Monotonic Tree

Yuqing Song and Aidong Zhang

Department of Computer Science & Engineering
State University of New York at Buffalo
Buffalo, NY 14260, USA

Abstract. Contour trees have been used in geographic information systems (GIS) and medical imaging to display scalar data. Contours are only defined for continuous functions. For an image represented by discrete data, a continuous function is first defined as an interpolation of the data. Then the contour tree is defined on this continuous function. In this paper, we introduce a new concept termed monotonic line, which is directly defined on discrete data. All monotonic lines in an image form a tree, called monotonic tree. As compared with contour trees, monotonic trees avoid the step of interpolation, thus can be computed more efficiently. Monotonic tree can also be used as a hierarchical representation of image structures in computer imagery.

1 Introduction

The concepts of contour trees have been developed by Morse [1], Roubal and Peucker [2], and recently by van Kreveld et al. [3]. In geographic information systems (GIS), contour trees are used to display scalar data defined over the plane, or the three-dimensional space. For example, the elevation in the landscape can be modeled by scalar data over the plane, where a contour (also called an isoline) is a line where the elevation function assumes the same value. Contour trees are also used in medical imaging to show the scanned data.

Contours are only defined for continuous functions. For an image represented by discrete data, a continuous function is first defined as an interpolation of the data. Then the contour tree is defined on this continuous function. In this paper, we introduce a new concept termed monotonic line, which is directly defined on discrete data.

We observe that for any 2D Morse function, a curve is a normal contour with value v iff it's a boundary of the set $\{x \in \mathbb{R}^2 | f(x) > v\}$. This is not true for non-Morse functions. However, the equivalent condition is more general, and can be used to define contours for discontinuous or discrete functions. Specifically, an *outward-falling/climbing monotonic line* of an gray image is a boundary where the image assumes higher/lower values in the pixels adjacent to the boundary from inside than those from outside (see Figure 1(a)). The two kinds of monotonic lines correspond to positive and negative contour lines in [1], respectively. To make the boundary of the image domain a monotonic line, we extend the

A. Braquelaire, J.-O. Lachaud, and A. Vialard (Eds.): DGCI 2002, LNCS 2301, pp. 114–123, 2002.

input image to the whole digital plane such that the extended function assumes $-\infty$ out of the domain.

It can be proved that monotonic lines don't cross each other, i.e., if $l_1 = \partial X$, $l_2 = \partial Y$ are two monotonic lines, where X, Y are two simply connected regions, then $X \subseteq Y$, $Y \subseteq X$ or $X \bigcap Y = \emptyset$. Based on this property, we can define a parent-child relation: monotonic line l_1 is the parent of monotonic line l_2, if l_2 is directly enclosed by l_1. Under this parent-child relation, all monotonic lines in an image form a rooted tree, called *monotonic tree*. See Figure 1(b)(c). A monotonic tree can be reduced. A maximal sequence of uniquely enclosing monotonic lines is called a *monotonic slope*. All monotonic slopes in an image form the *topological monotonic tree* (TMT). See Figure 1(d).

Algorithms for computing traditional contour trees can be easily modified to compute monotonic trees or topological monotonic trees. Because the monotonic line is directly defined on pixels, the interpolation step is avoided. Thus the monotonic trees and the topological monotonic trees can be computed more efficiently.

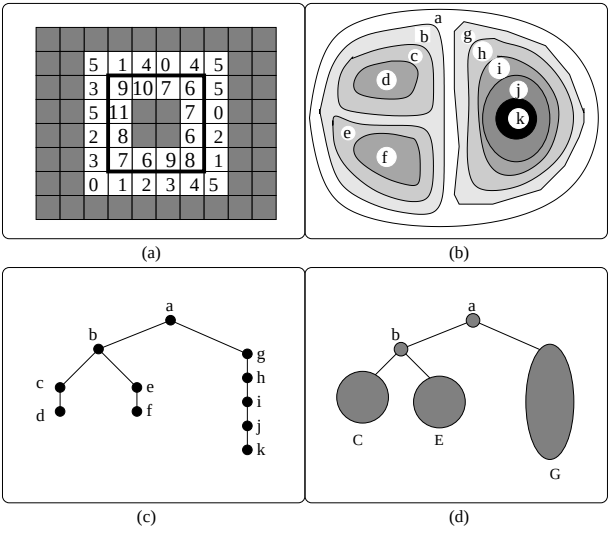

Fig. 1. (a) An outward-falling monotonic line (the solid line in the figure), (b) a set of monotonic lines, (c) the monotonic tree, (d) the topological monotonic tree (TMT).

Monotonic tree can be used as a hierarchical representation of image structures in computer imagery. As compared with other models such as wavelet, the monotonic tree model has following advantages.

(1) The monotonic tree retrieves and represents the structures of an image at all scales. In addition, these structures are organized hierarchically as a tree,

which gives us a better way to analyze the relationship between different levels.

(2) The monotonic tree retrieves the structures of an image directly and maintains their original shapes.

An example is shown in Figure 2. In this example, the tree region and water wave region are characterized by the shapes and permutation of the TMT elements in these two regions, which makes it possible to recognize the trees and water waves by classifying and clustering the TMT elements. Based on our monotonic tree model, we made an online demo for scenery analysis, which is available at "http://monet.cse.buffalo.edu:8888".

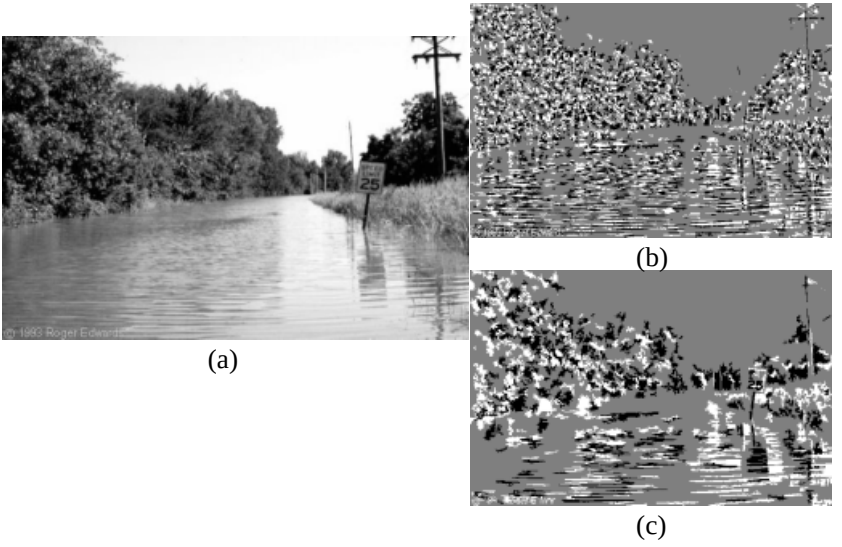

Fig. 2. (a) Original image, (b) the elements of the TMT at a smaller scale, and (c) the elements of the TMT at a larger scale. In (b) and (c), the TMT elements are shown by black and white regions in a gray background.

2 Preliminary

In our theoretical discussion, we choose hexagonal grid for the digital plane. We use notations in [6] with modifications. The digital plane is a pair $(\mathbf{V}_h, \Pi_h)$, where $\mathbf{V}_h$ is the set of pixels:

$$\mathbf{V}_h = \{h_1(1,0) + h_2(-0.5, \sqrt{0.75}) | h_1, h_2 \in \mathbb{Z}\}, \tag{1}$$

and Π_h is the edge-adjacency shown in Figure 3(a). Π_h is a symmetric binary relation on $\mathbf{V}_h$ such that $\mathbf{V}_h$ is Π_h-connected. Formally,

$$\Pi_h = \{(p,q) | p, q \in \mathbf{V}_h, and \ \| p - q \| = 1\}. \tag{2}$$

Each pixel P has 6 edges, named as $e_i(P)$ for $i = 0, 1, ..., 5$. See Figure 3(b). Each $e_i(P)$ is an element of Π_h. We define function $next_P$ on $\{e_i(P)\}_{i=0}^{5}$ by

$$next_P(e_i(P)) = e_{(i+1)\%6}(P). \tag{3}$$

Function $next_P$ defines the counter-clockwise direction in the border of P.

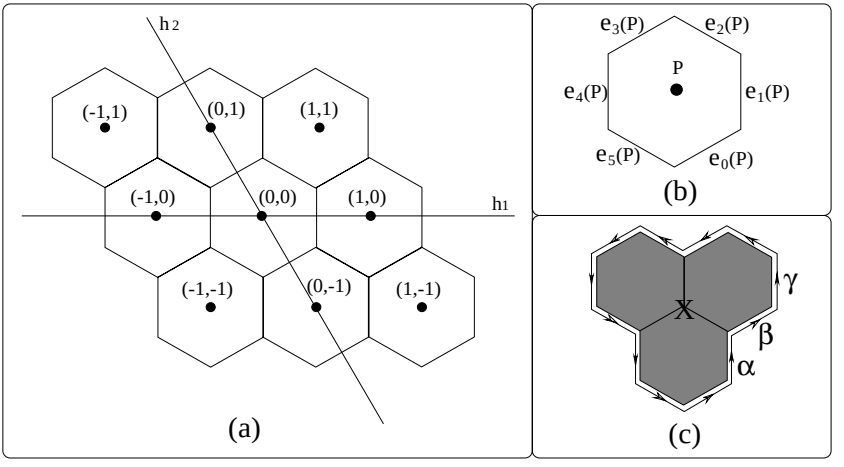

Fig. 3. (a) Hexagonal grid, (b) six edges of pixel P, and (c) the connected boundary of a digital region.

For any subset X of $\mathbf{V}_h$, its *border* is defined as

$$\partial X = \{(p, q) \in \Pi_h | p \in X, and\ q \notin X\}. \tag{4}$$

We define function $next_X$ on ∂X such that for $e = (p, q) \in \partial X$ with $p \in X$ and $q \notin X$,

$$next_X(e) = \begin{cases} next_p(e) & \text{if } next_p(e) \in \partial X; \\ next_q^{-1}(e) & \text{otherwise.} \end{cases} \tag{5}$$

Function $next_X$ defines the counter-clockwise direction in the border of X. For example, in Figure 3(c), $next_X(\alpha) = \beta$ and $next_X(\beta) = \gamma$.

The following definition defines *connected regions*.

Definition 1. *A region X is called a* ***connected region*** *if $(X, \Pi_h|_X)$ is a connected graph, i.e., for any $p, q \in X$, there exists a Π_h-connected path in X connecting p and q, where a* ***Π_h-connected path*** *is a sequence of pixels $\{p_i\}_{i=1}^{n}$ such that $(p_i, p_{i+1}) \in \Pi_h$ for $i = 1, 2, ..., n-1$.*

For a connected region with no holes, its border is a *connected boundary*, which is an *boundless list* of pixel edges. See Figure 3(c). Formally, we give two definitions.

Definition 2. *A **boundless list** is a pair $(X, next)$ such that X is a set and*
(1) $next$ is a bijective function from X to X; and
(2) $\forall x, y \in X$, there exists an integer $n \geq 0$ such that either $x = next^n(y)$, or $y = next^n(x)$.
*A **circular list** is a boundless list $(X, next)$ such that X is a finite set.*

It's easy to see that for a circular list $(X, next)$ and any $x, y \in X$, there exists an integer $n \geq 0$ such that $x = next^n(y)$.

Definition 3. *For a region $X \subseteq \boldsymbol{V}_h$, a **connected boundary** of X is a subset S of its border ∂X such that $next_X(S) = S$ and $(S, next_X|_S)$ is a boundless list.*

About the boundaries of regions, we have following lemma.

Lemma 1. *For any $X \subseteq \boldsymbol{V}_h$,*
(1) $\partial X = \partial(\boldsymbol{V}_h - X)$;
(2) $next_X = (next_{\boldsymbol{V}_h - X})^{-1}$;
(3) the border of X can be decomposed into a set of connected boundaries; and
(4) if X is bounded (i.e., it's finite), then its connected boundaries are circular lists.

This lemma is obvious. Next we give the definition for simple connection.

Definition 4. *A region $X \subseteq \boldsymbol{V}_h$ is called **simply connected** if both X and $\boldsymbol{V}_h - X$ are connected.*

The simple connection we defined here is a little bit different from the traditional definition on Euclidean plane. However, if we add an infinite pixel to the hexagonal grid and make it a digital sphere, then our definition fits the definition of simple connection on Euclidean sphere.

About simple connection, we have following lemma.

Lemma 2. *For a bounded region $X \subset \boldsymbol{V}_h$, X is simply connected iff $(\partial X, next_X)$ is a circular list. That is to say, for a bounded region X, $(\partial X, next_X)$ is a circular list iff both X and $\boldsymbol{V}_h - X$ are connected.*

This lemma is an equivalent of Jordan's curve theorem on the hexagonal grid. Due to the limited space, in this paper, we give our lemmas and theorems with the proof omitted.

Next, we model gray images on the hexagonal grid.

Definition 5. *A **gray image** is a pair (f, Ω) such that $\Omega \subset \boldsymbol{V}_h$ is a bounded and simply connected region, and f is a real valued function defined on Ω.*

For any gray image (f, Ω), we extend the function f to the whole plane by

$$f_E(p) = \begin{cases} f(p) \text{ if } p \in \Omega; \\ -\infty \text{ otherwise.} \end{cases} \tag{6}$$

f_E is called the ***extended function*** of f. In fact, instead of choosing $-\infty$, we can choose any value which is less than all values assumed by f, or greater than all values assumed by f.

3 Monotonic Line

In this and following sections, let $I = (f, \Omega)$ be a fixed gray image, and f_E be the extended function of f. For a region $X \subseteq \mathbf{V}_h$, we denote the immediate interior [6] of ∂X as $IntBorderPixelSet(X)$, and the immediate exterior as $ExtBorderPixelSet(X)$, i.e.,

$$IntBorderPixelSet(X) = \{x \in X | (x, y) \in \Pi_h \text{ for some } y \notin X\}; \quad (7)$$

$$ExtBorderPixelSet(X) = \{y \in \mathbf{V}_h - X | (x, y) \in \Pi_h \text{ for some } x \in X\}. \quad (8)$$

Now we can define ***monotonic line***.

Definition 6. *A **monotonic line** of I is a boundary ∂X such that $X \subseteq \Omega$ is simply connected and not empty, and there exists some $v \in \mathbb{R}$ with the property that either of the following is true:*
(1) $\forall x \in IntBorderPixelSet(X), f_E(x) > v$, *and* $\forall y \in ExtBorderPixelSet(X), f_E(y) < v$;
(2) $\forall x \in IntBorderPixelSet(X), f_E(x) < v$, *and* $\forall y \in ExtBorderPixelSet(X), f_E(y) > v$.
*If (1) is true, ∂X is called **outward falling**; if (2) is true, ∂X is called **outward climbing**. We denote the set of all monotonic lines of I as $MonotonicLineSet(I)$.*

We can prove that monotonic lines don't cross each other, i.e., for any $\partial X, \partial Y$ in $MonotonicLineSet(I)$, one of the following is true: $X \subseteq Y$, $Y \subseteq X$ or $X \bigcap Y = \emptyset$.

Theorem 1. *$\forall \partial X, \partial Y \in MonotonicLineSet(I)$, one of following is true:* $X \subseteq Y$, $Y \subseteq X$ *or* $X \bigcap Y = \emptyset$.

The basic idea to prove this theorem is simple. Suppose we have two monotonic lines ∂X and ∂Y crossing each other. Let a, b, c, d be four pixels around a crossing point. See Figure 4. By the definition of monotonic line, we discuss on four cases:
(1) both ∂X and ∂Y are outward falling;
(2) ∂X is outward falling, ∂Y is outward climbing;
(3) ∂X is outward climbing, ∂Y is outward falling; and
(4) Both ∂X and ∂Y are outward climbing.
For each case, we can get a contradiction. For example, in case (1), there exists v_1 such that

$$f_E(a) > v_1, f_E(c) > v_1, \text{ and } f_E(b) < v_1, f_E(d) < v_1;$$

and there exists v_2 such that

$$f_E(c) > v_2, f_E(d) > v_2, \text{ and } f_E(a) < v_2, f_E(b) < v_2.$$

Then we get both $f_E(a) > v_1 > f_E(d)$ and $f_E(d) > v_2 > f_E(a)$, which is a contradiction.

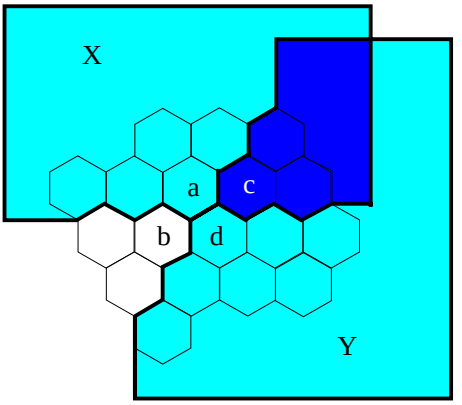

Fig. 4. ∂X crosses ∂Y.

4 Monotonic Tree and Topological Monotonic Tree

We first define some relations on $MonotonicLineSet(I)$.

Definition 7. *For any monotonic lines* $\partial X, \partial Y \in MonotonicLineSet(I)$,

- ∂X ***encloses*** ∂Y, *denoted as* $Enclose(\partial X, \partial Y)$, *if* $X \supset Y$;
- ∂X ***directly encloses*** ∂Y, *denoted as* $DirectEnclose(\partial X, \partial Y)$, *if* $Enclose(\partial X, \partial Y)$ *and there is no* $\partial Z \in MonotonicLineSet(I)$ *such that* $X \supset Z \supset Y$;
- ∂X ***uniquely directly encloses*** ∂Y, *denoted as* $UniqueDirectEnclose(\partial X, \partial Y)$, *if* $DirectEnclose(\partial X, \partial Y)$ *and* $\forall \partial Z \in MonotonicLineSet(I)$, $DirectEnclose(\partial X, \partial Z) \Rightarrow \partial Y = \partial Z$.

The relation $DirectEnclose$ is a parent-child relation on the set of monotonic lines in gray image I. Based on Theorem 1, we can easily prove that:

Theorem 2. $(MonotonicLineSet(I), DirectEnclose)$ *is a rooted tree, and* $\partial\Omega$ *is its root.*

The tree $(MonotonicLineSet(I), DirectEnclose)$ is called the ***monotonic tree*** of image I, and denoted as $MonotonicTree(I)$.

Next we can define monotonic slope and topological monotonic tree.

Definition 8. *A* ***monotonic slope*** s *is a maximal sequence of monotonic lines* $s = \{l_i\}_{i=1}^{n}$ *with* $n \geq 1$ *such that*
(1) $\forall i = 1, 2, ..., n-1$, $UniqueDirectEnclose(l_i, l_{i+1})$; *and*
(2) *either all* l_i *are outward-falling, or all* l_i *are outward-climbing.*

s is called ***outward-falling/outward-climbing*** *if all l_i are outward-falling/outward-climbing. The first monotonic line l_1 is called the* ***enclosing line*** *of the slope s. The set of all monotonic slopes is denoted as $MonotonicSlopeSet(I)$.*

We can also define some relations on $MonotonicSlopeSet(I)$.

Definition 9. *For any $s_a, s_b \in MonotonicSlopeSet(I)$, we define*
(1) $Enclose(s_a, s_b)$ *if* $\exists l_a \in s_a, \exists l_b \in s_b, Enclose(l_a, l_b)$;
(2) $DirectEnclose(s_a, s_b)$ *if* $\exists l_a \in s_a, \exists l_b \in s_b, DirectEnclose(l_a, l_b)$.

The relation $DirectEnclose$ is a parent-child relation on the set of all monotonic slopes.

Theorem 3. *$(MonotonicSlopeSet(I), DirectEnclose)$ is a rooted tree, and the monotonic slope which contains $\partial\Omega$ is the root.*

The tree $(MonotonicSlopeSet(I), DirectEnclose)$ is called the ***topological monotonic tree*** of I, and denoted as $TopologicalMonotonicTree(I)$.

5 Properties of Monotonic Tree

In this section, we introduce some theorems about monotonic tree.

Theorem 4. ***Inside Intersecting Theorem***
For any $\partial X, \partial Y \in MonotonicLineSet(I)$, if

$$IntBorderPixelSet(X) \bigcap IntBorderPixelSet(Y) \neq \emptyset,$$

then

$$\partial X \text{ is outward falling} \Leftrightarrow \partial Y \text{ is outward falling.}$$

This theorem is equivalent to the statement that for a sequence of monotonic lines which intersect from inside, either they are all outward falling, or they are all outward climbing. See Figure 5(a).

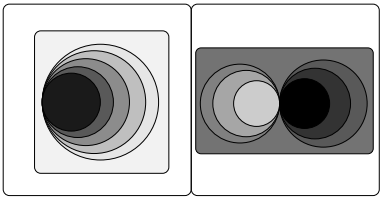

Fig. 5. (a) An inside-intersecting sequence, and (b) two inside-intersecting sequences which intersect from outside.

Theorem 5. ***Outside Intersecting Theorem***
For any $\partial X, \partial Y \in MonotonicLineSet(I)$, *if*

$$IntBorderPixelSet(X) \bigcap ExtBorderPixelSet(Y) \neq \emptyset,$$

then

$$\partial X \text{ is outward falling} \Leftrightarrow \partial Y \text{ is outward climbing.}$$

This theorem is equivalent to the statement that for two inside-intersecting sequences, if they intersect from outside, then the monotonic lines in one sequence are all outward falling, and the monotonic lines in the other sequence are all outward climbing. See Figure 5(b).

Theorem 6. ***Separation Theorem***
Let x, y *be two pixels in* Ω.
(1) If there is a Π_h*-connected path* $P = \{w_1 = x, w_2, ..., w_n = y\}$ *in* Ω *such that* f *is constant along this path, i.e.,* $f(w_1) = f(w_2) = ... = f(w_n)$, *then* $\forall\ \partial X \in MonotonicLineSet(I), x \in X \Leftrightarrow y \in X$.
(2) If $f(x) \neq f(y)$, *then there is some* $\partial X \in MonotonicLineSet(I)$ *such that* $x \in X \Leftrightarrow y \notin X$.

This theorem states that for any two pixels in the image domain, (1) if they are connected by a path where the function assumes constant value, then no monotonic line separates them; and (2) if the function assumes different values at the two pixels, then there is some monotonic line separating them.

Theorem 7. ***Private Region Theorem***
For any $\partial X \in MonotonicLineSet(I)$, *let* $\{\partial Y_i\}_{i=1}^n$ $(n \geq 0)$ *be the set of children of* ∂X *in the monotonic tree. Then*
(1) $(X - \bigcup_{i=1}^n Y_i) \bigcap IntBorderPixelSet(X)$ *is not empty; and*
(2) $\forall x, y \in (X - \bigcup_{i=1}^n Y_i,)\ f(x) = f(y)$.

This theorem states that each monotonic line has a nonempty private region and that the function is constant over this region. Based on this theorem, we give following definition.

Definition 10. *For any* $\partial X \in MonotonicLineSet(I)$, *let* $\{\partial Y_i\}_{i=1}^n$ $(n \geq 0)$ *be the set of children of* ∂X *in the monotonic tree. We define the* ***private region*** *of* ∂X *to be:*

$$PRegion_I(\partial X) = X - \bigcup_{i=1}^n Y_i.$$

We define the ***assumed value*** *of* ∂X *to be the constant value assumed by* f *over* $PRegion_I(\partial X)$. *The assumed value of* ∂X *is denoted as* $Value_I(\partial X)$.

Theorem 8. ***Value Jumping Theorem***
For any $\partial P, \partial C \in MonotonicLineSet(I)$, *if* ∂P *is the parent of* ∂C, *then*
(1) $Value_I(\partial P) \neq Value_I(\partial C)$; *and*
(2) ∂C *is outward falling* $\Leftrightarrow Value_I(\partial P) < Value_I(\partial C)$.

The property (1) in this theorem says that there is a value jump (up or down) from a child to its parent. Value jumping is a natural property for contour trees, but it's not straight forward for monotonic trees.

6 Conclusion and Discussion

Contour trees are only defined on continuous functions. When applying the contour tree model to computer imagery, we have to make an interpolation of the discrete data, which make the computation not efficient. We solved this problem by introducing our monotonic tree model.

One main difficulty of computer imagery comes from the discrete and noise nature of images. While we have plenty of powerful theories to analyze the structures of smooth functions, we need more for discrete functions. The monotonic tree can be used as a theoretical tool for discrete functions.

The capacity of the monotonic tree model can be extended. The topological structure of the monotonic lines in an image is captured by the topological monotonic tree. We may further define *differential slope* and *differential monotonic tree* to capture the differential information. A differential slope may be defined as a sequence of monotonic lines where the gradient is smooth.

References

1. S. Morse. Concepts of use in computer map processing. Communications of the ACM, 12(3):147-152, March 1969.
2. J. Roubal and T.K. Peucker. Automated contour labeling and the contour tree. In Proc. AUTO-CARTO 7, pages 472-481, 1985.
3. M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for iso-surface traversal. In Proc. 13th Ann. Sympos. Comput. Geom., pages 212-220, 1997.
4. Mark de Berg and Marc J. van Kreveld. Trekking in the alps without freezing or getting tired. In European Symposium on Algorithms, pages 121-132, 1993.
5. M. van Kreveld. Efficient methods for isoline extraction from a tin. International Journal of GIS, 10:523-540, 1996.
6. Gabor T. Herman. Geometry of Digital Spaces. Birkhauser Boston, 1998.