Monotonic Tree of Images and Its Application in Image Processing

Yuqing Song, Aidong Zhang
Department of Computer Science & Engineering
State University of New York at Buffalo
Buffalo, NY 14260

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 be reduced. The reduced monotonic tree can also be used as a hierarchical representation of image structures in image processing. In particular, we demonstrate its application on image smoothing and texture retrieval. Experiments show that our smoothing scheme is successful in both noise reduction and texture retrieval.

1 Introduction

1.1 Contour tree

The concepts of contour trees have been developed by Morse [11], Roubal and Poiker [12], and recently by van Kreveld et al. [17]. 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.

The example shown in Figure 1 is taken from [17]. In this example, the input data set is modeled by 2D triangle mesh with linear interpolation. The contours of all critical vertices in the mesh subdivide the 2D domain into regions. Every region between contours is bounded by two contours. To construct the contour tree, each contour in the subdivision corresponds to a node in the graph, and two nodes are connected (from max to min) if there is a region bounded by their corresponding contours. This graph is a tree, which is easy to show [3, 16], and it is called the contour tree.

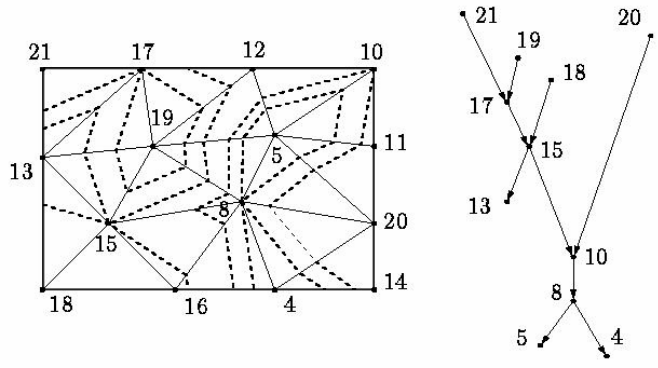

Figure 1: An example of 2D contour tree.

In general, for any continuous function $f : \mathbb{R}^d \to \mathbb{R}$, a *contour* is defined to be a connected component of the level set $\{x \in \mathbb{R}^d | f(x) = v\}$ for some $v \in \mathbb{R}$. For general functions, contours may be d-dimensional, $(d-1)$-dimensional, or lower dimensional. For *Morse functions* defined on $\mathbb{R}^d$, contours are either $(d-1)$-dimensional hypersurfaces (if they are normal contours), or isolated points (if they are degenerate contours). A function whose critical points are isolated is called a *Morse function*.

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.

1.2 Monotonic tree

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 of discontinuous functions or

discrete functions. For an image represented by a discrete function $f : \{0, 1, ..., M-1\} \times \{0, 1, ..., N-1\} \rightarrow \mathbb{R}$ and some value v, a boundary l of the set $\{x \in \mathbb{Z}^2 | f(x) > v\}$ is always closed and has such a property that the function is monotonic from one side of l to the other side, i.e., either of following is true:

(1) the function f assumes values higher than v at pixels adjacent to the interior side of l, and values no more than v at pixels adjacent to the exterior side of l. In this case, l is called an *outward-falling monotonic contour* or monotonic line;

(2) the function assumes values no more than v at pixels adjacent to the interior side of l, and values higher than v at pixels adjacent to the exterior side of l. In this case, l is called an *outward-climbing monotonic contour* or monotonic line.

Figure 2(a) gives an example of outward-falling monotonic line. The two kinds of monotonic lines correspond to positive and negative contour lines in [11]. However, our definition works for discontinuous functions and discrete functions. To make the boundary of the domain rectangle a monotonic line, we extend the input function to the whole plane $\mathbb{Z}^2$ such that the extended function assumes $-\infty$ out of the domain rectangle. 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 relationship: 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 relationship, all monotonic lines in an image form a rooted tree, called *monotonic tree*. For example, all monotonic lines in Figure 2(b) form a monotonic tree shown in Figure 2(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 a *reduced monotonic tree*. See Figure 2(d).

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

1.3 Multiscale, Noise and Texture in Image Processing

Multiscale (or multi-resolution) is one main methodology in image processing. Images have different features at different scales. As Lindeberg said in [9], "*... every operation on image data must be carried out on a window, whose size can range from a single point to the whole image. The type of information we can get from such an operation is largely determined by the relationships between structures in the image and the size of the window. Hence, without prior knowledge about what we are looking for, there is no reason to favor any particular scale. We should therefore try them all and operate at all window sizes.*" Since the type of information we can get from a window-based operation is also largely determined by the relationship between the shapes of the structures in the image and the shape of the window, we should also try all window shapes, which will make the window-based operations not efficient. We may wonder: can we get the structures of an image directly? The reduced monotonic tree of an image is one approach to retrieving and representing the structures of the image hierarchically. Each branch (i.e., subtree) of the reduced monotonic tree represents a structure, where the subbranches are substructures. As compared with other multiscale techniques such as wavelet, the reduced monotonic tree retrieves the image structures directly and maintains their original shapes. For the wavelet-based techniques the features are retrieved through operations such as convolving with the mother wavelets at different scales, thus the shapes of corresponding structures are transformed and sometimes lost in the retrieved features.

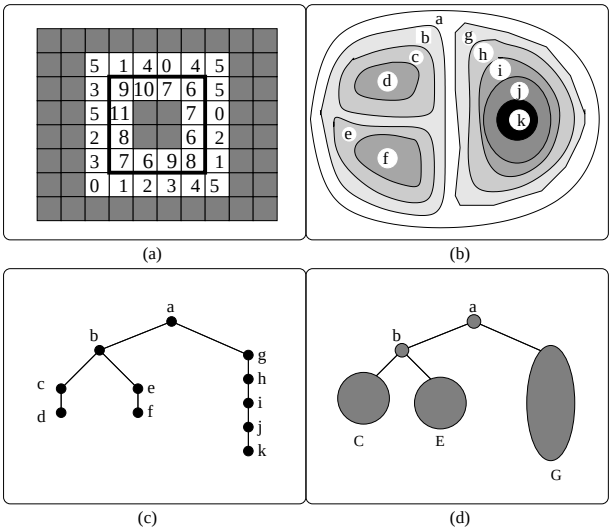

Figure 2: (a) An outward-falling monotonic line, (b) a set of monotonic lines, (c) the monotonic tree, (d) the reduced monotonic tree.

In this paper, we address the problems of image smoothing and texture retrieval to demonstrate the application of the reduced monotonic tree. Image smoothing, or noise removal, is one of the most important design goals of image enhancement. Existing smoothing techniques include linear filtering such as Gaussian smoothing operation, nonlin-

ear filtering such as Kuwahara filter [4], statistical methods [8, 1], wavelet [6, 18], and PDE based methods [10, 7]. Texture is defined by Tamura et al. [13] as *"what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule."* Texture analysis schemes include geometrical methods [15], random field models [2], and wavelet models [5].

Noise in images corresponds to high frequency parts. Usually so does texture. In many cases, existing smoothing techniques remove or reduce both noise and texture. While more effort has been made on image smoothing and texture analysis separately, less has been done to tell and handle the difference between texture and noise, especially when texture and noise are mixed together in one image. This situation makes texture retrieval difficult. On one hand, if we don't apply smoothing algorithms first, the existence of noise in an image affects badly the performance of texture analysis schemes on the image; on the other hand, if we apply smoothing algorithms first, it's a risk that the texture structure is also removed or damaged during the process of smoothing.

The reduced monotonic tree provides a frame in which we can uniformly handle image smoothing and texture retrieval. In a reduced monotonic tree, each branch represents a structure, whose scale is determined by the area covered by this branch. The fine scale detail, which represents the high frequency part of the image, consists of all branches whose areas are no more than a scale threshold. Given a scale threshold, we smooth the image by cutting all branches of the reduced monotonic tree whose areas are no more than the threshold.

What if the branches being cut are regular in shape and permutation, i.e., they don't correspond to the noise part of the image? In this case, we get texture. Essentially, the detail in small scales is either texture or noise. The main difference between texture and noise is that the texture of an object is an intrinsic part of the object, consisting of repetitive patterns which are regular in placement; the noise is a result of outer forces, and usually is irregular. The relationship among smoothing, texture and multiscale is illustrated in Figure 3.

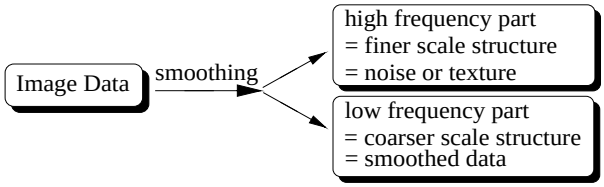

Figure 3: Relationship among smoothing, texture and multiscale.

Our texture retrieval scheme can be classified as a geometric method. The class of geometric texture analysis methods is characterized by their definition of texture as being composed of texture elements [14]. The texture elements are called *textons*. The main difficulty of geometric methods is at the step of texton retrieval. Once the textons are identified in the image, other techniques such as point pattern recognition, shape analysis and statistics can be used to analyze the features in the shape patterns and placement of the textons. Based on the reduced monotonic tree, for a given scale, the structure at this scale is represented by a set of branches (which cover disjoint regions in the image). If the branches at this scale are regular in shape and permutation, each of them is a texton. Otherwise, they represent the noise at this scale. Thus the reduced monotonic tree provides a facility helping us understand the difference between texture and noise.

2 Representation of Boundaries of Digital Regions as Circular Lists

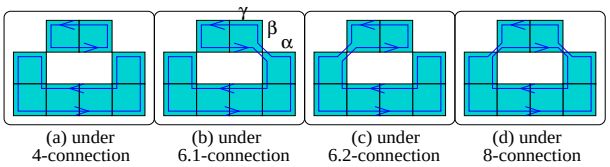

Figure 4: Boundaries of a digital region under different connections.

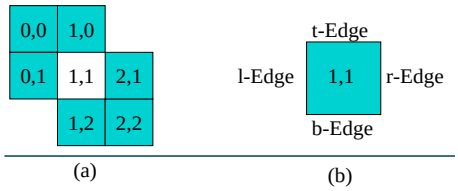

Figure 5: (a) 6.1-neighbors of pixel (1,1), (b) 4 pixel edges of (1,1).

The boundary of a digital region depends on what kind of pixel connection is assumed in digital plane $\mathbb{Z}^2$. See Figure 4. There are four kinds of neighbors in the digital plane: 4-neighbors, two 6-neighbors, and 8-neighbors. In this paper, we use one kind of 6-neighbors, and denote it as 6.1-neighbors: for each pixel (i, j), its **6.1-neighbors** consist of $(i, j-1), (i, j+1), (i-1, j), (i+1, j), (i-1, j-1)$, and $(i+1, j+1)$. See Figure 5(a). Two pixels p, q are called 6.1-**adjacent**, if $p = q$ or p is a 6.1-neighbor of q. A sequence of pixels $\{p_i\}_{i=1}^{n}$ is called a 6.1-**path**, if for all $1 \leq i < n$, p_i, p_{i+1} are 6.1-adjacent. For any digital region S and any $p, q \in S$, p, q are called 6.1-**connected**

in S if there is a 6.1-path in S connecting p, q. A digital region S is called 6.1-**connected** if any two pixels in S are 6.1-connected in S.

In digital geometry, the boundary of a digital region is represented by the boundary pixels. We choose a different way and represent a boundary by a circular list of pixel edges.

Definition 2.1 *Each pixel $(i, j) \in \mathbb{Z}^2$ corresponds to a square in $\mathbb{R}^2$:*

$Square(i, j) = [i - \frac{1}{2}, i + \frac{1}{2}] \times [j - \frac{1}{2}, j + \frac{1}{2}]$.

*For pixel (i,j), we define its **pixel edges** as the four edges of $Square(i, j)$. See Figure 5(b). More specifically, we define its top pixel edge to be*

t-Edge (i,j)$=\{(x, j - \frac{1}{2}) | i - \frac{1}{2} \leq x \leq i + \frac{1}{2}\}$,

its bottom pixel edge to be

b-Edge $(i, j) = \{(x, j + \frac{1}{2}) | i - \frac{1}{2} \leq x \leq i + \frac{1}{2}\}$,

its left pixel edge to be

l-Edge $(i, j) = \{(i - \frac{1}{2}, y) | j - \frac{1}{2} \leq y \leq j + \frac{1}{2}\}$, *and its right pixel edge to be*

r-Edge $(i, j) = \{(i + \frac{1}{2}, y) | j - \frac{1}{2} \leq y \leq j + \frac{1}{2}\}$.

The four edges are also denoted as (i,j,top), (i,j,bottom), (i,j,left) and (i,j,right), respectively. The set of the four edges are denoted as EdgeSet(i,j). For any pixel edge e of pixel p, it's also an edge of another pixel q, and we define the pixel set of e as:

$PixelSet(e) = \{p, q\}$.

Each pixel edge have two representations. For example, the pixel edge (i, j, top) can also be represented as $(i, j - 1, bottom)$.

Definition 2.2 *A **circular list** is a pair $< X, next >$ such that X is a finite set and*

(1) next is a bijective function from X to X; and

(2) $\forall x, y \in X$, there exists integer $n \geq 0$ such that $x = next^n(y)$, i.e., $x = \underbrace{next(next(...}_{n}(y)))$.

Definition 2.3 *For any set $X \subseteq \mathbb{Z}^2$, we define ∂X to be the set of all pixel edges which are in the border, i.e., $\partial X = \{e = (i, j, ename) | (i, j) \in X; ename = top, bottom, right, or\ left; and\ PixelSet(e) \not\subseteq X\}$.*

We define the "next" function on ∂X as:

$$if\ (i,j) \in X\ and\ (i,j,top) \in \partial X,\ Next^X_{6.1}(i,j,top) = \begin{cases} (i,j,left) & if\ (i-1,j-1) \notin X, (i-1,j) \notin X; \\ (i-1,j,top) & if\ (i-1,j-1) \notin X, (i-1,j) \in X; \\ (i-1,j-1,right) & if\ (i-1,j-1) \in X; \end{cases}$$

$$if\ (i,j) \in X\ and\ (i,j,bottom) \in \partial X,\ Next^X_{6.1}(i,j,bottom) = \begin{cases} (i,j,right) & if\ (i+1,j+1) \notin X, (i+1,j) \notin X; \\ (i+1,j,bottom) & if\ (i+1,j+1) \notin X, (i+1,j) \in X; \\ (i+1,j+1,left) & if\ (i+1,j+1) \in X; \end{cases}$$

$$if\ (i,j) \in X\ and\ (i,j,left) \in \partial X,\ Next^X_{6.1}(i,j,left) = \begin{cases} (i,j,bottom) & if\ (i,j+1) \notin X; \\ (i,j+1,left) & if\ (i,j+1) \in X, (i-1,j+1) \notin X; \\ (i-1,j+1,top) & if\ (i,j+1) \in X, (i-1,j+1) \in X; \end{cases}$$

$$if\ (i,j) \in X\ and\ (i,j,right) \in \partial X,\ Next^X_{6.1}(i,j,right) = \begin{cases} (i,j,top) & if\ (i,j-1) \notin X; \\ (i,j-1,right) & if\ (i,j-1) \in X, (i+1,j-1) \notin X; \\ (i+1,j-1,bottom) & if\ (i,j-1) \in X, (i+1,j-1) \in X. \end{cases}$$

In Figure 4(b), $Next^X_{6.1}(\alpha) = \beta$ and $Next^X_{6.1}(\beta) = \gamma$. It's easy to see that $\forall X \subseteq \mathbb{Z}^2, Next^X_{6.1}$ is well defined. In addition, $\forall X \subseteq \mathbb{Z}^2$ we have

(1) $\partial X = \partial(\mathbb{Z}^2 - X)$;

(2) $Next^{\mathbb{Z}^2 - X}_{6.1} = (Next^X_{6.1})^{-1}$; and

(3) if X is finite, then ∂X can be decomposed into a set of disjoint subsets $\{b_i\}_{i=1}^n$ with $n \geq 1$ such that each $< b_i, Next^X_{6.1}|_{b_i} >$ is a circular list. Each $< b_i, Next^X_{6.1}|_{b_i} >$ is called a **boundary** of X.

The property (2) above ensures that for a discrete function f defined on $\mathbb{Z}^2$, and some $v \in \mathbb{R}$, a boundary of $\{x \in \mathbb{Z}^2 | f(x) > v\}$ is also a boundary of $\{x \in \mathbb{Z}^2 | f(x) \leq v\}$. This property is not held by 4- or 8-neighbor connections, which is the reason why we don't choose 4- or 8-neighbor connections to define monotonic lines.

For finite X, $Next^X_{6.1}$ defines a direction on ∂X: on the outer boundary, the direction is counter clockwise; and on the inner boundaries, the direction is clockwise.

Definition 2.4 *A finite set $X \subseteq \mathbb{Z}^2$ is **simply 6.1-connected** if both X and $\mathbb{Z}^2 - X$ are 6.1-connected.*

Lemma 2.1 *For a finite set $X \subset \mathbb{Z}^2$, X is simply 6.1-connected iff $< \partial X, Next^X_{6.1} >$ is a circular list. That is to say, for a finite digital set X, $< \partial X, Next^X_{6.1} >$ is a circular list iff both X and $\mathbb{Z}^2 - X$ are 6.1-connected.*

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

3 Definition and Property of Monotonic Lines

Any gray image can be represented by a function $f : \{0, 1, ..., M - 1\} \times \{0, 1, ..., N - 1\} \to \mathbb{R}$. In this paper, it's assumed that an image is always represented this way. Let's extend the function to the whole plane $\mathbb{Z}^2$.

Definition 3.1 *Let $v_o = -\infty$. [1] For any function $f : \{0, 1, ..., M-1\} \times \{0, 1, ..., N-1\} \to \mathbb{R}$, we define the **extended function** of f to be a function $F : \mathbb{Z}^2 \longrightarrow \mathbb{R} \bigcup \{v_o\}$ such that*

$$F(i,j) = \begin{cases} f(i,j) & if\ 0 \leq i < M, 0 \leq j < N; \\ v_o & otherwise. \end{cases}$$

Denote F as $Extension(f)$.

Definition 3.2 *For any digital region $X \subseteq \mathbb{Z}^2$, we define:*

$OutBorderPixelSet_{6.1}(X) = \{x \in \mathbb{Z}^2 -$

[1] In fact, we can choose any v_o which is greater than all values assumed by f, or smaller than all values assumed by f.

$X|x$ *is* 6.1-*adjacent with some* $y \in X\}$;

$InBorderPixelSet_{6.1}(X) = \{x \in X|x$ *is* 6.1-*adjacent with some* $y \in \mathbb{Z}^2 - X\}$.

Definition 3.3 *For any image represented by function f, let Ω be its domain, and F be its extended function. A **monotonic line** of f is a boundary ∂X such that $X \subseteq \Omega$ is simply 6.1-connected and not empty, and there exists $v \in \mathbb{R}$ with the property that either of the following is true:*

(1) $\forall x \in InBorderPixelSet_{6.1}(X), F(x) > v$, $\forall y \in OutBorderPixelSet_{6.1}(X), F(y) < v$;

(2) $\forall x \in InBorderPixelSet_{6.1}(X), F(x) < v$, $\forall y \in OutBorderPixelSet_{6.1}(X), F(y) > v$.

*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 f as $MonotonicLineSet(f)$.*

Now we need to prove that monotonic lines don't cross each other. Here two distinct monotonic lines l_a and l_b don't cross each other doesn't mean $l_a \bigcap l_b = \emptyset$, since they can intersect tangentially. See Figure 6.

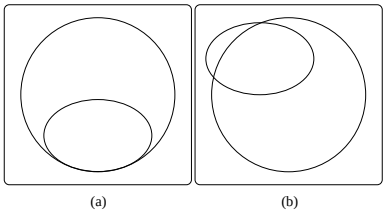

Figure 6: (a) l_a intersects l_b tangentially, (b) l_a crosses l_b.

Theorem 3.1 ***Theorem of No Crossing Monotonic Lines*** *For any image represented by function f, and any $\partial X, \partial Y$ in $MonotonicLineSet(f)$, one of following must be true: $X \subseteq Y$, $Y \subseteq X$ or $X \bigcap Y = \emptyset$.*

4 Monotonic Tree and Reduced Monotonic Tree

Now we can define relationships on $MonotonicLineSet(f)$.

Definition 4.1 *For any image represented by function f and any distinct monotonic lines $\partial X, \partial Y \in MonotonicLineSet(f)$, ∂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 $\partial Z \in MonotonicLineSet(f)$ such that $X \supset Z \supset Y$.*

The relationship $DirectEnclose$ is a parent-child relationship on $MonotonicLineSet(f)$.

Theorem 4.1 *For any image represented by function f, let Ω be its domain. Then $< MonotonicLineSet(f), DirectEnclose >$ is a rooted tree, and $\partial\Omega$ is its root.*

Definition 4.2 *For any image represented by function f, the tree $< MonotonicLineSet(f), DirectEnclose >$ is called the **monotonic tree** of f, denoted as $MonotonicTree(f)$.*

The monotonic tree can be reduced.

Definition 4.3 *For any image represented by function f and any $l_a, l_b \in MonotonicLineSet(f)$, l_a **uniquely directly encloses** l_b, denoted as $UniqueDirectEnclose(l_a, l_b)$, if*

(1) $DirectEnclose(l_a, l_b)$; and

(2) $\forall l_c \in MonotonicLineSet(f)$, if $DirectEnclose(l_a, l_c)$, then $l_b = l_c$.

Definition 4.4 *For any image represented by function f, a **monotonic slope** s is a maximal sequence of monotonic lines $s = \{l_i\}_{i=1}^n$ with $n \geq 1$ such that $\forall i = 1, 2, ..., n-1$, $UniqueDirectEnclose(l_i, l_{i+1})$. The first monotonic line l_1 is called the **enclosing line** of the slope s. The set of all monotonic slopes is denoted as $MonotonicSlopeSet(f)$.*

It's easy to see that for two monotonic slopes s_a and s_b in an image, if $s_a \neq s_b$, then $s_a \bigcap s_b = \emptyset$.

Definition 4.5 *For any image represented by function f and any $s_a, s_b \in MonotonicSlopeSet(f)$, 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 relationship $DirectEnclose$ is a parent-child relationship on $MonotonicSlopeSet(f)$.

Theorem 4.2 *For any image represented by function f, let Ω be its domain. Then $< MonotonicSlopeSet(f), DirectEnclose >$ is a rooted tree, and the monotonic slope which contains $\partial\Omega$ is the root.*

Definition 4.6 *For any image represented by function f, $< MonotonicSlopeSet(f), DirectEnclose >$ is called the **reduced monotonic tree** of f, denoted as $ReducedMonotonicTree(f)$.*

5 Image Smoothing and Texton Retrieval

In this section, we present a smoothing scheme based on the reduced monotonic tree. The basic idea is to cut the

branches (i.e., subtrees) in $ReducedMonotonicTree(f)$ which are small in the covered area.

Definition 5.1 ***Enclosing Line, Covered Region, Covered Area***

For any image represented by function f, and any branch (i.e., subtree) B of $ReducedMonotonicTree(f)$, let ∂X be the enclosing line of B's root. We define the enclosing line of B to be ∂X, the covered region of B to be X, and the covered area of B to be the area of X.

Definition 5.2 ***Maximal Branch***

For any image represented by function f, and some scale threshold T, a branch B of the reduced monotonic tree is called a maximal branch under T if

(1) its covered area $\leq T$; and

(2) there is no other branch C such that $B \subset C$ and C's covered area $\leq T$.

Definition 5.3 ***CuttingValue***

For any image represented by function f, and any branch B of $ReducedMonotonicTree(f)$, let $F = Extension(f)$ and ∂X be the enclosing line of B. If ∂X is outward falling, we define the cutting value of B to be:

$$CuttingValue(B; f) = \max_{x \in OutBorderPixelSet_{6.1}(X)} F(x);$$

else, we define:

$$CuttingValue(B; f) = \min_{x \in OutBorderPixelSet_{6.1}(X)} F(x).$$

It's easy to see that if $B \neq ReducedMonotonicTree(f)$, then $CuttingValue(B; f)$ is always finite.

Lemma 5.1 *For any image represented by function f, and some scale threshold T, let $\{B_i\}_{i=1}^n$ be the set of the maximal branches in $ReducedMonotonicTree(f)$ under T. For each B_i, let X_i be its covered region. Then we have:*

$$\forall i \neq j, X_i \bigcap X_j = \emptyset.$$

Based on the lemma above, we can define:

Definition 5.4 ***Smoothing by Cutting Branches***

Let T be some scale threshold. For any image represented by function f, let Ω be its domain and $\{B_i\}_{i=1}^n$ be the set of the maximal branches under T. For each B_i, let X_i be its covered region. We define the approximation of f by cutting $\{B_i\}_{i=1}^n$ to be a function $g : \Omega \to \mathbb{R}$ such that:

$$g(x) = \begin{cases} CuttingValue(B_i; f) & \text{if } x \in X_i \text{ for some } i; \\ f(x) & \text{otherwise.} \end{cases}$$

Denote g as $SmoothingCB(f; T)$.

The aim of smoothing is to remove small-scale details and get the large-scale features. On the other hand, texture retrieval is a process to retrieve and analyze the features of regular small-scale details.

Definition 5.5 *For any image represented by function f, and any scale threshold T, the set of maximal branches under T is denoted as $MaximalBranchSet(f; T)$. A maximal branch is called **positive** if its enclosing line is outward falling; it's called **negative** if its enclosing line is outward climbing.*

If the elements in $MaximalBranchSet(f; T)$ are regular in their shape and permutation, each of them is a texton. Under the reduced monotonic tree, the small-scale details are formally represented by the set $MaximalBranchSet(f; T)$, which provides a base for other techniques such as point pattern recognition, shape analysis and statistics.

6 Experiments

In this section, we include four tests on smoothing by cutting maximal branches. Maximal branches are mainly shown in black-white way: positive maximal branches are shown in white color (gray level 255); negative ones are shown in black color (gray level 0); and the background is shown in gray color (gray level 127). Due to the limited space, only for test two, we show the maximal branches in gray colors. For these examples, the scale thresholds are selected manually so that noise and texture are displayed clearly at these scales.

Test one is shown in Figure 7. This test is a representative of our tests on smoothing, which show that our smoothing scheme is successful in noise reduction while maintaining edge retention.

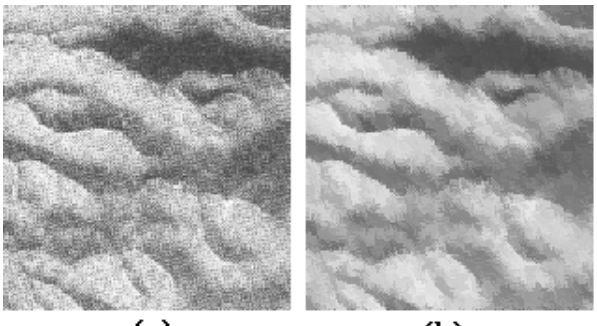

Figure 7: Test one: (a) original image I_1 with size 150×160, (b) smoothed image $SmoothingCB(I_1; 50)$.

Test two is shown in Figure 8. In this test, the stars in the national flag form very regular texture. Each star appears

in a positive maximal branch. Moreover, the characteristic features (such as eyes, nose, mouth) in the face are also retrieved as maximal branches, which means the reduced monotonic tree can be used for object recognition too.

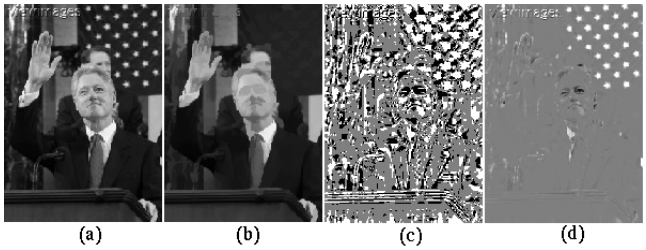

Figure 8: Test two: (a) original image I_2 with size 148×204, (b) $SmoothingCB(I_2; 80)$, (c)$MaximalBranchSet(I_2; 80)$ in black-white, and (d) $MaximalBranchSet(I_2; 80)$ in gray colors.

In test three (Figure 9), two facts complicate texture retrieval: (1) the image is noisy; and (2) the texture of the orange itself has complicated structures. We take a multiscale approach for this example. We smooth the image step by step with scales 10, 50 and 300. We can see that the maximal branches being cut in the first step (Figure 9(b)) mainly correspond to noise. The maximal branches being cut in the second (Figure 9(c)(e)) and last (Figure 9(d)(f)) steps correspond to texture structures at different scales. For this example, the concentric feature of the texture is better captured by the maximal branches at scale 50.

In the last test (Figures 10 through 12), we again take multiscale approach. In this example, different kinds of textures coexist: stairway, trees, and water. The difference of these textures is clearly displayed in the maximal branches being retrieved at the second step (Figure 12).

Of course, in these examples, the scale thresholds are chosen manually, and the structures in the maximal branches are viewed and judged by human beings. However, from these examples we can see that if we combine our scheme with other techniques such as point pattern recognition, shape analysis, and statistics, we will soon realize our dream of automatically removing noise and detecting texture in images. In addition, the reduced monotonic tree is a representation of structures at all scales in an image. Thus it can also be used to retrieve and analyze structures at large scales, which will help object detection and recognition.

7 Conclusion

In this paper, we introduced a model termed monotonic tree, which has advantages over the contour trees in image processing. The reduced monotonic tree can be used as a hierarchical representation of image structures. In particular,

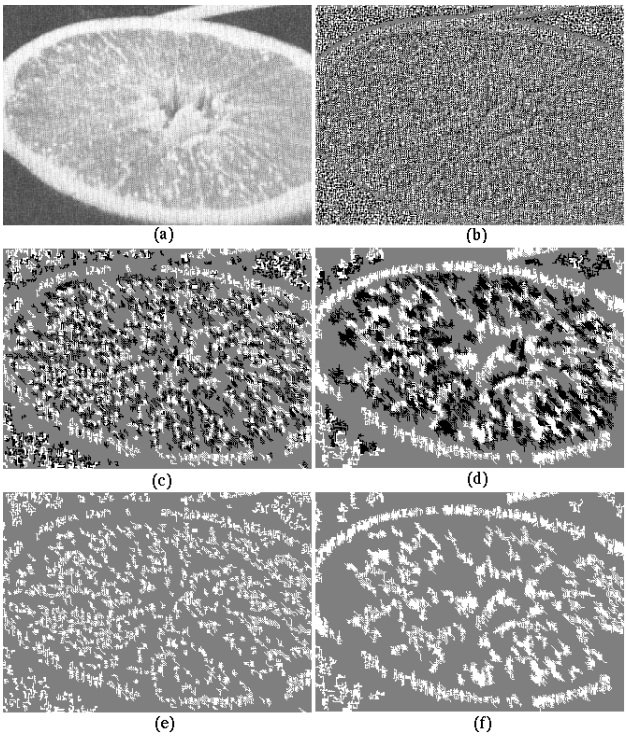

Figure 9: Test three: (a) original image I_3 with size 330×236, (b)$MaximalBranchSet(I_3; 10)$, (c)$MaximalBranchSet(I_3^{10}; 50)$, where $I_3^{10} = SmoothingCB(I_3; 10)$, (d) $MaximalBranchSet(I_3^{50}; 300)$, where $I_3^{50} = SmoothingCB(I_3^{10}; 50)$, (e) positive branches in $MaximalBranchSet(I_3^{10}; 50)$, and (f) positive branches in $MaximalBranchSet(I_3^{50}; 300)$.

we demonstrated the application of the reduced monotonic tree on image smoothing and texture retrieval. The reduced monotonic tree can also be used to retrieve and analyze structures at large scales, which will help object detection and recognition.

References

[1] M.J. Black and G. Sapiro. Edges as outliers: Anisotropic smoothing using local image statistics. In *Scale-Space Theory in Computer Vision*, pages 259–270, 1999.

[2] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(1):25–39, January 1983.

[3] 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.

Figure 10: Test four: (a) original image I_4 with size 400×320.

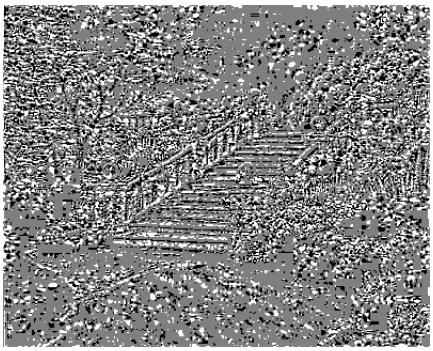

Figure 11: Test four: (b) $MaximalBranchSet(I_4; 30)$.

Figure 12: Test four: (c) $MaximalBranchSet(I_4^{30}; 200)$, where $I_4^{30} = SmoothingCB(I_4; 30)$.

[4] M. Kuwahara et al. Processing of ri-angiocardiographic images. *Digital Processing of Biomedical Images*, 1976.

[5] F. Farrokhnia and A.K. Jain. A multi-channel filtering approach to texture segmentation. In *CVPR91*, pages 364–370, 1991.

[6] L. Gagnon and F. Drissi-Smaili. Speckle noise reduction of airborne sar images with symmetric daubechies wavelets. *Performance Evaluation Of Signal And Image Processing Systems*, 2759:14–24, April 1996.

[7] J. Kacur and K. Mikula. Solution of nonlinear diffusion appearing in image smoothing and edge detection. *Applied Numerical Mathematics*, 50:47–59, 1995.

[8] C.H. Li, P.C. Yuen, and P.K.S. Tam. A probabilistic image model for smoothing and compression. In *Proceedings of the The International Conference on Information Technology: Coding and Computing*, 2000.

[9] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[10] R. Malladi and J.A. Sethian. Flows under min/max curvature flow and mean curvature: Applications in image processing. In *European Conference on Computer Vision*, pages I:251–262, 1996.

[11] S. Morse. Concepts of use in computer map processing. *Communications of the ACM*, 12(3):147–152, March 1969.

[12] J. Roubal and T.K. Peucker. Automated contour labelling and the contour tree. In *Proc. AUTO-CARTO 7*, pages 472–481, 1985.

[13] H. Tamura, T. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Trans. Systems, Man and Cybernetics*, 8:460–473, June 1978.

[14] M. Tuceryan and A. K. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, pages 207–248. World Scientific Publishing Co., 1998.

[15] M. Tuceryan and A.K. Jain. Texture segmentation using voronoi polygons. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(2):211–216, February 1990.

[16] M. van Kreveld. Efficient methods for isoline extraction from a tin. *International Journal of GIS*, 10:523–540, 1996.

[17] 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.

[18] N. Weyrich and G.T. Warhola. Wavelet shrinkage and generalized cross-validation for image denoising.

IEEE Trans. Image Processing, 7(1):82–90, January 1998.