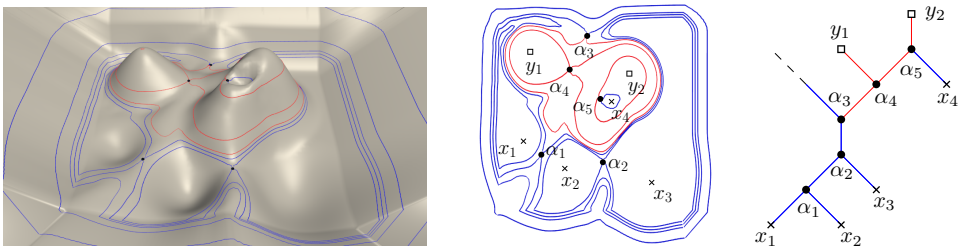
Terrains. Let $\mathbb{M} = (V, E, F)$ be a triangulation of $\mathbb{R}^2$, with vertex, edge, and face (triangle) sets V, E, and F, respectively, and let $n = |V|$. For simplicity we focus on $\mathbb{M}$ being a triangulation of $\mathbb{R}^2$, but our algorithm works for any genus-zero 2-manifolds. We assume that V contains a vertex v_∞ 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 $h : \mathbb{M} \to \mathbb{R}$ be a *height function*. We assume that the restriction of h to each triangle of $\mathbb{M}$ is a linear map, that h approaches $-\infty$ at v_∞, and that the heights of all vertices are distinct. Given $\mathbb{M}$ and h, the graph of h, called a *terrain* and denoted by Σ, is an xy-monotone triangulated surface whose triangulation is induced by $\mathbb{M}$. The vertices, edges, and faces of Σ are in one-to-one correspondence with those of $\mathbb{M}$ and with a slight abuse of terminology we refer to V, E, and F, as vertices, edges, and triangles of both Σ and $\mathbb{M}$.

Critical points. For a vertex v of $\mathbb{M}$, the *star* of v, denoted by $\mathrm{St}(v)$, consists of all triangles incident on v. The *link* of v, denoted by $\mathrm{Lk}(v)$, is the boundary of $\mathrm{St}(v)$, i.e., the cycle formed by the edges of $\mathbb{M}$ that are not incident on v but belong to the triangles of $\mathrm{St}(v)$. The lower (resp. upper) link of v, $\mathrm{Lk}^-(v)$ (resp. $\mathrm{Lk}^+(v)$), is the subgraph of $\mathrm{Lk}(v)$ induced by vertices u with $h(u) < h(v)$ (resp. $h(u) > h(v)$).

A *minimum* (resp. *maximum*) of $\mathbb{M}$ is a vertex v for which $\mathrm{Lk}^-(v)$ (resp. $\mathrm{Lk}^+(v)$) is empty. A maximum or a minimum vertex is called an *extremal* vertex. A non-extremal vertex v is *regular* if $\mathrm{Lk}^-(v)$ (and also $\mathrm{Lk}^+(v)$) is connected, and *saddle* otherwise. A vertex that is not regular is called a *critical* vertex. For simplicity, we assume that each saddle vertex v is *simple*, meaning that $\mathrm{Lk}^-(v)$ (and also $\mathrm{Lk}^+(v)$) consists of two connected components. If $\mathbb{M}$ contains a non-simple saddle, then we can split it into multiple simple saddles.

Level sets and contours. For $\ell \in \mathbb{R}$, the *ℓ-level set*, *ℓ-sublevel set*, and *ℓ-superlevel set* of $\mathbb{M}$, denoted by $\mathbb{M}_\ell$, $\mathbb{M}_{<\ell}$, $\mathbb{M}_{>\ell}$, respectively, consist of points $x \in \mathbb{M}$, with $h(x) = \ell$, $h(x) < \ell$, and $h(x) > \ell$, respectively. We refer to a level set $\mathbb{M}_\ell$ where $\ell = h(v)$ for some critical vertex v as a *critical level*. A *contour* of $\mathbb{M}$ is a connected component of a level set of $\mathbb{M}$. Each vertex $v \in V$ is contained in exactly one contour in $\mathbb{M}_{h(v)}$, which we call *the contour of v*. A contour not passing through a critical vertex is a simple polygonal cycle with non-empty interior. A contour passing through an extremal vertex is a single point, and by our assumption, a contour passing through a saddle consists of two simple cycles with the saddle vertex being their only intersection point. A contour C not passing through a vertex can be represented by the cyclic sequence of edges of $\mathbb{M}$ denoted by $\mathbb{E}(C)$, that it passes through. Two contours are called *combinatorially identical* if their cyclic sequences are the same.

Let $\varepsilon = \varepsilon(\Sigma)$ denote a sufficiently small positive value, in particular, smaller than the height difference between any two vertices of $\mathbb{M}$. An *up-contour* of a saddle vertex α is any contour of $\mathbb{M}_{h(\alpha)+\varepsilon}$ that intersects an edge incident on α. Similarly, a *down-contour* of α is any contour of $\mathbb{M}_{h(\alpha)-\varepsilon}$ that intersects an edge incident on α. If α has two up-contours and one down-contour it is called a *positive saddle vertex*. If it has two down-contours and one up-contour it is called a *negative saddle vertex*. A simple saddle v is either negative or positive.



(a) (b) (c)

Figure 1 (a) (b) An example terrain depicted with contours through saddle vertices and showing the critical vertices of the terrain: α_1 (α_5) is a blue (red) $-$ve saddle, and α_3 (α_4) is a lue (red) $+$ve saddle. (c) The contour tree of the terrain in (a).

Following the notation in [1], a contour C of $\mathbb{M}_\ell$ is called *blue* if points locally in the interior of C belong to $\mathbb{M}_{<\ell}$ and *red* otherwise. A positive (resp. negative) saddle vertex is colored by the color of its unique down-contour (resp. up-contour). Refer to Figure 1 to see the possible saddle colors.

Contour trees. Consider raising ℓ from $-\infty$ to ∞. The contours continuously deform, but no changes happen to the topology of the level set as long as ℓ varies between two consecutive critical levels. A new contour appears as a single point at a minimum vertex, and an existing contour contracts into a single point and disappears at a maximum vertex. An existing contour (the down-contour of v) splits into two new contours (the up-contours of v) at a positive saddle vertex v, and two contours (the down-contours of v) merge into one contour (the up-contour of v) at a negative saddle vertex v. The *contour tree* $\mathbb{T}$ of h is a tree on the critical vertices of $\mathbb{M}$ that encodes these topological changes of the level set. An edge (v, w) of $\mathbb{T}$ *represents* the contour that appears at v and disappears at w.

More formally, two contours C_1 and C_2 at levels ℓ_1 and ℓ_2, respectively, are called *equivalent* if C_1 and C_2 belong to the same connected component of $\Gamma = \{x \in \mathbb{R}^2 \mid \ell_1 \leq h(x) \leq \ell_2\}$ and that component of Γ does not contain any critical vertex. An equivalence class of contours starts and ends at critical vertices. If a class starts at a critical vertex v and ends at w, then (v, w) is an edge in $\mathbb{T}$. We refer to v as a *down neighbor* of w and to w as an *up neighbor* of v. Equivalently $\mathbb{T}$ is the quotient space in which each contour is represented by a point and connectivity is defined in terms of the quotient topology. Let $\rho : \mathbb{M} \to \mathbb{T}$ be the associated quotient map, which maps all points of a contour to a single point on an edge of $\mathbb{T}$. Fix a point p in $\mathbb{M}$. If p is not a critical vertex, $\rho(p)$ lies in the relative interior of an edge in $\mathbb{T}$; if p is an extremal vertex, $\rho(p)$ is a leaf node of $\mathbb{T}$; and if p is a saddle vertex then $\rho(p)$ is a non-leaf node of $\mathbb{T}$. See Figure 1.

We assume that each vertex of $\mathbb{T}$ is labeled with the corresponding critical vertex of $\mathbb{M}$. The combinatorial description of $\mathbb{T}$ is the set of its vertices along with their labels, and the set of its edges. We also consider augmenting the contour tree with regular vertices to produce the *augmented contour tree* $\mathbb{T}_A$. For each regular vertex v in $\mathbb{M}$, we insert a degree two vertex into $\mathbb{T}$ at $\rho(v)$.

Ascent and descent trees. We construct *descent trees* as follows: For each vertex $u \in \mathbb{M}$, if u is not a minimum then we choose a vertex $w \in \mathrm{Lk}^-(u)$ and create the edge (u, w). This process results in a forest of trees, each rooted at a minimum vertices of $\mathbb{M}$. We denote the descent tree rooted at x as $\Pi^{\downarrow}(x)$. Similarly, we construct a forest of *ascent trees*, obtained

by creating the edge (v, w) from every vertex v to a single vertex w in $\mathrm{Lk}^+(v)$ unless v is a maximum. Each ascent tree is rooted at a maximum vertex y and is denoted by $\Pi^{\uparrow}(y)$. Note that the descent tree forest partitions the vertices of $\mathbb{M}$ and the same is true for ascent trees. Lemma 1 below describes how the ascent and descent trees relate to the contour tree.

▶ **Lemma 1.** *Let v be a regular vertex in $\mathbb{M}$, and let x (resp. y) be the minimum (resp. maximum) of $\mathbb{M}$ such that $\Pi^{\downarrow}(x)$ (resp. $\Pi^{\uparrow}(y)$) contains v. Then the path P in $\mathbb{T}$ between x and y contains $\rho(v)$ and the heights of the vertices on P are monotone.*

3 Time Varying Contour Tree

Suppose the height function varies with time. That is, we have a one parameter family of height functions over $\mathbb{M}$, $h : \mathbb{M} \times \mathbb{R} \to \mathbb{R}$, where the extra dimension in the domain is time. In this section, we characterize how and when the contour tree of $(\mathbb{M}, h)$ changes with time. In particular, we describe all the combinatorial changes in the contour tree. For simplicity, we assume h to be *generic* in the sense: (i) at any given time t, at most two vertices have the same height; (ii) the heights of any two vertices become equal at a finite set of time values; and (iii) if $h(u, t_0) = h(v, t_0)$ then the function $h(u, t) - h(v, t)$ changes sign at $t = t_0$.

Edelsbrunner et al. [12] showed that if h is a smooth function, then the topology of $\mathbb{T}$ changes at time t if (i) a critical point u becomes degenerate (i.e., the Hessian at u becomes singular), or (ii) two saddle points u and v lie on the same contour. In either case, an edge (u, v) of $\mathbb{T}$ is degenerate in the sense that the interval $[h(u, t), h(v, t)]$ is a single point. In our setting, where $h(\cdot, t)$ is a piecewise-linear function, the two corresponding events are: (i) two adjacent vertices u and v with $h(u, t) = h(v, t)$, and one of them is a saddle and the other is an extremal vertex just before or after the event; (ii) two saddle vertices lie on the same contour. The former event is called a *birth* or a *death* event, and the latter is called an *interchange* event.

Besides these two events, there is another event in the piecewise-linear case, namely, a critical point shifts from one vertex to its neighbor — no new critical points are created, none is destroyed, and the topology of $\mathbb{T}$ does not change. Only the label of a node in $\mathbb{T}$ changes. We refer to this event as a *shift* event. We will refer to events occurring when the heights of two neighboring vertices become equal as *local* events. We note that an interchange event can also occur when the height of two adjacent vertices becomes equal, so a local event may correspond to an interchange event as well.

If an event occurs at time t, then we refer to t^- (resp. t^+) as the time $t - \varepsilon$ (resp. $t + \varepsilon$) for some arbitrarily small $\varepsilon > 0$.

Local events. Suppose a local event occurs at time t_0 at which $h(v, t_0) = h(u, t_0)$, where u is a neighbor of v. For simplicity, we assume that $h(v, t_0^-) < h(u, t_0^-)$ and $h(v, t_0^+) > h(u, t_0^+)$. The case when $h(v, t_0^-) > h(u, t_0^-)$ is symmetric. In the following sections we describe in detail the changes that occur in $\mathbb{T}$ during the three kinds of local events. We assume the interval $[t_0^-, t_0^+]$ to be sufficiently small so that there is no other vertex whose height lies between those of u and v during this interval.

Shift event. A shift event occurs at t_0 if one of u and v, say v, was a critical vertex and the other vertex, u, was a regular vertex at t_0^-, and the critical vertex shifts from v to u at t_0^+. This event does not cause any change in the topology of $\mathbb{T}$ but the node of $\mathbb{T}$ that was labeled v changes its label to u.

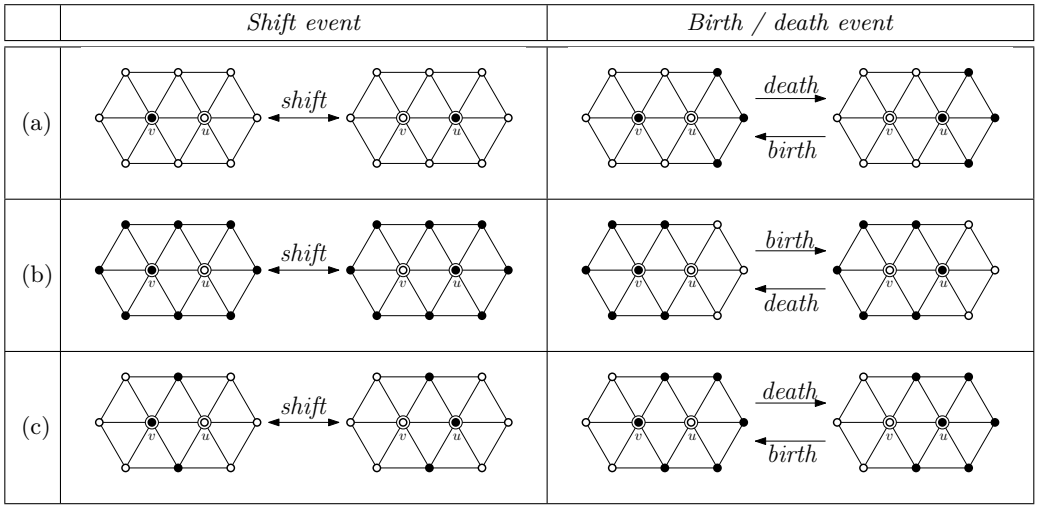


Figure 3 Illustration of local events: (a) v is a minimum vertex; (b) v is a regular vertex; (c) v is a saddle vertex. Hollow vertices are higher than u and v, and filled vertices are lower than u and v. In all examples v is raised, i.e., $h(v, t_0^-) < h(u, t_0^-)$ and $h(v, t_0^+) > h(u, t_0^+)$.

Birth/death event. A *birth event* occurs at time t_0 if both u and v were regular vertices at t_0^-, and they become critical vertices at t_0^+. A *death event* occurs when both u and v were critical vertices at t_0^- and become regular vertices at t_0^+. See Figure 2 for the change in the topology of $\mathbb{T}$. We now describe in detail how $\mathbb{T}$ changes at a birth event; a death event is similar. There are two possibilities: (i) v becomes a negative saddle and u a minimum, or (ii) v becomes a maximum and u a positive saddle. Suppose $\rho(u), \rho(v)$ lie on the edge (α, β) of $\mathbb{T}$. Then we split (α, β) into two edges by adding a node corresponding to the new saddle and creating a new edge incident on this node whose other endpoint is a leaf. In case (i), v is the node added on the edge (α, β) and u is the new leaf, and in (ii) u is the node on (α, β) and v is the new leaf.

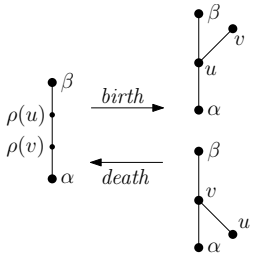


Figure 2 Illustration of the change in the topology of the contour tree in birth/death events.

A local event that corresponds to an interchange event is described in the next subsection. Using an exhaustive case analysis (omitted from this version), it can be shown that the above description includes all possible changes in the contour tree that may be caused by a local event. See Figure 3.

3.1 Interchange events

An interchange event occurs at time t_0 if two saddle vertices α, β lie on the same contour, i.e., $\rho(\alpha) = \rho(\beta)$, at time t_0. Suppose $h(\alpha, t_0^-) < h(\beta, t_0^-)$. There are four cases, depending on whether α and β are positive or negative saddles. Let us assume that α is a negative saddle. (As we will see below the case when α is positive can be reduced to this case by reversing the z-axis and/or time axis.) Then there are two cases: (i) α is negative and β is positive , and (ii) both α and β are negative saddles. We refer to them as *mixed* and *negative* interchange events, respectively. We need a few notations to describe the interchange events.

At time t_0^- all contours in the equivalence class (α, β) are combinatorially identical because $\rho(w)$, for any vertex w of $\mathbb{M}$, does not lie on the interior of the edge (α, β) of $\mathbb{T}$.

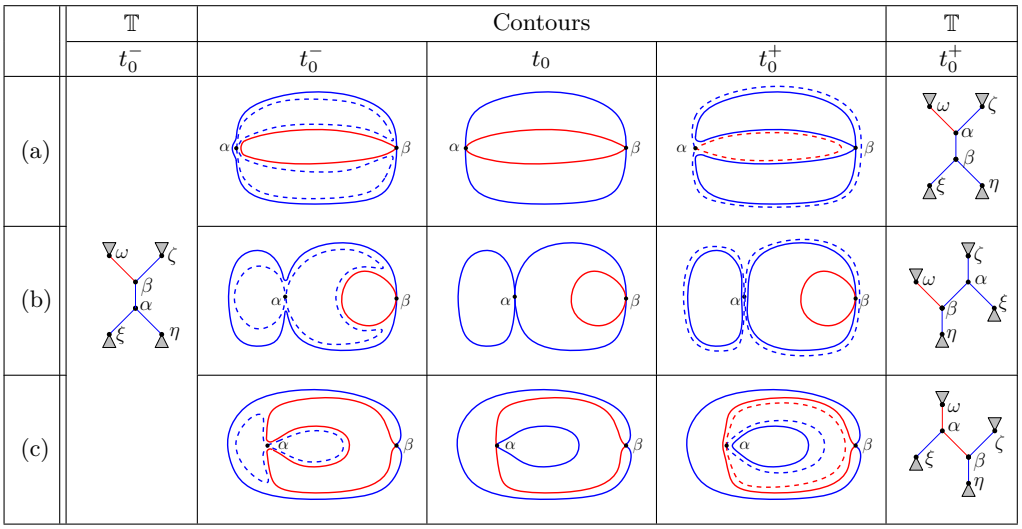


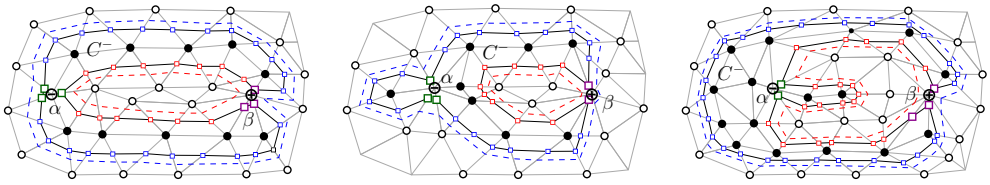
Figure 4 Illustration of topological changes in $\mathbb{M}_\ell$ and contour tree transitions during a mixed interchange event. Dashed contour lines are contours of α and solid contour lines are contours of β at time t_0^- and t_0^+; each of them consists of two simple cycles. The two contours merge into one contour at time t_0 (middle column). (a) a sign change event, (b) a blue event, (c) a red event.

With a slight abuse of notation, we will therefore simply refer to all contours in the class (α, β) as the contour C^- without specifying a height. Similarly, at time t_0^+, all contours in the class (β, α) are combinatorially identical, and we refer to them as C^+. We label the vertices of C^- and of C^+ that lie on edges incident to α (resp. β) with α (resp. β).

Mixed interchange event. In this case, α is negative and β is positive. We assume that the edge (α, β) is blue at t_0^-, so both α and β are blue at t_0^- (see Section 2). The case when (α, β) is red reduces to this case by reversing the direction of the z-axis. Let ξ and η be the two down neighbors of α, and let ζ, ω be the up neighbors of β at time t_0^-. Since α is blue, both (ξ, α) and (η, α) are blue edges. Since β is blue, one of (β, ζ) and (β, ω) is red and the other is blue (See Figure 1). Assume that the edge (β, ζ) is blue and (β, ω) is red, i.e., the up-contour C_ω^- of β is red and the up-contour C_ζ^- of β is blue. Refer to Figure 4.

Since α is a negative saddle, β is a positive saddle, and C^- intersects both components of $\mathrm{Lk}^+(\alpha)$ and of $\mathrm{Lk}^-(\beta)$, the vertices of C^- labeled with α form two disconnected intervals in C^-, and the same is true for vertices labeled with β.[1] Since β is the only vertex between C^- and the up-contours C_ζ^- and C_ω^- of β, every vertex of C^- either lies on an edge of $\mathbb{E}(C_\zeta^-) \cup \mathbb{E}(C_\omega^-)$ or is labeled with β. We mark the vertices of C^- that lie on the edges of $\mathbb{E}(C_\omega^-)$ as red and the vertices that lie on the edges of $\mathbb{E}(C_\zeta^-)$ as blue, in accordance with the colors of the contours C_ω^- and C_ζ^-, respectively. Refer to Figure 5. Similarly, let C_ξ^- and C_η^- be the down-contours of α, then every vertex of C^- either lies on an edge of $\mathbb{E}(C_\xi^-) \cup \mathbb{E}(C_\eta^-)$ or is labeled with α. There are three types of mixed interchange events depending on the relative positions of the vertices of C^- that are marked α or β. See Figure 5.

[1] If α and β are adjacent in $\mathbb{M}$, there is a vertex of C^- (the intersection point of C^- with the edge $\alpha\beta$ of $\mathbb{M}$) that is marked both α and β. In this case, we simply consider vertices marked only α (resp. β). It can be shown that β is an endpoint of a connected component of $\mathrm{Lk}^+(\alpha)$ and that component contains more than one vertex. Therefore the vertices of C^- marked only α form two disconnected intervals in C^-, and a similar argument applies to β.



(a) (b) (c)

Figure 5 Illustration of the possible colorings of C^- for mixed interchange events. α is a negative saddle and β is positive. Green (resp. purple) contour vertices represent the vertices marked with α (resp. β). Hollow vertices of $\mathbb{M}$ have height higher than both α and β, and filled vertices have lower height. Blue dashed lines indicate C^-_ζ, and red dash lines C^-_ω. (a) Vertices marked α in C^- lie on the edges of both $\mathbb{E}(C^-_\zeta)$ and $\mathbb{E}(C^-_\omega)$, (b) vertices marked α lie on the edges of $\mathbb{E}(C^-_\zeta)$, (c) vertices marked α lie on the edges of $\mathbb{E}(C^-_\omega)$.

(i) Vertices marked α and β in C^- are interleaved, i.e., vertices marked α are both red and blue; we refer to this as a *sign-interchange* event.
(ii) All vertices marked α are blue; we refer to this as a *blue* event.
(iii) All vertices marked α are red; we refer to this as a *red* event.

In case (ii) and (iii), all vertices marked β lie in one of $\mathbb{E}(C^-_\xi)$ and $\mathbb{E}(C^-_\eta)$. Without loss of generality, assume that they lie on the edges of $\mathbb{E}(C^-_\eta)$. The following lemma then characterizes the change of $\mathbb{T}$ at a mixed interchange event.

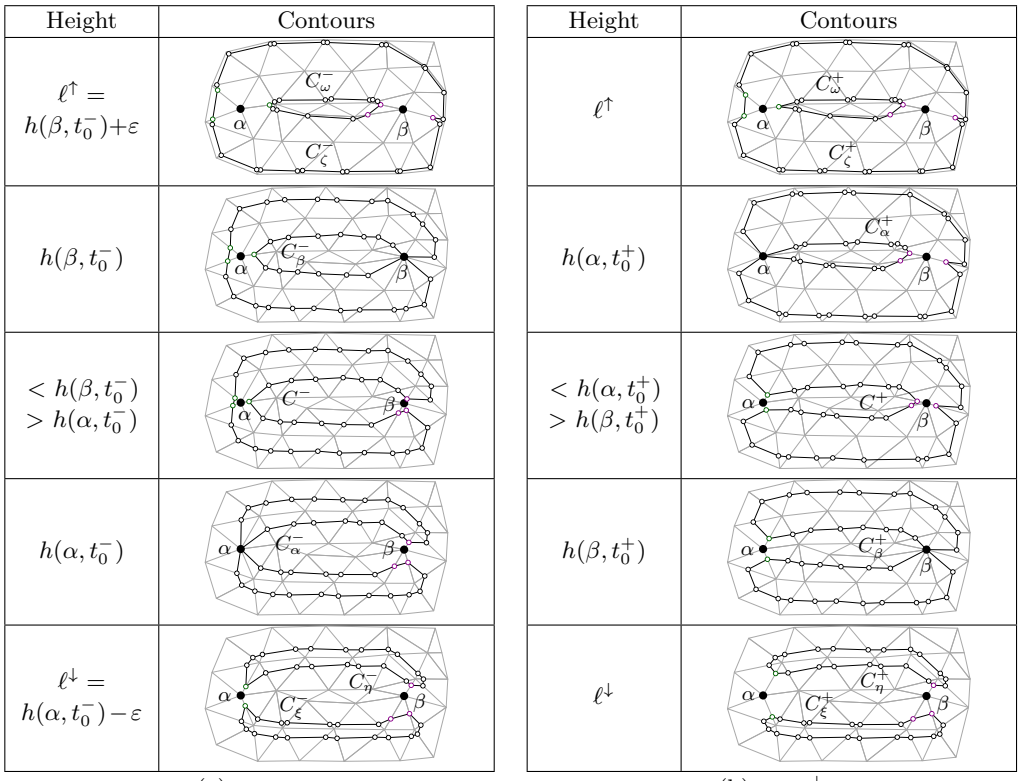
▶ **Lemma 2.** *Suppose a mixed interchange event occurs at time t_0 involving an edge (α, β) of $\mathbb{T}$. If the edge (α, β) is blue at t_0^- and $h(\alpha, t_0^-) < h(\beta, t_0^-)$, then the following change occurs in $\mathbb{T}$ at t_0:*

(i) *Sign-interchange event: The topology of $\mathbb{T}$ does not change. The only change in $\mathbb{T}$ is that the label α and β of $\mathbb{T}$ get swapped, i.e., the edge (α, β) becomes (β, α), and α (resp. β) becomes a positive (resp. negative) saddle at t_0^+; Figure 4 (a).*
(ii) *Blue event: The edge (α, β) becomes (β, α) but the color of (β, α) and of the saddles α, β remain blue. At time t_0^+, η is the down neighbor of β, α and ω are the up neighbors of β, ξ and β are the down neighbors of α, and ζ is the up neighbor of α; Figure 4 (b).*
(iii) *Red event: The edge (α, β) becomes (β, α), and its color becomes red and so does the saddle α. At time t_0^+, η is the down neighbor of β, α and ζ are the up neighbors of β, ξ and β are the down neighbors of α, and ω is the up neighbor of α; Figure 4 (c).*

Proof. Consider the sign-interchange event. Let C^-_ξ, C^-_η and C^-_ζ, C^-_ω be the down-contours of α and the up-contours of β at time t_0^-, respectively. Let $\ell^\downarrow = h(\alpha, t_0^-) - \varepsilon$ and $\ell^\uparrow = h(\beta, t_0^-) + \varepsilon$ be the levels of the down-contours of α and of the up-contours of β, respectively. We can choose the values of t_0^+ and ε carefully so that $\ell^\uparrow > h(\alpha, t_0^+)$.

First we fix the time t_0^- and monitor the contour C^- as we decrease the level toward $h(\alpha, t_0^-)$. The vertices of C^- marked α converge to the vertex α at height $h(\alpha, t_0^-)$, and we obtain the contour C^-_α of α. Since the vertices marked α and β in C^- are interleaved, each of the two simple cycles of C^-_α contains the vertices marked β (i.e., each of them intersects the edges incident on β), and therefore so do C^-_ξ and C^-_η. Similarly, each of C^-_ζ and C^-_ω contains vertices marked α. Refer to Figure 6 (a).

Next we fix the heights $\ell^\downarrow$ and $\ell^\uparrow$ and move forward in time from t_0^- to t_0^+. As time moves toward t_0^+, C^-_ξ, C^-_η, C^-_ζ and C^-_ω continuously deform but no topological changes occur to any of them. Let us consider the deformation of C^-_ξ. The deformation can be represented by a continuous function $F : \mathbb{S} \times [t_0^-, t_0^+] \to \mathbb{R}^2$, where $F(\cdot, t_0^-) = C^-_\xi$ and $F(\cdot, t)$ denote the



(a) $t = t_0^-$ (b) $t = t_0^+$

Figure 6 Illustration of a sign-change event. The vertices marked α are illustrated as green vertices and the vertices marked β as purple vertices.

deformation of C_ξ^- at time $t \in [t_0^-, t_0^+]$. By construction, $\mathbb{E}(F(\cdot, t))$ remains the same for all $t \in [t_0^-, t_0^+]$. Let $C_\xi^+ = F(\cdot, t_0^+)$. Similarly define C_η^+, C_ζ^+, and C_ω^+. See Figure 6 (b).

Now we fix the time to t_0^+ and increase the height from $\ell^\downarrow$ to $\ell^\uparrow$ and monitor how the contours C_ξ^+ and C_η^+ deform. Recall that $h(\beta, t_0^+) < h(\alpha, t_0^+)$, so as we increase the height, we first encounter β and the vertices of C_ξ^+ and C_η^+ marked β converge to the vertex β of $\mathbb{M}$. Hence, β is now a negative saddle. Since both C_ξ^+ and C_η^+ are blue, so is the saddle β at t_0^+. C^+, the up-contour C^+ of β at time t_0^+, is therefore a blue contour. $\mathbb{E}(C^+)$ contains all edges of $\mathbb{E}(C_\xi^+) \cup \mathbb{E}(C_\eta^+)$ that are not incident on β plus the edges whose lower endpoint is β (at time t_0^+). Furthermore the vertices marked α and β in C^+ are interleaved (as in C^-). If we continue to increase the height, the vertices of C^+ marked α converge to α as we reach $h(\alpha, t_0^+)$, and C^+ splits into two contours at α. Since C^+ is a blue contour, α is a blue positive saddle at t_0^+. The up-contours of α at time t_0^+ will be C_ζ^+ and C_ω^+, so ζ and ω will be up-neighbors of α. See Figure 6 (b). This completes the proof for the sign-interchange case.

Next, suppose the red or blue event occurs at time t_0. The proof proceeds along the same lines as for the previous case. Let $\ell^\downarrow$, $\ell^\uparrow$, C_ξ^+, C_η^+, C_ζ^+, and C_ω^+ be the same as above. If the vertices marked α and β are not interleaved in C^-, then by our assumption the vertices marked β lie on the edges of $\mathbb{E}(C_\eta^+)$, these vertices converge to β at height $h(\beta, t_0^+)$ and C_η^+ splits into two contours at β. Note that as we increase the height C_ξ^+ also deforms but does not meet β, as it has no vertices marked β. Hence β is a blue positive saddle, with η as the down neighbor of β. Let C_β^+ be the contour of β at time t_0^+.

C_β^+ consists of two simple cycles, one with vertices lying on edges of $\mathbb{E}(C_\omega^+)$ and the other with vertices lying on edges of $\mathbb{E}(C_\zeta^+)$. As we increase the height to $h(\alpha, t_0^+)$, the vertices

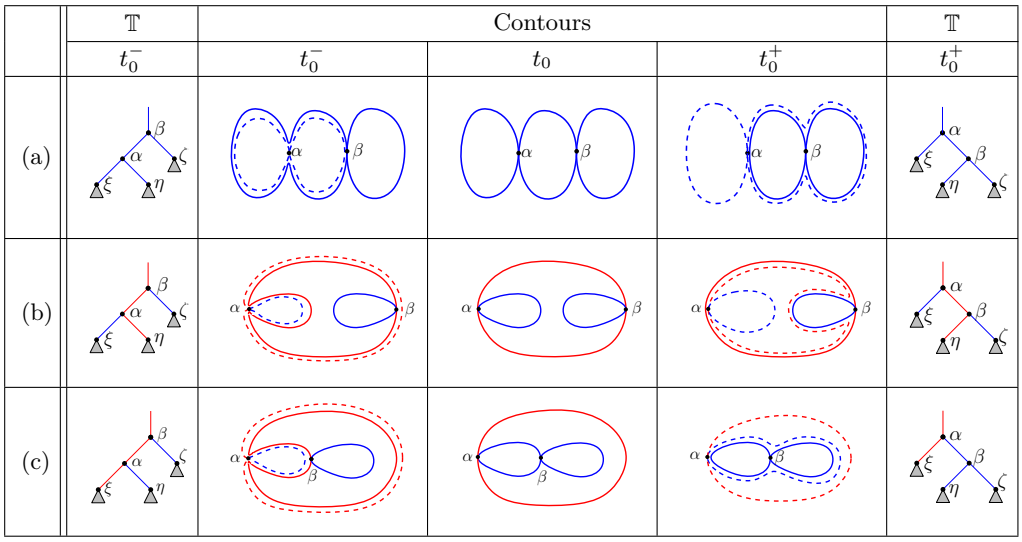


Figure 7 Illustration of topological changes in $\mathbb{M}_\ell$ and contour tree transitions during negative interchange events. Dashed lines are contours of α and solid lines are contours of β. (a) is when β is blue. (b) and (c) are when β is red.

marked α in C^+ and C^+_ξ converge to α, and the two contours merge into a single contour C^+_α at α. If the blue (resp. red) event occurs at t_0, i.e., vertices marked α lie on the edges of $\mathbb{E}(C_\zeta)$ (resp. $\mathbb{E}(C_\omega)$), then α is a blue (resp. red) negative saddle at time t_0^+, the up-contour of α is C^+_ζ (resp. C^+_ω), and the contour C^+_ω (resp. C^+_ζ) remains an up-contour of β. See Figure 4 (b) and (c). ◀

The above lemma characterizes the changes in the contour tree at a mixed interchange event under the assumption that α was a blue negative saddle at t_0^-. As mentioned above, the other cases can be reduced to the above case. In particular, if α is a red negative saddle at time t_0^-, reverse the direction of the z-axis. Now β becomes a blue negative saddle, α a blue positive saddle, and β lies below α, which is precisely the case described in Lemma 2. If α is a blue positive saddle, reverse the direction of time and swap β and α (see Figure 4 (b,c)). Finally, if α is red positive saddle, then by reversing the direction of time as well as that of the z-axis, we reduce this case to that described in Lemma 2.

Negative interchange event. Let ξ, η be the two down neighbors of α at t_0^-, and let ζ be the other down neighbor of β (α is a down neighbor of β at t_0^-). See Figure 7. The change in topology of $\mathbb{T}$ at a negative interchange event is similar to performing a rotation at node β. That is, the edge (α, β) becomes the edge (β, α) at time t_0^+, and one of the down subtrees of α (rooted in ξ and η) becomes a down subtree of β. Next we describe the change of $\mathbb{T}$ in more detail.

Let C^-_ξ and C^-_η be the down-contours of α at time t_0^-. Since both α and β are negative saddles and C^- is the up-contour of α and one of the down contour of β (at time t_0^-), the vertices of C^- labeled α form two disconnected intervals and those labeled β form a single connected interval. Since α is the only vertex between C^- and the down-contours of α, vertices of C^- either lie in the interior of an edge in $\mathbb{E}(C^-_\xi) \cup \mathbb{E}(C^-_\eta)$, or is labeled with α. Furthermore the vertices of C^- marked β form a connected interval, the edges incident on β that intersect C^- belong to one of $\mathbb{E}(C^-_\xi)$ and $\mathbb{E}(C^-_\eta)$ but not both. Using these observations,

Figure 7 illustrates different possible cases at a negative interchange event. Lemma 3 below describes how the topology of $\mathbb{T}$ changes. Its proof is similar to that of Lemma 2 and is omitted from this version.

▶ **Lemma 3.** *If the vertices of C^- labeled β lie on the edges of $\mathbb{E}(C_\eta^-)$, then at time t_0^+, η becomes a down neighbor of β, ζ the other down neighbor of β, and ξ and β become the down neighbors of α.*

Figure 7 also illustrates how the colors of α and β change at t_0^+. Indeed, if β is blue at t_0^- then α is also blue at t_0^-, and both of them remain blue at t_0^+ (Figure 7 (a)). If β is red, then α is blue or red. Suppose α is red. If η is red at t_0^- (recall we assume that the edges incident to β that intersect C^- belong to $\mathbb{E}(C_\eta^-)$), then both α and β remain red at t_0^+ (Figure 7 (b)). But if η is blue at t_0^- then α remains red but β becomes blue at t_0^+ (Figure 7 (c)). Finally, if β is red and α is blue at t_0^-, then this corresponds to third case if we reverse the direction of the time axis (see Figure 7 (c), with the roles of α and β being swapped).

4 KDS for $\mathbb{T}$

In this section, we describe an algorithm for maintaining the contour tree as the height function varies over time, using the KDS framework by Basch et al. [5]. The KDS framework maintains a set of *certificates* that ensure the correctness of the configuration. As the objects move, the certificates fail at certain time instances, called *events*. For each certificate, the closest time that the certificate fails is computed and maintained in an *event queue*. When the current time reaches the time of the next event in the event queue, a repair mechanism is invoked to update the configuration, the set of certificates, and the event queue.

We represent the time-varying height function by specifying the height of each vertex $v \in V$ as a function $f_v : \mathbb{R} \to \mathbb{R}$ of time. We assume that each f_v is a polynomial and that the maximum degree of these polynomials is bounded by a constant. We also assume that the set $\{f_v \mid v \in V\}$ is generic in the sense described in the beginning of Section 3. We describe the algorithm in the real-RAM model [8], in which various operations on constant-degree polynomials (e.g., finding the roots) can be performed in $O(1)$ time.

Data structure. To maintain $\mathbb{T}$ efficiently, we also maintain: (i) a descent tree $\Pi^{\downarrow}(x)$ for every minimum x; (ii) an ascent tree $\Pi^{\uparrow}(y)$ for every maximum y; (iii) for every non-extremal vertex v, a *link pointer* to the first vertex of each connected component of $\mathrm{Lk}^-(v)$ and $\mathrm{Lk}^+(v)$, in clockwise order. We store $\mathbb{M}$ using a standard triangulation data structure such as DCEL [7]. We maintain an *event queue* as a priority queue so that the next event can be computed efficiently. We represent the contour tree, the ascent trees, and the descent trees as *link-cut trees* [16], which support each of the following operations in $\mathrm{O}(\log n)$ time.

- Link(v, w): Given two vertices v, w, connect the trees containing v and w by inserting the edge (v, w).
- Cut(v, w): Given v, w, split the tree containing v and w by removing the edge (v, w).
- Root(v): Return the root of the tree containing v.
- Expose(v, w): Return a pointer to a balanced binary search tree storing vertices on the path P from v to w in the order as they appear on P.

These operations enable the following operations on $\mathbb{T}$: (a) NextTo(v, w): Given two critical vertices v and w, return the vertex adjacent to v in $\mathbb{T}$ on the path from v to w. (b) FindEdge(v): For a regular vertex v, return the edge (α, β) of $\mathbb{T}$ that contains $\rho(v)$.

NextTo is straightforward and takes $O(\log n)$ time. FindEdge(v) is implemented as follows. Using Root(v) on the ascent trees and the descent trees, we find the minimum x and the maximum y such that $v \in \Pi^{\downarrow}(x)$ and $v \in \Pi^{\uparrow}(y)$. By Lemma 1, (α, β) is on the path P between x and y in $\mathbb{T}$, and the heights of the vertices on P are monotone. We obtain P using Expose(x, y) and then find (α, β) on P. This procedure takes $O(\log n)$ time.

Certificates. Each edge in an ascent/descent tree is an edge of $\mathbb{M}$ and the link pointers also correspond to the edges of $\mathbb{M}$, so these auxiliary structures change only when the heights of two endpoints of an edge become equal. The time instance at which an ascent/descent tree or a link pointer needs to be updated is referred to as *auxiliary* event. As described in Section 3, $\mathbb{T}$ changes only when the heights of the endpoints of an edge of $\mathbb{M}$ or of $\mathbb{T}$ become equal. Hence, our KDS maintains the following two sets of certificates to certify the correctness of the structures at any given time: (i) a certificate for each edge (u, v) of $\mathbb{M}$, that fails when $h(v, t) = h(u, t)$, corresponding to a local event; and (ii) a certificate for each edge (α, β) of $\mathbb{T}$ that fails when $h(\alpha, t) = h(\beta, t)$, corresponding to an interchange event.

Initialization. We initialize by constructing $\mathbb{T}$ at time zero, using the static algorithm [9], in $O(n \log n)$ time. The ascent/descent trees and the link pointers can be initialized in $O(n \log n)$ time. Finally, we initialize the event queue with the certificates associated with edges of $\mathbb{M}$ and $\mathbb{T}$.

Repair mechanism. We now describe how we update the KDS at each event. Besides $\mathbb{T}$, we also have to update the auxiliary structures and the event queue. First consider a local event at time t_0 at which $h(v, t_0) = h(u, t_0)$, where (u, v) is an edge of $\mathbb{M}$. Assume that $h(v, t_0^-) < h(u, t_0^-)$ and $h(v, t_0^+) > h(u, t_0^+)$. This event may be an auxiliary, shift, birth/death, or an interchange event (or multiple of them). Processing a shift event is straightforward as it only involves updating the label of a node of $\mathbb{T}$ (see Section 3) and the certificates, along with their failure times, for the edges of $\mathbb{T}$ adjacent to that node of $\mathbb{T}$. We will describe the processing of an interchange event later, so we briefly sketch the processing of auxiliary and birth/death events.

Birth/death event. Suppose u, v were regular vertices at t_0^- and become critical vertices at t_0^+. We find the edge (α, β) of $\mathbb{T}$ that contains u, v using the FindEdge(u) operation. Once we know (α, β), $\mathbb{T}$ can be updated in $O(\log n)$ time, using Link and Cut operations. Finally, we update the certificates and their failure times corresponding to the new edges in $\mathbb{T}$. Processing a death event is similar but simpler since u, v are vertices of $\mathbb{T}$ at t_0^-.

Auxiliary event.] Updating the link pointers of u and v is straightforward. Next suppose (v, u) is an edge of an ascent tree at t_0. We remove the edge (v, u). If v becomes a maximum at t_0^+, then v becomes the root of an ascent tree; otherwise we choose another vertex w from $\mathrm{Lk}^+(v)$ and add the edge (v, w). If u was a maximum at t_0^-, i.e., u was the root of an ascent tree, we add the edge (u, v). These changes in the ascent tree can be performed in $O(\log n)$ time using Link, Cut, and Root operations. A descent tree is updated in an analogous manner.

Interchange event. We describe how to process a mixed interchange event; other interchange events can be processed similarly. Following the set up in Section 3, suppose a mixed interchange event occurs at t_0 at which $h(\alpha, t_0) = h(\beta, t_0)$ with $h(\alpha, t_0^-) < h(\beta, t_0^-)$, α being a blue negative saddle at t_0^-, and β being a blue positive saddle at t_0^-. Once we know whether this event is sign-interchange, blue, or red event, we can update $\mathbb{T}$ in $O(\log n)$ time in accordance with the characterization of events in Lemma 2.

Borrowing the notation from Section 3, let ζ, ω be the two up neighbors of β at t_0^- with (β,ζ) being blue and (β,ω) being red. Similarly let C^- be a contour of the edge (α,β), C_ζ^- (resp. C_ω^-) the up-contour of β corresponding to the edge (β,ζ) (resp. (β,ω)). The vertices of the contour C^- labeled α consist of two disconnected intervals, say, I_1 and I_2, each corresponding to one connected component of $\mathrm{Lk}^+(\alpha)$ at t_0^-. We also note that all edges of $\mathbb{E}(C^-)$ that contain the one connected interval of vertices of C^- marked α belong to $\mathbb{E}(C_\zeta^-)$ or $\mathbb{E}(C_\omega^-)$ but not both. For each of I_1 and I_2, we determine which of the two sets, $\mathbb{E}(C_\zeta^-)$ or $\mathbb{E}(C_\omega^-)$, these edges belong to. Consider I_1 and choose an arbitrary vertex from this interval. Suppose this vertex lies on the edge $(\alpha,u_1)\in\mathbb{M}$; u_1 can be identified using the link pointers of α. Using the $\textsc{Root}(u_1)$ procedure, we determine the maximum y such that $u_1\in\Pi^{\uparrow}(y)$. Consider the path $P_{\beta y}$ from β to y in $\mathbb{T}$. Let $\gamma_1\in\{\zeta,\omega\}$ be the vertex next to β in $P_{\beta y}$; γ_1 can be determined by calling $\textsc{NextTo}(\beta,y)$. Then the edge $(\alpha,u_1)\in\mathbb{E}(C_{\gamma_1}^-)$. Similarly we choose a vertex u_2 from the other component of $\mathrm{Lk}^+(\alpha)$ and find the vertex $\gamma_2\in\{\zeta,\omega\}$ such that $(\alpha,u_2)\in\mathbb{E}(C_{\gamma_2}^-)$. If $\gamma_1\neq\gamma_2$, then the event at hand is a sign-interchange event, if $\gamma_1=\gamma_2=\zeta$ then it is a blue event, and if $\gamma_1=\gamma_2=\omega$ then it is a red event. The total time spent in computing γ_1 and γ_2 is $O(\log n)$.
Hence, a mixed interchange event can be processed in $O(\log n)$ time. Similarly other interchange events can be processed in $O(\log n)$ time.

KDS analysis. The shift, birth/death, and interchange events are *external* events as they change $\mathbb{T}$, and the auxiliary events are *internal* events as they only change auxiliary data structures and do not affect $\mathbb{T}$. Since auxiliary events are local events, the number of internal events is $\mathrm{O}(n)$. Hence, the KDS processes $\mathrm{O}(\kappa+n)$ events, where κ is the number of external events. Each certificate of the KDS is associated with an edge of $\mathbb{M}$ or $\mathbb{T}$, so it maintains $\mathrm{O}(n)$ certificates at any time. Each event will cause the update of $O(1)$ number of certificates. The maximum number of certificates in which any one vertex can ever appear is bounded by the degree of the vertex in $\mathbb{M}$. Although the degree of a vertex in $\mathbb{M}$ can be $\Omega(n)$ in the worst-case, it is often a small constant in practice.

▶ **Theorem 4.** *Let $\mathbb{M}$ be a triangulation of $\mathbb{R}^2$ with n vertices, and $h:\mathbb{M}\times\mathbb{R}\to\mathbb{R}$ a time-varying height function such that the height of each vertex of $\mathbb{M}$, as a function of time, is specified by a polynomial whose maximum degree is bounded by a constant. The contour tree of h can be maintained by a linear-size KDS that processes $O(\kappa+n)$ events, where κ is the number of external events. The KDS processes each event in $O(\log n)$ time in the real-RAM model.*

▶ Remark. (i) It is easy to construct a height function so that the number of combinatorial changes in $\mathbb{T}$ is $\Omega(n^2)$. As the worst-case number of internal events is much smaller than the worst-case number of external events, our KDS is *weakly efficient* in the terminology of KDS [14]. (ii) If the maximum degree of a vertex is large, our KDS can be modified so that a vertex appears in $O(\log n)$ certificates, but the number of internal events increases to $O(\lambda_s(n)\log n)$, where s is a constant and $\lambda_s(\cdot)$ is the maximum length of an order s Davenport-Schinzel sequence and is almost linear [2]. Each event now takes $O(\log^2 n)$ time to process.

5 Extensions

In this section, we describe kinetic data structures for maintaining the augmented contour tree and join/split tree.

Augmented contour tree. Recall that the augmented contour tree $\mathbb{T}_A$ is obtained by adding to the edges of $\mathbb{T}$ the images of regular vertices of $\mathbb{M}$ under the quotient map ρ. We refer to these newly added degree two vertices in $\mathbb{T}_A$ as *regular vertices* as well, while the original vertices of $\mathbb{T}$ are called *critical vertices*. We can easily modify the KDS framework described in Section 4 to maintain this augmented contour tree $\mathbb{T}_A$. We provide a brief description below.

Specifically, the only new events that we need to handle are *(regular-regular event)*: when two regular vertices u, v lie on the same contour, and *(regular-critical event)*: when a regular vertex u and a critical vertex v lie on the same contour. Note that (u, v) is an edge of $\mathbb{T}_A$ in both cases. When a regular-regular event happens, there are two possibilities: (i) either u and v swap their order along the edge (u, v); or (ii) it causes a birth event, in which case (u, v) is also necessarily an edge in $\mathbb{M}$.

At a regular-critical event at time t_0, there are two cases: (i) u is a regular vertex and v is an extremal vertex. In this case, it is easy to verify that (u, v) has to also be an edge in $\mathbb{M}$, and it corresponds to a shift event. (ii) u is a regular vertex and v is a saddle vertex. If it does not correspond to a shift event, we need to identify which edge incident on v in $\mathbb{T}_A$ contains u after the event. Assume $h(u, t_0^-) < h(v, t_0^-)$, and let w be a vertex in $\mathrm{Lk}^+(u)$. It is easy to see that w and u are in the same connected component in $\mathbb{M}_{>(h(v,t_0^+))}$ at time t_0^+. Thus, u is moved onto the incident edge of v whose endpoint is the one obtained by $\textsc{NextTo}(v, y)$, where $\Pi^{\uparrow}(y)$ contains w. If $h(u, t_0^-) > h(v, t_0^-)$, we consider w as a vertex in $\mathrm{Lk}^-(u)$ and use the descent trees.

The additional events can be clearly handled in $\mathrm{O}(\log n)$ time. Note that when a local event associated with an edge (u, v) of $\mathbb{M}$ occurs, there is an edge (u, v) in $\mathbb{T}_A$ and thus certificates associated with edges of $\mathbb{M}$ are unnecessary. Each certificate failure makes a combinatorial change in $\mathbb{T}_A$. Therefore, this data structure is strongly efficient.

▶ **Theorem 5.** *Let $\mathbb{M}$ be a triangulation of $\mathbb{R}^2$ with n vertices, and $h : \mathbb{M} \times \mathbb{R} \to \mathbb{R}$ a time-varying height function such that the height of each vertex of $\mathbb{M}$, as a function of time, is specified by a polynomial whose maximum degree is bounded by a constant. The augmented contour tree of h can be maintained by a linear-size KDS that processes $O(\kappa)$ events, where κ is the number of external events. The KDS processes each event in $O(\log n)$ time in the real-RAM model.*

Join and split trees. As described, the contour tree $\mathbb{T}$ encodes the topological changes in $\mathbb{M}_\ell$ as we increase ℓ from $-\infty$ to ∞. The *join tree* encodes the topological changes in $\mathbb{M}_{<\ell}$. As we increase ℓ from $-\infty$ to ∞, connected components in $\mathbb{M}_{<\ell}$ appear at minimum vertices and merge at negative saddle vertices, but they never split or disappear as components never shrink. In the join tree, only minimum vertices and negative saddle vertices appear as nodes. Similarly, the *split tree* encodes the topological changes in $\mathbb{M}_{>\ell}$, in which only maxima and positive saddles appear as nodes. As the join and split trees contain a subset of nodes in the contour tree, the events that occur on these trees of time-varying height functions are a subset of the events that occur on the contour tree. For example, the events that occur on the join tree are the events that involve the minima and negative saddle vertices, i.e., the negative interchange events, sign-interchange change events and a subset of the local events. The algorithm described above can therefore also be used to maintain the join and split trees, the only difference being the reduced set of events. Note that to be able to detect the sign-interchange events efficiently, we need to maintain $\mathbb{T}$ simultaneously with the join tree $\mathbb{J}$ or augment the positive saddle vertices to the edges of $\mathbb{J}$.

▶ **Theorem 6.** *Let $\mathbb{M}$ be a triangulation of $\mathbb{R}^2$ with n vertices, and $h : \mathbb{M} \times \mathbb{R} \to \mathbb{R}$ a time-varying height function such that the height of each vertex of $\mathbb{M}$, as a function of time, is specified by a polynomial whose maximum degree is bounded by a constant. The join tree and the split tree of h can be maintained by a linear-size KDS that processes $O(\kappa + n)$ events, where κ is the number of events at which the contour tree of h changes. The KDS processes each event in $O(\log n)$ time in the real-RAM model.*

6 Conclusion

In this paper, we characterized the combinatorial changes in the evolution of the contour tree of a time-varying piecewise linear height function over $\mathbb{R}^2$, and described the first KDS for maintaining the contour tree that efficiently handled such changes. Adapting this KDS, we also presented how to maintain the augmented contour tree and join/split tree.

Our analysis of events and our KDS are limited to a height function on simple 2-manifolds. A natural question of course is to extend this analysis and the KDS to a height function on higher dimensional space, say, a simple 3-manifold. A more challenging question is to develop an efficient KDS for maintaining the Reeb graph of a function over a 2-manifold with non-zero genus. Finally, another interesting question is to design a KDS for maintaining the contour tree when not only the height function changes with time but $\mathbb{M}$ itself changes over time.

References

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

2 P. K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. Elsevier Science Publishers, 2000.

3 Pankaj K. Agarwal, Lars Arge, and Ke Yi. I/O-efficient batched union-find and its applications to terrain analysis. In *Proc. 22nd Annu. Sympos. Comput. Geom.*, pages 167–176, 2006.

4 Lars Arge, Morten Revsbæk, and Norbert Zeh. I/O-efficient computation of water flow across a terrain. In *Proc. 26th Annu. Sympos. Comput. Geom.*, pages 403–412, 2010.

5 J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

6 K.G. Bemis, D. Silver, P.A. Rona, and C. Feng. Case study: a methodology for plume visualization with application to real-time acquisition and navigation. In *Proc. IEEE Conf. Visualization*, pages 481–494, 2000.

7 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

8 Vasco Brattka and Peter Hertling. Feasible real random access machines. *J. Complexity*, 14(4):490–526, December 1998.

9 Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Comput. Geom.*, 24(2):75–94, 2003.

10 Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Comput. Geom.*, 43(1):42–58, 2010.

11 A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitásová. Terrastream: From elevation data to watershed hierarchies. In *Proc. ACM Sympos. Advances in Geographic Information Systems*, page 28, 2007.

12 Herbert Edelsbrunner, John Harer, Ajith Mascarenhas, Valerio Pascucci, and Jack Snoeyink. Time-varying Reeb graphs for continuous space-time data. *Comput. Geom.*, 41(3):149–166, 2008.

13 L. Guibas. Modeling motion. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 1117–1134. Chapman and Hall/CRC, 2nd edition, 2004.

14 Leonidas J. Guibas. Kinetic data structures – a state of the art report. In *Proc. Workshop Algorithmic Found. Robot.*, pages 191–209, 1998.

15 N. Max, R. Crawfis, and D. Williams. Visualization for climate modeling. *IEEE Comput. Graphics and Appl.*, 13:34–40, 1993.

16 Daniel D. Sleator and Robert E. Tarjan. A data structure for dynamic trees. *J. Comput. Sys. Sci.*, 26(3):362–391, 1983.

17 B. S. Sohn and Bajaj C. L. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006.

18 A. Szymczak. Subdomain aware contour trees and contour evolution in time-dependent scalar fields. In *Proc. Conf. Shape Model. and Appl.*, pages 136–144, 2005.

19 S. P. Tarasov and M. N. Vyalyi. Construction of contour trees in 3D in $O(n \log n)$ steps. In *Proc. 14th Annu. Sympos. Comput. Geom.*, pages 68–75, 1998.

20 M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. 13th Annu. Sympos. Comput. Geom.*, pages 212–220, 1997.