Recursive Erosion, Dilation, Opening, and Closing Transforms

Su Chen and Robert M. Haralick, *Fellow, IEEE*

***Abstract*— In this paper, a new group of recursive morphological transforms on the discrete space Z^2 are discussed. The set of transforms include the recursive erosion transform (RET), the recursive dilation transform (RDT), the recursive opening transform (ROT), and the recursive closing transform (RCT). The transforms are able to compute in constant time per pixel erosions, dilations, openings, and closings with all sized structuring elements simultaneously. They offer a solution to some vision tasks that need to perform a morphological operation but where the size of the structuring element has to be determined after a morphological examination of the content of the image.**

The computational complexities of the transforms show that the recursive erosion and dilation transform can be done in $N + 2$ operations per pixel, where N is the number of pixels in the base structuring element. The recursive opening and closing transform can be done in $14N$ operations per pixel based on our experimental results.

I. Introduction

THE theory of mathematical morphology has been developed over the years by many researchers [6] [7]. The most common operations of mathematical morphology are the erosions, dilations, openings, and closings. The morphological operations are related to the Minkowski addition and subtraction of subsets in Euclidean space, using translations, unions, and intersections.

The morphological methods have excelled in image analysis techniques because of its sound mathematical basis and nonlinear nature. There are areas where morphological methods have been successfully applied, including image enhancement, segmentation, shape analysis, texture analysis, etc.

Successful application of mathematical morphology often critically depends on the choice of the size and shape of the structuring elements. Therefore, more advanced morphological algorithms are able to adaptively determine the shape and size of the structuring elements after first examining the contents of the input image. Generally, this process will involve a search that may be computationally expensive. However, in some vision tasks, we do have some prior knowledge on the shape of the structuring elements, and only the size of the structuring elements needs to be adaptively determined.

The general formulation of this type of problems can be as follows: Suppose K is any shaped structuring element containing the origin, i.e., $O \in K$. We are interested in performing the morphological operations with respect to the following set of structuring elements:

$$(\oplus_n K) = \begin{cases} \{O\} & \text{if } n = 0 \\ \underbrace{K \oplus K \oplus \cdots \oplus K}_{n} & \text{if } n = 1, 2, 3, \ldots. \end{cases}$$

By introducing a new set of recursive morphological transforms, namely, the recursive erosion transform (RET), the recursive dilation transform (RDT), the recursive opening transform (ROT), and the recursive closing transform (RCT), we are able to provide an elegant solution to the above problem. The transforms offers a novel paradigm for the computation of morphological operations on binary images.

In the next section, we will first review the definitions, the notations, and some of the well-known properties of the basic morphological operations. Then, we will give some insight on the recursive operators. In Section III, we will discuss the erosion transform (ET) and its relationship to the binary morphological erosions. An efficient two-pass recursive algorithm to compute the ET is described. In Section IV, we will define the dilation transform (DT) and show how the binary dilations with any sized structuring element can be computed via an efficient two-pass recursive DT algorithm. In Section V, the opening transform (OT) is discussed, and a two-pass recursive OT algorithm is developed. The ROT can be used to compute binary openings with arbitrary-sized structuring elements. In Section VI, we will introduce the closing transform (CT) and show how the binary closings with any sized structuring element can be computed via an efficient two-pass recursive CT algorithm. Finally, in Section VII, we will give some experimental results.

II. Definitions and Notations

In this section, we will first review the binary morphological operations on discrete space Z^2. Then, a short introduction to the recursive operators is provided.

A. Review of Mathematical Morphology

Mathematical morphology is a general method for image processing based on set theory [6] [7]. Images are considered as sets of points on which set operations can be performed, such as the translation, union, and intersection. Let A, B, C, K denote sets in Z^2 and O be the origin of Z^2.

Definition 1: The *translation* of A by $z \in Z^2$ is denoted by A_z and is defined as $A_z = \{a + z \mid a \in A\}$.

Definition 2: The *reflection* of a set K is denoted by $\check{K}$ and is defined as $\check{K} = \{-x \mid x \in K\}$.

Manuscript received May 18, 1993; revised December 13, 1993. The associate editor coordinating the review of this paper and approving it for publication was Prof. Nikolas P. Galatsanos.

The authors are with the Intelligent Systems Laboratory, Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA.

IEE Log Number 9408205.

Definition 3: The *erosion* of A by a structuring element K is denoted by $A \ominus K$ and is defined as $A \ominus K = \{x \in Z^2 \mid x + b \in A \text{ for every } b \in K\}$.

Geometrically, the erosion operation can be interpreted as translating set A by all points in the set $\check{K}$ and then taking the intersection, i.e., $A \ominus K = \cap\{A_b \mid b \in \check{K}\}$.

Definition 4: The *dilation* of A by a structuring element K is denoted by $A \oplus K$ and is defined by $A \oplus K = \{c \in Z^2 \mid c = a + b \text{ for some } a \in A \text{ and } b \in K\}$.

Geometrically, the dilation operation can be viewed as the union of all translates of set A by points in the set K, i.e., $A \oplus K = \cup\{A_b \mid b \in K\}$. Thus, dilation operations are commutative and associative: $A \oplus B = B \oplus A$, $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

Dilation is a dual operation of erosion. Their duality relation can be expressed as $A \oplus B = (A^c \ominus \check{K})^c$, where A^c denotes the set-theoretic complement of A.

Definition 5: The *opening* of a set A by a structuring element K is denoted by $A \circ K$ and is defined as $A \circ K = (A \ominus K) \oplus K$.

Geometrically, the opening can be characterized as the union of all translations of the structuring element K that fit inside the set A, i.e., $A \circ K = \cup\{K_t \mid K_t \subseteq A, t \in Z^2\}$.

Definition 6: The *closing* of a set A by a structuring element K is denoted by $A \bullet K$ and is defined as $A \bullet K = (A \oplus K) \ominus K$.

Geometrically, the closing includes any points satisfying the condition that anytime the point can be covered by some translation of $\check{K}$, the intersection between the translated $\check{K}$ and A is not empty, i.e., $A \bullet K = \{x \mid x \in \check{K}_t \text{ for some } t \text{ implies } \check{K}_t \cap A \neq \phi\}$.

Similar to the dilation operation, closing is a dual operation of opening. Their duality relation can be expressed as $A \bullet B = (A^c \circ \check{K})^c$.

Definition 7: The *n-fold dilation* of a set K is denoted by $(\oplus_n K)$ and is defined as

$$(\oplus_n K) = \begin{cases} \{O\} & \text{if } n = 0 \\ \underbrace{K \oplus K \oplus \cdots \oplus K}_{n} & \text{if } n = 1, 2, 3, \ldots. \end{cases}$$

$(\oplus_n K)$ is extensive under increasing n when K contains the origin. Consequently, we define n as the scale factor of the structuring element $(\oplus_n K)$ and K as its base.

B. Review of Recursive Operators

The *recursive operator* is a operator whose output depends not only on the input pixels that lie within the domain of its kernel but also on one or more previously computed output values.

Associated with a particular recursive operator will be a particular scan order of input pixels. This scan order governs the order in which the operation is to be applied. A scanning function can be used to uniquely specify such a scan order.

Definition 8: The *scanning function* S is defined as a one-to-one onto mapping from a finite set $I = \{(x_1, x_2) \in Z^2 \mid 0 \leq x_1 < m_1, 0 \leq x_2 < m_2\}$ to the set $\{1, 2, \ldots, m_1 m_2\}$. If $p \in I, q \in I$ and $S(p) < S(q)$, then the output value at p is computed before the output value at q.

Definition 9: The *reverse scanning function* S^{-1} of a scanning function S is defined as a one-to-one onto mapping from a finite set $I \subseteq Z^2$ to the set $\{1, 2, \ldots, m_1 m_2\}$ such that $S^{-1}(p) = m_1 m_2 + 1 - S(p)$.

Definition 10: A *sequentially computable* recursive operator is a recursive operator with respect to a scanning function S in the finite set $I \subseteq Z^2$ such that whenever an output value at $p \in I$ is computed, it only depends on input pixel values and those output pixel values at $q \in I$ satisfying $S(q) < S(p)$.

Definition 11: The *decomposition* of a kernel K containing the origin by a scanning function S is defined as the two subsets K^f and K^b of K, where

$$\begin{cases} K = K^f \cup K^b \cup O \\ K^f = \{q \in K \mid S(q + p) < S(p), \text{ for all p} \in I \subseteq Z^2\} \\ K^b = \{q \in K \mid S(q + p) > S(p), \text{ for all p} \in I \subseteq Z^2\}. \end{cases}$$

Furthermore, we define K^f as the forward subkernel of K because K^f defines the domain of those points in K on which the current output are depending in the scanning S. In the same way, we define K^b as the backward subkernel of K because K^b defines the domain of those points in K on which the current output are relying in the scanning S^{-1}.

III. Recursive Erosion Transform

We begin with the discussion of our first recursive morphological transform—the recursive erosion transform (RET). The RET offers an efficient way to compute the erosions of a binary image with an arbitrarily sized structuring element.

A. Definition

The erosion transform of a binary image is based on the successive morphological erosions on the image. It is a generalization of the distance transform commonly known in the literature [1].

Given a binary image I, denote by A the set of all the binary one pixels (which are also called the "feature pixels" or "foreground pixels"). The erosion transform of A with respect to a structuring element K produces a gray-scale image where the gray level of each pixel $x \in A$ is the generalized distance of x to the image background, i.e., the largest positive integer n such that $x \in A \ominus \{\oplus_{n-1} K\}$. The generalized distance at a pixel x indicates the maximum number of consecutive erosions of A by K such that x is still contained in the eroded image foreground.

Definition 12: The *erosion transform* of a set $A \subseteq Z^2$ with respect to a structuring element $K \subseteq Z^2$ is denoted by $\mathrm{ET}[A, K]$ and is defined as

$$\mathrm{ET}[A, K](x) = \begin{cases} \max\{n \mid x \in A \ominus_{n-1} K\} & \text{if } x \in A \\ 0 & \text{if } x \notin A, \end{cases}$$

where $A \ominus_n K = A \ominus (\oplus_n K)$.

From the above definition, one can easily see that the support domain of $\mathrm{ET}[A, K]$ is A and that it has the following thresholding property. Once the ET has been computed, it only requires a simple thresholding to compute a binary erosion

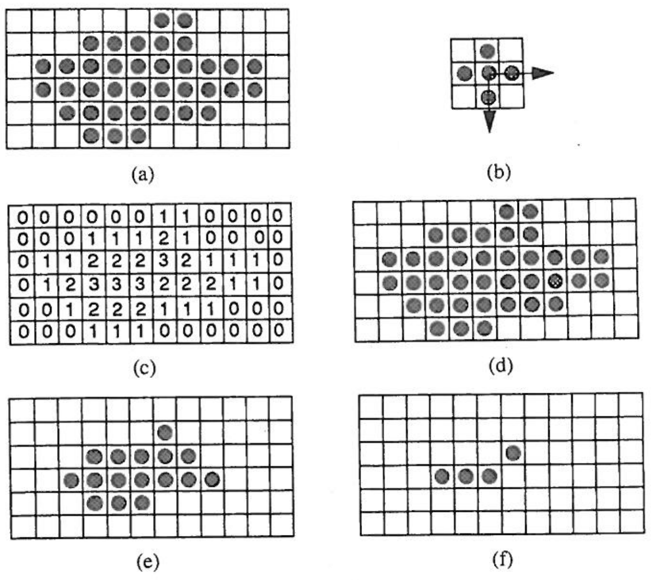


Fig. 1. Instance of the erosion transform: (a) 2-D biary image; (b) diamond-shaped structuring element; (c) ET of the binary image; (d) threshold ET image ($n = 0$); (e) thresholded ET image ($n = 1$); (f) thresholded ET image ($n = 2$).

with any sized structuring element. The erosion transform and its subsequent thresholding processes are illustrated in Fig. 1.

Proposition 1: Let n be a nonnegative integer. If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element containing the origin $O \in K$, and $B_n = \{x \in A \mid \mathrm{ET}[A, K](x) > n\}$, and then, $B_n = A \ominus (\oplus_n K)$.

Fig. 1 illustrates an instance of the erosion transform. The brute-force iterative algorithm for computing the erosion transform is constructed by successively eroding the input image, i.e.

$$\begin{cases} A_1 & = A \\ A_{n+1} & = A_n \ominus K, n = 1, 2, \dots. \end{cases}$$

The erosion transform at $x \in A$ is computed as the maximum n such that $x \in A_n$. Its worst-case computational complexity is $MN/2$ operations/pixel for a N-point structuring element and a $M \times M$ input image if we assume that $M/2$ erosions is required [3]. In the next section, we will describe an efficient two-pass recursive erosion transform (RET) algorithm that requires only a maximum of $N + 2$ operations/pixel.

B. Recursive Erosion Transform Algorithm

In their pioneering work [1], Rosenfeld and Pfaltz described a two-pass recursive algorithm to compute the city-block and chess-board distance transform of a binary image. The algorithm is based on the following idea: Global distances in the image can be calculated by propagating local distances, i.e., distances between neighboring pixels. As a generalization, Bertrand [2], [4] and Haralick [3] described a two-pass recursive algorithm to compute what they call the generalized distance transform. In the morphological terminology, the generalized distance transform is nothing but the erosion transform that we defined previously. The distance transform using the four-neighborhood corresponds to an erosion transform with a five-pixel cross structuring element, and the distance transform using an eight-neighborhood corresponds to an erosion transform with a 3×3 box structuring element.

The recursive ET algorithm computes the erosion transform in only two passes over the image; the first pass goes from pixel to pixel in a top-to-bottom and left-to-right sequence (forward pass), and the second pass goes in the reverse direction (backward pass). In each pass, a subset of the local neighborhood of each pixel is examined, and a recursive operator is applied. The derivations of the neighborhood recursive operators are provided in [2] and [3]. As a quick reference, we list below the two propositions from [2] and [3] that immediately lead to the two-pass RET algorithm.

Proposition 2 establishes the local recursive property of the ET. It indicates that the ET at any pixel is one plus the mimimum of its neighborhood ET values.

Proposition 2: If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element containing the origin $O \in K$ and $\hat{K} = \{x \in K \mid x \neq O\}$, then

$$\mathrm{ET}[A, K](x) = \begin{cases} \min\{\mathrm{ET}[A, K](x + b) \mid b \in \hat{K}\} + 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

Generally, the above form of recursive operator is not sequentially computable. However, by observing that the recursive operator consists of a minimization operator and has the properties that it is monotonically decreasing, commutative, and associative, we can decompose the whole minimization process into two subprocesses, where each of them is sequentially computable. This is the essence of the following Proposition 3, which shows that the ET can be computed in a two-pass recursive algorithm.

Proposition 3: If $A \subseteq Z^2$ is a set, and K^f, K^b are a decomposition of the structuring element $K \subseteq Z^2$ by a scanning function S, then $\mathrm{ET}[A, K](x) = \min\{\mathrm{ET}[A, K^f](x), \min\{\mathrm{ET}[A, K](x + b) \mid b \in K^b\} + 1\}$.

C. Algorithm Description

Let $A \subseteq Z^2$ be a set, and let $K \subseteq Z^2$ be a structuring element containing the origin. Let the scanning function S be chosen as the order of top-to-bottom and left-to-right. Accordingly, let the reverse scanning function S^{-1} be the order of bottom-to-top and right-to-left. The RET algorithm can be summarized as follows:

Algorithm Recursive Erosion Transform—2D Case

1) Decompose the K with respect to the scanning function S: The forward structuring element will be K^f, and the backward structuring element will be K^b.
2) In the first pass of RET (forward scanning function S), perform the following filtering on each pixel x in the input image (Proposition 2):
   - If $x \notin A$, then $\mathrm{ET}[A, K^f](x) = 0$.
   - If $x \in A$, then $\mathrm{ET}[A, K^f](x) = \min\{\mathrm{ET}[A, K^f](x + f) \mid f \in K^f\} + 1$.
3) In the second pass of RET (backward scanning function S^{-1}), perform the following filtering on each pixel x on the output of the first pass (Proposition 3):
   - If $\mathrm{ET}[A, K^f](x) = 0$, then $\mathrm{ET}[A, K](x) = 0$.

- Else $\mathrm{ET}[A,K](x) = \min\{\mathrm{ET}[A,K^f](x), \min\{\mathrm{ET}[A,K](x+b) \mid b \in K^b\} + 1\}$.

4) End.

D. Computational Complexity

Suppose there are N^f points in K^f and N^b points in K^b. For a N-point structuring element ($N = N^f + N^b + 1$), the number of operations per pixel in the forward scanning is $N^f + 1$ for the binary one pixel (N^f comparisons and 1 addition) and 1 for a binary zero pixel (one comparison). The number of operations per pixel in the backward scanning is $N^b + 2$ for the binary one pixel ($N^b + 1$ comparisons and one addition) and 1 for a binary zero pixel (one comparison). As a result, the recursive ET algorithm requires $N + 2$ operations per binary one pixel and two operations per binary zero pixel. The experimental results on the timing performance of RET will be presented in Section VII.

IV. Recursive Dilation Transform

In this section, we discuss the dual transform of the RET—the recursive dilation transform (RDT). It provides an efficient way to compute the dilations of a binary image with any sized structuring element.

A. Definition

The dilation transform of a binary image is based on the successive morphological dilations of the image. The dilation transform of a set A with respect to a structuring element K generates a gray-scale image where the gray level of each pixel $x \in Z^2$ is the generalized distance of x to set A, i.e., the smallest positive integer n so that $x \in A \oplus (\oplus_{n-1} K)$. If no such n exists, where $x \notin A \oplus (\oplus_{n-1} K)$ for all n, then the dilation transform at $x \in Z^2$ is designated as zero.

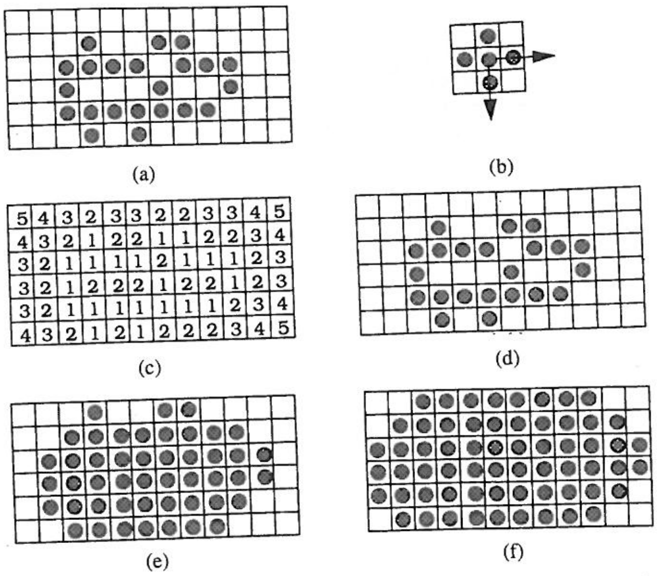
Definition 13: The dilation transform of a set $A \subseteq Z^2$ by a structuring element $K \subseteq Z^2$ is denoted by $\mathrm{DT}[A,K]$ and is defined as

$$\mathrm{DT}[A,K](x) = \begin{cases} \min\{n \mid x \in A \oplus (\oplus_{n-1} K)\} & \text{if } \exists n, x \in A \oplus (\oplus_{n-1} K) \\ 0 & \text{if } \forall n, x \notin A \oplus (\oplus_{n-1} K). \end{cases}$$

The above definition indicates that the support domain of $\mathrm{DT}[A,K]$ is Z^2 because dilation is extensive. The following Proposition 4 shows how to apply the dilation transform to compute the binary dilation with an arbitrary sized structuring element. The dilation transform and its subsequent thresholding processes are illustrated in Fig. 2.

Proposition 4: Let n be a positive integer. If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element containing the origin and $B_n = \{x \in Z^2 \mid 0 < \mathrm{DT}[A,K](x) \leq n\}$, then $B_n = A \oplus (\oplus_{n-1} K)$.

The dilation transform can be computed in two ways. One is the brute-force iterative algorithm, and the other is a recursive algorithm based on the duality relation between the DT and the ET. The brute-force algorithm requires successively dilating



5	4	3	2	3	3	2	2	3	3	4	5
4	3	2	1	2	2	1	1	2	2	3	4
3	2	1	1	1	1	2	1	1	1	2	3
3	2	1	2	2	2	1	2	2	1	2	3
3	2	1	1	1	1	1	1	1	2	3	4
4	3	2	1	2	1	2	2	2	3	4	5

Fig. 2. Instance of the dilation transform. The dotted boundary indicates that the domain of the image extends to infinity. DT in a window is shown: (a) 2-D binary image; (b) diamond-shaped structuring element; (c) DT of the binary image; (d) thresholded ET image ($n = 1$); (e) thresholded DT image ($n = 2$); (f) thresholded DT image ($n = 3$).

the input image

$$\begin{cases} A_1 & = A \\ A_{n+1} & = A_n \oplus K, n = 1, 2, \ldots. \end{cases}$$

The dilation transform at $x \in Z^2$ is computed as the minimum n such that $x \in A_n$. Suppose the DT image with a maximum value of $\rho + 1$ is to be produced for an input $M \times M$ binary image and N-point structuring element, the computational complexity for the brute-force algorithm would be ρN operations/pixel. When ρ is large, there is a lot of computation involved.

On the other hand, the algorithm based on the duality relation requires first taking the complement of the input image and then computing the RET on the complemented image with the reflected structuring element. There are difficulties in actual implementation of the algorithm because the complementation of a finite set is an infinite set, and we cannot consider the computation of RET on an infinite set. Furthermore, there are cases when the RET on the complemented image may generate infinite values in our finite domain of interest. Fig. 3 shows one example. When the origin of the structuring element is at the top-most left point of the structuring element, the RET will have infinite values near the upper-left corner of the input image.

The traditional wisdom to get around the above difficulties is to allow the complemented image to be expanded sufficiently large to guarantee some adhoc algorithms to work. Given an input $M \times M$ binary image and that a DT image with a maximum value of $\rho + 1$ is to be computed, then output DT image will be of size $[M + \rho(S_r - 1)] \times [M + \rho(S_c - 1)]$, where S_r and S_c are the row and column sizes of the structuring element, respectively. To guarantee the correct output DT image, a row expansion of $2\rho(S_r - 1)$ and a column expansion of $2\rho(S_c - 1)$ on the complemented image are required. This will result in an increase in the time and space complexity

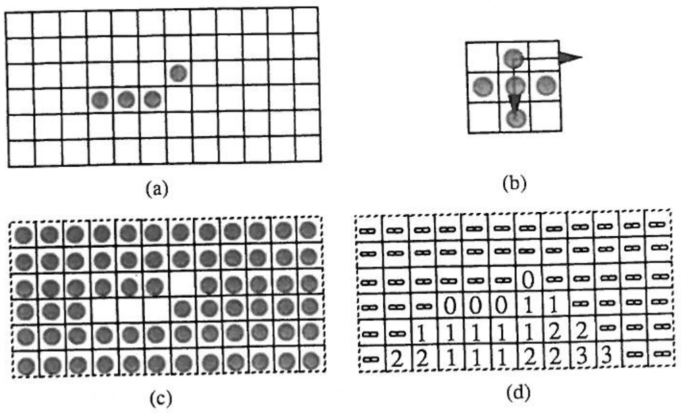


Fig. 3. Example where the RET on the complemented image generates infinite values in a finite domain of interest. The dotted boundary indicates that the domain of the image extends to infinity: (a) 2-D binary image; (b) diamond shaped structuring element, its origin being at the top-most left point; (c) the complemented binary image; (d) ET on the complemented image.

and is not trivial. In the next section, a simple two-pass recursive dilation algorithm that takes a maximum of $N+2$ operations/pixel is described. The algorithm does not require input image expansion.

B. Recursive Dilation Transform Algorithm

We have summarized the local recursive property of ET in Proposition 2. Since the dilation transform may be considered to be the dual transform of erosion transform, we can easily show that the dilation transform shares the similar property as that of ET.

Proposition 5 establishes the local recursive nature of the DT. It shows that the DT at any pixel can be computed as one plus the minimum of all its nonzero neighborhood DT values.

Proposition 5: Let $A \subseteq Z^2$ be a set, and let $K \subseteq Z^2$ be a structuring element containing the origin $O \in K$. Denote $\check{K}$ as the reflection of K and $\dot{K} = \{k \in \check{K} \mid k \neq O\}$. If $x \in Z^2$ and $C = \{(x+k) \mid \mathrm{DT}[A,K](x+k) > 0, k \in \dot{K}\}$ is the set of nonzero neighborhood pixels of x, then

$$\mathrm{DT}[A,K](x) = \begin{cases} 1 & \text{if } x \in A \\ \min\{DT[A,K](y) \mid y \in C\} + 1 & \text{if } x \notin A \text{ and } C \neq \phi \\ 0 & \text{otherwise.} \end{cases}$$

The above proposition will lead us to a recursive algorithm for computing the dilation transform. The algorithm calculates the DT at a pixel as one plus the minimum of all its nonzero neighborhood DT values. To avoid the direct computation of the set of nonzero neighborhood values at each pixel, we can initially reset all binary zero pixels in the input image to have an infinite value. By doing this, the minimization operator in Proposition 5 reduces to the same form as that is in Proposition 2. Consequently, we can compute the DT via a similar two-pass recursive algorithm employed in the RET.

C. Algorithm Description

Let $A \subseteq Z^2$ be a set, and let $K \subseteq Z^2$ be a structuring element containing the origin. Let the scanning function S be chosen as the order of left-to-right and top-to-bottom. Accordingly, let the reverse scanning function S^{-1} be the order of right-to-left and bottom-to-top.

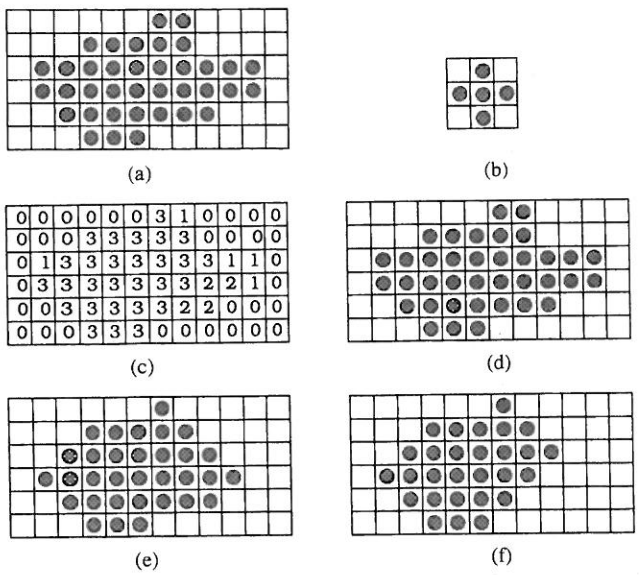
As indicated previously, the support domain of $\mathrm{DT}[A,K]$ is Z^2. Due to practical storage limitations, we constrain the DT output to have a maximum value of $\rho+1$. Consequently, we define $\mathrm{DT}^\rho = \{\mathrm{DT}[A,K](x) \leq \rho+1 \mid x \in Z^2\}$, where ρ is any chosen nonnegative integer. The following RDT algorithm computes DT^ρ for any given ρ. If the input is a $M \times M$ binary image, the output DT^ρ is a $[M+\rho(S_r-1)] \times [M+\rho(S_c-1)]$ image.

Algorithm Recursive Dilation Transform—2D Case

1) Reflect the K around its origin. Let the reflected structuring element be $\check{K}$.
2) Decompose the $\check{K}$ with respect to the scanning function S: The forward structuring element will be $\check{K}^f$; the backward structuring element will be $\check{K}^b$.
3) Initialization: Reset all binary zero pixels to have a very large value MAX_VALUE $> \rho+1$.
4) In the first pass of RDT (forward scanning function S), perform the following filtering on each pixel x in the input image (Proposition 5):
   - If $x \in A$, then $\mathrm{DT}[A,\check{K}^f](x) = 1$.
   - If $x \notin A$, then $\mathrm{DT}[A,\check{K}^f](x) = \min\{\mathrm{DT}[A,\check{K}^f](x+f) | f \in \check{K}^f\} + 1$.
5) In the second pass of RDT (backward scanning function S^{-1}), perform the following filtering on each pixel x in the output image of the first pass (Proposition 3):
   - If $\mathrm{DT}[A,\check{K}^f](x) = 1$, then $\mathrm{DT}[A,K](x) = 1$
   - Else $\mathrm{DT}[A,K](x) = \min\{\mathrm{DT}[A,\check{K}^f](x), \min\{\mathrm{DT}[A,K](x+b) | b \in \check{K}^b\} + 1\}$.
6) Reset all values in $\mathrm{DT}[A,K]$ that are greater than $\rho+1$ to 0.
7) End.

D. Computational Complexity

Suppose there are N^f points in $\check{K}^f$ and N^b points in $\check{K}^b$. For a N-point structuring element ($N = N^f + N^b + 1$), the forward scanning requires N^f+1 operations on a binary zero pixel (N^f comparisons and one addition) and one operation on a binary one pixel (one comparison); the backward scanning requires N^b+2 operations on a binary zero pixel (N^b+1 comparisons and and addition) and one operation on a binary one pixel (one comparison). As a result, the recursive RDT algorithm requires $N+2$ operations per binary zero pixel and two operations per binary one pixel. Since the RDT algorithm does not require input image expansion, it is space efficient. The experimental results on the timing performance of RDT will be presented in Section VII.



0	0	0	0	0	0	3	1	0	0	0	0
0	0	0	3	3	3	3	3	0	0	0	0
0	1	3	3	3	3	3	3	3	1	1	0
0	3	3	3	3	3	3	3	2	2	1	0
0	0	3	3	3	3	3	2	2	0	0	0
0	0	0	3	3	3	0	0	0	0	0	0

Fig. 4. Instance of the opening transform: (a) 2-D binary image; (b) diamond shaped structuring element; (c) OT of the binary image; (d) thresholded OT image ($n = 0$); (e) thresholded OT image ($n = 1$); (f) thresholded OT image ($n = 2$).

V. Recursive Opening Transform

The recursive opening transform efficiently computes the binary opening with any sized structuring element. It also provides a quick way to calculate the pattern spectrum [8] of an image. The pattern spectrum is nothing more than a histogram of the opening transform.

A. Definition

The opening transform of an image A with respect to a structuring element K puts in each binary one pixel $x \in A$ the largest positive integer n such that $x \in A \circ (\oplus_{n-1}K)$, which means that for some translation t, $(\oplus_{n-1}K)$ can be translated by t to cover x yet remain in A.

Definition 14: The *opening transform* of a set $A \subseteq Z^2$ by a structuring element $K \subseteq Z^2$ is denoted by $\mathrm{OT}[A,K]$ and is defined as

$$\mathrm{OT}[A,K](x) = \begin{cases} \max\{n \mid x \in A \circ (\oplus_{n-1}K)\} & \text{if } x \in A \\ 0 & \text{if } x \notin A. \end{cases}$$

From the above definition, one can easily see that the support domain of $\mathrm{OT}[A,K]$ is A and that the opening transform has the following properties. The first proposition shows how we can use the opening transform to compute the binary opening with an arbitrary sized structuring element. The next proposition indicates that the opening transform does not depend on the origin of K. Without loss of generality, we will always assume that the origin is contained in the structuring element. The opening transform and its subsequent thresholding processes are illustrated in Fig. 4.

Proposition 6: Let n be a nonnegative integer. If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element, and $B_n = \{x \in A \mid \mathrm{OT}[A,K](x) > n\}$; then, $B_n = A \circ (\oplus_n K)$.

Proposition 7: If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element, and $K' = (K)_t$ is a translation of K by $t \in Z^N$, then $\mathrm{OT}[A,K'](x) = \mathrm{OT}[A,K](x)$, for all x $\in A$.

The OT of an image can be computed via the following brute-force algorithm: $A_{n+1} = A \circ (\oplus_n K)$, $n = 0, 1, 2, \ldots$. The opening transform at $x \in A$ is computed as the maximum n such that $x \in A_n$. Suppose the input image size is $M \times M$ and that K is a N-point structuring element. The above brute-force algorithm requires $M(MN + 6N + 8)/8$ operations/pixel [5]. In comparison, the following two-pass recursive opening transform algorithm takes about $14N$ operations per pixel on the basis of our experimental results.

B. Recursive Opening Transform Algorithm

The ET at a given image pixel is defined as the generalized distance of the pixel to the image background, i.e., the maximum number of consecutive erosions of the image that the image pixel is still contained in the image foreground. Therefore, it also indicates how far one can dilate the given image pixel so that the dilated pattern is still contained in the image. As an example, suppose $\mathrm{ET}[A,K](x) = n$. From the definition of the ET, it is true that $x \in A \ominus (\oplus_{n-1}K)$, where n is defined to be as maximal as possible. Since the structuring element contains the origin $O \in K$, the extensive property of dilation will guarantee that $(\oplus_{n-1}K)_x \subseteq A \circ (\oplus_{n-1}K) \subseteq A$, where n is also the maximum.

The above observations suggest a procedure to compute the OT. In the first part of the procedure, the ET of the image is calculated. The efficient recursive ET algorithm described in Section III can be utilized to accomplish this task. In the second part of the procedure, each pixel in the ET image is maximally dilated based on its ET value. In the meantime, the ET value will be propagated along the way with the dilation process. The process can be described as a label propagation process. The OT at each image pixel is defined as the maximum of all labels that can be propagated to that pixel. Later in the section, we will explicitly state the relationship of the OT with the ET and the label propagation process (Proposition 8), and we will also demonstrate that the label propagation process can be recursively computed (Proposition 10). However, before going any further, we need first to introduce some additional terminology.

In order to describe a propagation process, we need to define a propagating entity. A propagating entity will indicate a label to be propagated and its corresponding propagating distance and center. For the recursive OT, the propagating center could be any pixel on the image; the ET value at that pixel is the label to be propagated, and the propagating distance is the ET value at that pixel minus one. We will adopt the following notion of propagator to delineate the propagating entity of the recursive OT.

A propagator has a pixel location, a label to be propagated, and a nonnegative integer (the propagating factor), indicating how many times the propagating label can yet be propagated.

Definition 15: A *propagator* on Z^2 is defined as a triple and is denoted by $\varphi = (l, f, x)$, where

l	propagating label that φ propagates
f	propagating factor with $f-1$ specifying the number of times that φ can yet propagate
x	propagating center indicating the location of φ.

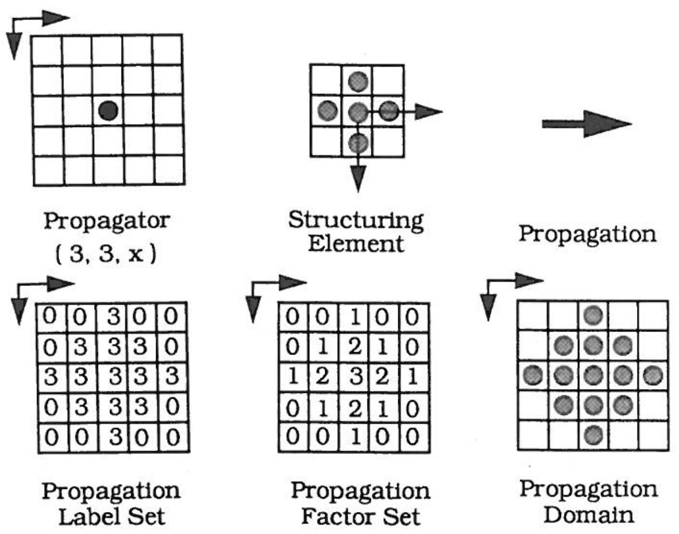


Fig. 5. Instance of the propagation process. x is (2, 2) in this case. The propagation $\mathcal{P}[\varphi, K]$ generates a set of nondummy propagators in the domain of $(\oplus_{f-1}K)_x$.

The label function $\mathcal{L}(\varphi) = l$ returns the propagating label of the propagator. The factor function $\mathcal{F}(\varphi) = f$ returns the propagating factor of the propagator.

A propagator can be acted upon by a propagation process $\mathcal{P}$ with a structuring element K. The propagation process $\mathcal{P}[\varphi, K]$ computes the set of pixels that lie within an $(f-1)$ generalized distance of x and propagates the propagating label l to those pixels.

Definition 16: The *propagation* $\mathcal{P}$ of a propagator $\varphi = (l, f, x)$ with a structuring element K is denoted by $\mathcal{P}[\varphi, K]$; it produces a propagator at each $z \in Z^2$. Let $\mathcal{P}[\varphi, K](z)$ denote the created propagator at $z \in Z^2$, and it is defined as

$$\mathcal{P}[\varphi, K](z) = \begin{cases} (l, \mathrm{ET}[(\oplus_{f-1}K)_x, K](z), z) & \text{if } z \in (\oplus_{f-1}K)_x \\ (0, 0, z) & \text{otherwise,} \end{cases}$$

where the operator $(\oplus_{f-1}K)_x$ means translating $(\oplus_{f-1}K)$ by the element x.

As shown in Fig. 5, the propagation $\mathcal{P}[\varphi, K]$ generates a set of nondummy propagators in the domain of $(\oplus_{f-1}K)_x$. The propagating label l of φ is propagated all the way to the extent of $(\oplus_{f-1}K)_x$, whereas the propagating factor at z is $\mathrm{ET}[(\oplus_{f-1}K)_x, K](z)$. Proposition 8 relates the OT to the ET and the propagation process.

Proposition 8: Let $A \subseteq Z^2$ be a set, let $O \in K \subseteq Z^2$ be a structuring element containing the origin, and let the ET of the set A with respect to K be denoted by $\mathrm{ET}[A, K]$. If we define

$$\varphi_z = \begin{cases} (\mathrm{ET}[A, K](z), \mathrm{ET}[A, K](z), z) & \text{if } z \in A \\ (0, 0, z) & \text{if } z \notin A, \end{cases}$$

and

$$B(x) = \begin{cases} \max\{\mathcal{L}\{\mathcal{P}[\varphi_z, K](x) \text{ for all } z \in A\} & \text{if } x \in A \\ 0 & \text{if } x \notin A, \end{cases}$$

then $B(x) = \mathrm{OT}[A, K](x)$ for all $x \in A$.

An interpretation of Proposition 8 is that given the ET of a set A with respect to a structuring element K, we first create a propagator φ_z at each point $z \in A$. Then, we propagate each φ_z in the domain of A and stack these propagations on top of each other according to their own origin z. If we define $B(x)$ to be the maximum propagating label among all the propagators at x, then $B(x) = \mathrm{OT}[A, K](x)$ for all $x \in A$.

The following Proposition 9 shows the relationship between the propagations of two or more propagators located at the same pixel.

Proposition 9: If $\varphi_1 = (l_1, f_1, x)$ and $\varphi_2 = (l_2, f_2, x)$ are two propagators at pixel x, $l_1 \geq l_2$, and $f_1 \geq f_2$,, then for all $z \in A$,

$$\mathcal{L}\{\mathcal{P}[\varphi_2, K](z)\} \leq \mathcal{L}\{\mathcal{P}[\varphi_1, K](z)\}.$$

The proposition shows that when there are multiple propagators at the same pixel, the propagations of some of them may not have an impact on the final result of the OT. Thus, those propagators can be pruned off. It is, therefore, useful to introduce the following concept of the propagator list.

Definition 17: The *propagator list* $\{\Phi(x) = \{(l_0, f_0, x), (l_1, f_1, x), \ldots, (l_m, f_m, x)\}$ at a propagating pixel x is defined as an ordered list of propagators that satisfy the following constraints:

- $l_i > l_j$, if $i < j$, where $i, j = 0, 1, \ldots, m$;.
- $f_i < f_j$, if $i < j$, where $i, j = 0, 1, \ldots, m$.

The propagators in $\Phi(x)$ are propagatable, and the sequence of the propagators in $\Phi(x)$ indicates the propagation sequence of the propagators. The foremost propagator propagates first.

C. Recursive Propagation Algorithm

In this section, we shall show that the propagation operation $\mathcal{P}[\varphi, K]$ can be computed recursively. We shall consider one special case where the origin O of the structuring element $K \subseteq Z^2$ is the top-most left point in the domain of K. In the 2-D case, the top-most left point of K can be defined in the following way: $O = (r_0, c_0)$, where $r_0 = \min\{r \mid (r, c) \in K\}$ is the minimum row coordinate of all the pixels in K, and $c_0 = \min\{c \mid (r_0, c) \in K\}$ is the minimum column coordinate of the pixels in K that have the minimum row coordinate r_0.

Under this assumption, the next proposition establishes that the propagation of a propagator can be done through a one-pass recursive algorithm.

Proposition 10: Let $\varphi = (l, f, x)$ be a propagator on Z^2, K be a structuring element with its origin O at the top-most left point of K, and $\check{K} = \{-x \mid x \in \mathrm{K} \text{ and } \mathrm{x} \neq O\}$ be a reflection of K. If $\mathcal{R}[\varphi, K]$ is defined as

$$\mathcal{R}[\varphi, K](z) = \begin{cases} (l, f', z) & \textit{if } z \in (\oplus_{f-1}K) \\ (0, 0, z) & \textit{if } z \notin (\oplus_{f-1}K) \end{cases}$$

where $f' = \max\{f \cdot \delta(z, x), \max\{\mathcal{F}\{\mathcal{R}[\varphi, K](z+b)\}, b \in \check{K}^b\} - 1\}$, and $\delta(z, x)$ is a δ-function in Z^2, i.e.

$$\delta(z, x) = \begin{cases} 1 & \textit{if } z = x \\ 0 & \textit{if } z \neq x, \end{cases}$$

then for all $z \in Z^2$, $\mathcal{P}[\varphi, K](z) = \mathcal{R}[\varphi, K](z)$.

The proposition states that the propagation $\mathcal{P}[\varphi, K]$ can be computed by scanning a propagator array that at first contains

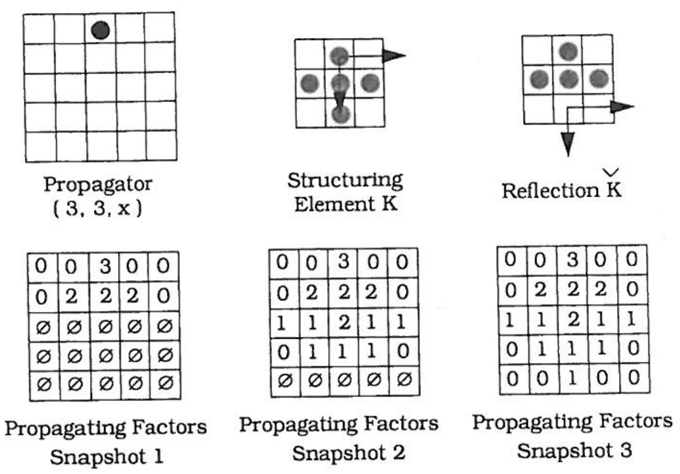


Fig. 6. Instance of the recursive propagation process. x is (0, 2) in this case. ϕ is used to identify those pixels that have not been scanned.

only one propagator φ. At each pixel other than the x on the scanning path, we assign the propagating factor to be the $\max\{0, f'' - 1\}$, where the f'' is the maximum propagating factor at those points covered by the $\check{K}^b$ (the origin of the $\check{K}^b$ is centered on the point). If the resulting propagating factor is nonzero, we assign the propagating label to be l; otherwise, assign the propagating label to be zero. Fig. 6 shows an example of the recursive propagation process.

D. Algorithm Description

Let $A \subseteq Z^2$ be a set, and let $K \subseteq Z^2$ be a structuring element containing the origin. Let the scanning function S be chosen as the order of left-to-right and top-to-bottom. Accordingly, let the reverse scanning function S^{-1} be the order of right-to-left and bottom-to-top. The ROT algorithm can be summarized as follows:

Algorithm Recursive Opening Transform—2D Case

1) Set the origin O of K to be the top-most left point in K.
2) The first pass of ROT (scanning function S^{-1}): Execute the RET algorithm described in Section III to compute $\mathrm{ET}[A, K](x)$. Only the backward pass of RET is needed.
3) Decompose the K with respect to the scanning function S. Let K^f, K^b be the corresponding forward and backward structuring element. In this particular case, K^f is an empty set ϕ.
4) Flip the backward filtering structuring element K^b, and let it be $\check{K}^b$.
5) The second pass of ROT (scanning function S): For each pixel x on the scanning path, do the following: Let $\Phi(x)$ be the propagator list at pixel x if the following hold:
   - If $\mathrm{ET}[A, K](x) = 0$, set $\Phi(x) = \{(0, 0, x)\}$.
   - If $\mathrm{ET}[A, K](x) \neq 0$, perform the following steps: Let S_0, S_n be propagator sets.
     (a) $S_0 = \{(l_0, f_0, x) \mid l_0 = \mathrm{ET}[A, K](x), f_0 = \mathrm{ET}[A, K](x)\}$.
     (b) $S_n = \{(l, f - 1, x) \mid (l, f, x + b) \in \Phi(x + b), b \in \check{K}^b \text{ and } f > 1\}$.
     (c) $\Phi(x) =$ *MakePropagator List*$(S_0 \cup S_n)$; The *MakePropagatorList* is a procedure used to make a propagator list out of a given propagator set. The procedure can be efficiently implemented by using a hashed list.
   - $\mathrm{OT}[A, K](x) = \mathcal{L}\{\mathit{TopPropagator}[\Phi(x)]\}$: The $\mathcal{L}$ is the label function, and the *TopPropagator* is a function to get the first propagator in the propagator list.
6) End.

E. Computational Complexity

Suppose K is a N-point structuring element, and the first pass of the ROT requires N operations per binary one pixel and one operation per binary zero pixel. In the second pass of the ROT, the *MakePropagatorList(S)* function could be implemented in $O(\gamma)$ operations if the size of propagator set S is γ. In the worst-case scenario, $\gamma = N^2$, i.e., each of its N neighbors propagates N different labels to x. Under this condition, the ROT algorithm requires $N + O(N^2)$ operations per binary one pixel and two operations per binary zero pixel. The operations involved are simple integer additions or comparisons. Our experimental results in Section VII show that the ROT takes on average $14N$ operations per binary one pixel.

VI. Recursive Closing Transform

In this section, we discuss the dual transform of the ROT—the recursive closing transform (RCT). It provides an efficient way to compute the closings of a binary image with any sized structuring element.

A. Definition

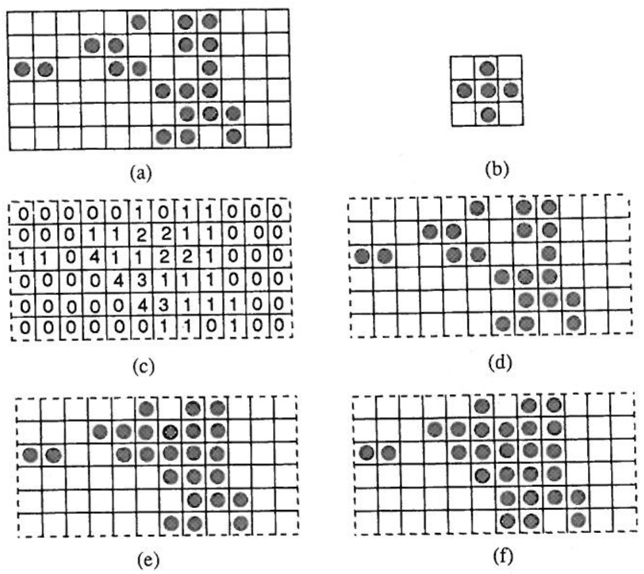
Similar to the DT, which is a dual transform of the ET, the CT is a dual transform of the OT. The closing transform of a set A with respect to a structuring element K produces a gray-scale image where the output at each pixel $x \in Z^2$ is the smallest positive integer n so that $x \in A \bullet (\oplus_{n-1} K)$. If no such n exists, where $x \notin A \bullet (\oplus_{n-1} K)$ for all n, then the closing transform at $x \in Z^2$ is designated as zero.

Definition 18: The *closing transform* of a set $A \subseteq Z^2$ by a structuring element $K \subseteq Z^2$ is denoted by $\mathrm{CT}[A, K]$ and is defined as

$$\mathrm{CT}[A, K](x) = \begin{cases} \min\{n \mid x \in A \bullet (\oplus_{n-1} K)\} & \text{if } \exists n, x \in A \bullet (\oplus_{n-1} K) \\ 0 & \text{if } \forall n, x \notin A \bullet (\oplus_{n-1} K). \end{cases}$$

The above definition indicates that the support domain of $\mathrm{CT}[A, K]$ is Z^2 because closing is extensive. Proposition 11 shows that once the CT has been calculated, it only requires a simple thresholding to compute a binary closing with any sized structuring element. Proposition 12 illustrates that the closing transform, like the OT, does not depend on the origin of the K. The closing transform and its subsequent thresholding processes are illustrated in Fig. 7.

Proposition 11: Let n be a positive integer. If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element, and $B_n = \{x \in Z^2 \mid 0 < \mathrm{CT}[A, K](x) \leq n\}$, then $B_n = A \bullet (\oplus_{n-1} K)$.



0	0	0	0	0	1	0	1	1	0	0	0
0	0	0	1	1	2	2	1	1	0	0	0
1	1	0	4	1	1	2	2	1	0	0	0
0	0	0	0	4	3	1	1	1	0	0	0
0	0	0	0	0	4	3	1	1	1	0	0
0	0	0	0	0	0	1	1	0	1	0	0

Fig. 7. Instance of the closing transform. The dotted boundary indicates that the domain of the image extends to infinity. CT in a window is shown: (a) 2-D binary image; (b) diamond-shaped structuring element; (c) CT of the binary image; (d) thresholded CT image ($n = 1$); (e) thresholded CT image ($n = 2$); (f) thresholded CT image ($n = 3$).

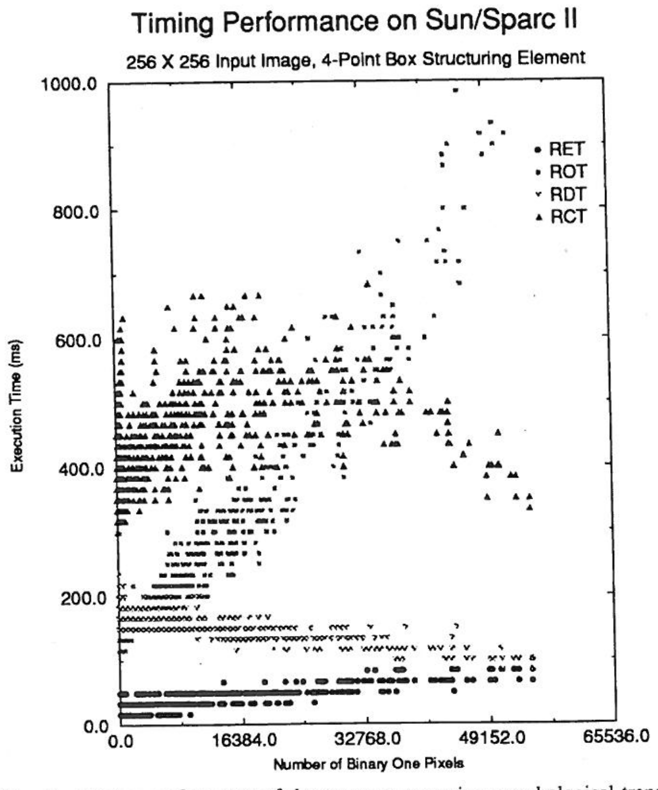


Fig. 8. Timing performance of the two-pass recursive morphological transform algorithm.

Proposition 12: If $A \subseteq Z^2$ is a set, $K \subseteq Z^2$ is a structuring element, and $K' = (K)_t$ is a translation of K by $t \in Z^N$, then $\mathrm{CT}[A, K'](x) = \mathrm{CT}[A, K](x)$, for all x$\in A$.

One way of computing the CT of an image is by the following brute-force algorithm: $A_{n+1} = A \bullet (\oplus_n K), n = 0, 1, 2, \ldots$. The closing transform at $x \in Z^2$ is computed as the minimum n such that $x \in A_n$. If the input image size is $M \times M$ and K is a N-point structuring element, the brute-force algorithm requires $M(MN + 6N + 8)/8$ operations/pixel in the worst-case scenario.

Another way of calculating the CT is by a recursive algorithm based on the duality relation between the CT and the OT. The algorithm first takes the complement of the input image and then computes the ROT on the complemented image with the reflected structuring element. This kind of approach shares the same problems that we have encountered in Section IV. Therefore, a two-pass recursive closing transform algorithm is developed, and it takes about $14N$ operations per pixel on the basis of our experimental results.

B. Recursive Closing Transform Algorithm

The closing transform of a set A with respect to a structuring element K puts in each pixel $x \in Z^2$ the smallest positive integer n so that $x \in A \bullet (\oplus_{n-1} K)$. It is known that the closing operation $A \bullet (\oplus_{n-1} K)$ can be accomplished by the opening operation on the complement set of A, i.e., $A \bullet (\oplus_{n-1} K) = [A^c \circ (\oplus_{n-1} \check{K})]^c$. Based on this duality relation and thresholding property of the closing transform (Proposition 11), we can easily prove Proposition 13, which reveals the relationship between the closing transform and the opening transform.

Proposition 13: Let $n > 1$. If $C_n = \{x \mid \mathrm{CT}[A, K](x) = n\}$ and $O_n = \{x \mid \mathrm{OT}[A^c, \check{K}](x) \geq n\}$, then $C_n = O_{n-1} - O_n$.

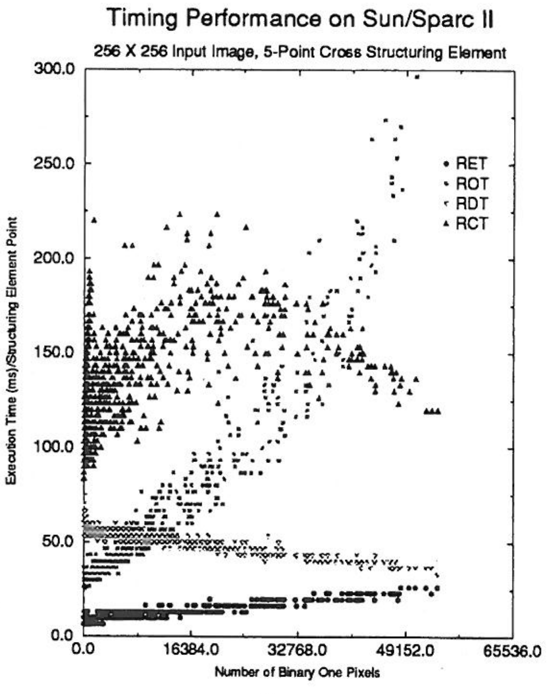
The above relation suggests a procedure to compute the closing transform. Suppose we want to compute $\mathrm{CT}^\rho = \{\mathrm{CT}[A, K](x) \leq \rho + 1 \mid x \in Z^2\}$, where ρ is any chosen nonnegative integer. We denote such closing transform as CT^ρ since we constrain the CT output to be less than or equal to $\rho + 1$. The computation of CT^ρ is quite straightforward if we know how to calculate $\mathrm{OT}[A^c, \check{K}]$. Based on Proposition 13, the two have the following relationship: Assume $x \in A^c$

$$\mathrm{CT}^\rho[A, K](x) = \begin{cases} \mathrm{OT}[A^c, \check{K}] + 1 & \text{if } \mathrm{OT}[A^c, \check{K}] \leq \rho \\ 0 & \text{otherwise.} \end{cases}$$

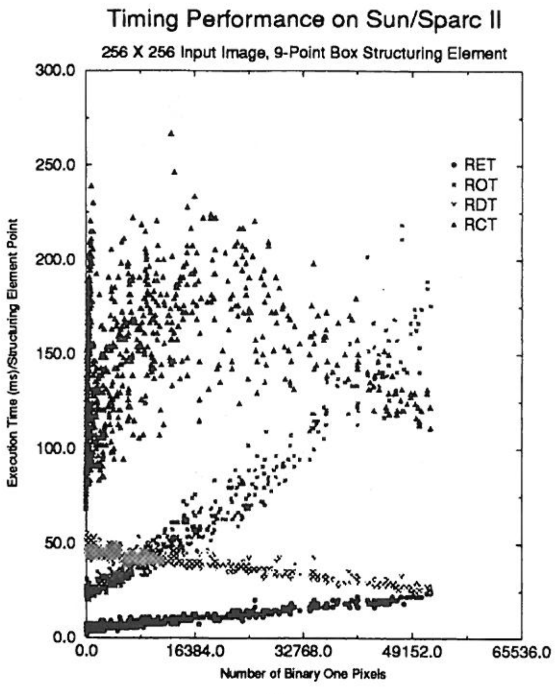
In Section V, we already knew that $\mathrm{OT}[A^c, \check{K}]$ can be calculated by an efficient two-pass ROT algorithm. Therefore, we could compute the $\mathrm{CT}^\rho[A, K]$ via a similar two-pass recursive algorithm. In the first pass, $\mathrm{DT}[A, K]$ is calculated by the recursive DT algorithm described in Section IV. In the second pass, $\mathrm{OT}[A^c, \check{K}]$ is computed by employing the same recursive label propagation process used in the ROT. Since $(\mathrm{DT}[A, K](x) - 1)$ is the number of times one can consecutively dilate x with $\check{K}$ and the dilated pattern is still contained in the A^c, a given pixel x will initially have a propagator $[\mathrm{DT}[A, K](x), \mathrm{DT}[A, K](x) - 1, x]$. The detailed description of the RCT is given in the next section.

C. Algorithm Description

Let $A \subseteq Z^2$ be a set, and let $K \subseteq Z^2$ be a structuring element containing the origin. Let the scanning function S be chosen as the order of left-to-right and top-to-bottom. Accordingly, let the reverse scanning function S^{-1} be the order of right-to-left and bottom-to-top.



(a)



(b)

Fig. 9. Normalized execution time of the two-pass recursive morphological transform algorithm: (a) Five-point cross structuring element; (b) nine-point box structuring element.

This RCT algorithm computes CT^{ρ} for any given ρ. Due to the extensive property of $\mathrm{CT}[A, K]$, if the input is a $M \times M$ binary image, the output CT^{ρ} is a $[M + \rho(S_r - 1)] \times [M + \rho(S_c - 1)]$ image.

Algorithm Recursive Closing Transform—2D Case

1) Set the origin O of K to be the bottom-most right point in K.
2) The first pass of RCT (scanning function S^{-1}): Execute the RDT algorithm described in Section IV to compute $\mathrm{DT}^{\rho+1}$ using the structuring element K. Only the backward pass of RDT is needed.
3) Decompose the K with respect to the scanning function S: Let K^f, K^b be the corresponding forward and backward structuring element. In this particular case, K^b is an empty set ϕ.
4) The second pass of RCT (scanning function S): For each pixel x on the scanning path, do the following: Let $\Phi(x)$ be the propagator list at pixel x if the following hold:
   - If $\mathrm{DT}[A, K](x) = 0$, set $\Phi(x) = \{(0, 0, x)\}$.
   - If $\mathrm{DT}[A, K](x) \neq 0$, perform the following steps: Let S_0, S_n be propagator sets.
     (a) $S_0 = \{(l_0, f_0, x) \mid l_0 = \mathrm{DT}[A, K](x), f_0 = \mathrm{DT}[A, K](x) - 1\}$;
     (b) $S_n = \{(l, f - 1, x) \mid (l, f, x + a) \in \Phi(x + a), a \in K^f \text{ and } f > 1\}$;
     (c) $\Phi(x) = \mathit{MakePropagatorList}(S_0 \cup S_n)$. The *MakePropagatorList* is a procedure used to make a propagator list out of a given propagator set. The procedure can be efficiently implemented by using a hashed list.
   - $\mathrm{CT}[A, K](x) = \mathcal{L}\{\mathit{TopPropagator}[\Phi(x)]\}$. The $\mathcal{L}$ is the label function, and the *TopPropagator* is a function to get the first propagator in the propagator list.
   - If $\mathrm{CT}[A, K](x) > \rho + 1$, set $\mathrm{CT}[A, K](x) = 0$.
5) End.

D. Computational Complexity

Suppose K is a N-point structuring element; the first pass of the RCT requires N operations per binary zero pixel and one operation per binary one pixel. The second pass of the RCT, as analyzed in the ROT, requires $O(N^2)$ operations per binary zero pixel and one operation per binary one pixel. As a summary, the RCT algorithm requires $N + O(N^2)$ operations per binary zero pixel and two operations per binary one pixel. The operations involved are simple integer additions or comparisons. Our experimental results in Section VII show that the ROT takes on average $14N$ operations per binary zero pixel.

VII. Results and Discussions

The correctness of the above RET, RDT, ROT, and RCT algorithms were also confirmed by the experimental results. To do this, an experimental system was set up in such a way that it will be able to generate random test images, run through the recursive morphological transform algorithms, and compare the results with those obtained by the brute-force algorithms. Thousands of randomly generated test images have passed through such verifications.

The timing performance of the set of recursive algorithms were also measured on the Sun/Sparc II workstation. The *C* programs were initially compiled with the optimization flag on. In all the experiments, the ET, the DT$^\rho$, the OT, and the CT$^\rho$ of the input images were computed where $\rho = 32$. The input image sizes were chosen to be 256×256.

Fig. 8 explains the algorithms' execution time as a function of the number of binary one pixels in the input image when the four-point box structuring element is used (origin at the upper-left corner). If we assume that 50% of the input image is covered by the binary one pixels, the timing curve shows that it takes approximately 70 ms to do any sized box erosions and about 500 ms to perform any sized box opening. The curve also indicates that it takes approximately 120 ms to compute any box dilations of size up to $\rho = 32$ and about 500 ms to calculate any box closings of size up to $\rho = 32$.

Fig. 9 summarizes the recursive algorithms' execution time normalized by the number of pixels in the structuring element. Fig. 9(a) is obtained when the five-point cross structuring element is used, and Fig. 9(b) is measured when the nine-point box structuring element is applied. The two set of curves look almost identical. This demonstrates that the execution time of the recursive algorithms are proportional to the number of points in the structuring element.

If we perform linear regressions to the timing curves in Fig. 9, the ratio of the slopes of the two fitted curves of the OT and ET is approximately 14:1. It is known that the ET requires $N+2$ operations per binary one pixel. Thus, the OT will require on average $14N$ operations per binary one pixel. By the same argument, the CT will require on average $14N$ operations per binary zero pixel.

VIII. Conclusion

In this paper, we define the recursive morphological transforms of a binary image, namely, the recursive erosion transform (RET), the recursive dilation transform (RDT), the recursive opening transform (ROT), and the recursive closing transform (RCT). These transforms provide us a new efficient way of computing the binary erosion, dilation, opening, and closing with arbitrary sized structuring element. They have both theoretical and practical significance to real-time vision systems. The set of recursive morphological transforms also offers a solution to some tasks that need to perform a morphological operation but where the size of the structuring element has to be determined after a morphological examination of the content of the image.

References

[1] A. Rosenfeld and J. L. Pfaltz, "Distance functions in digital pictures," *Patt. Recogn.*, vol. 1, 1968, pp. 33–61.
[2] G. Bertrand and X. Wang, "An algorithm for a generalized distance transformation based on Minkowski operations," in *Proc. 9th ICPR* (Rome), Nov. 1988, pp. 1163–1167.
[3] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision.* Reading, MA: Addison-Wesley, 1991.
[4] X. Wang and G. Bertrand, "Some sequential algorithms for a generalized distance transformation based on Minkowski operations," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 14, no. 11, pp. 1114–1121, 1992.
[5] S. Chen, R. M. Haralick, and T. Kanungo, "Recursive opening transform," in *Proc. CVPR* (Champaign), June 1992, pp. 560–565.
[6] J. Serra, *Image Analysis and Mathematical Morphology.* London: Academic, 1982.
[7] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-9, no. 4, pp. 532–550, July 1987.
[8] P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 7, pp. 701–716, July 1989.

Su Chen was born in Fuzhou, China, in 1965. He received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1987 and 1989, respectively.

Since 1990, he has been with the Intelligent Systems Laboratory, University of Washington, where he is a graduate research assistant, working toward the Ph.D. degree in electrical engineering. His Ph.D. dissertation is on the layout analysis of scan-digitized paper documents. His main research interests include computer vision, image processing, graphics, OCR systems, document information processing, management, and retrieval.

Robert M. Haralick (F'84) received the B.A. in mathematics from the University of Kansas in 1964, the B.S. degree in electrical engineering in 1966, and the M.S. degree in electrical engineering in 1967. He received the Ph.D. degree from the University of Kansas in 1969.

He joined the faculty of the Electrical Engineering Department at the University of Kansas, where he last served as Professor from 1975 to 1978. In 1979, he joined the Electrical Engineering Department at Virginia Polytechnic Institute and State University, where he was a Professor and Director of the Spatial Data Analysis Laboratory. From 1984 to 1986, he served as Vice President of Research at Machine Vision International, Ann Arbor, MI. He is the Boeing Clairmont Egtvedt Professor in Electrical Engineering at the University of Washington. His recent work has been in shape analysis and extraction using the techniques of mathematical morphology, robust pose estimation, techniques for making geometric inferences from perspective projection information, propagation of random perturbations through image analysis algorithms, and document analysis.

Dr. Haralick was elected Fellow of IEEE for his contributions to computer vision and image processing. He serves on the Editorial Board of IEEE Transactions on Pattern Analysis and Machine Intelligence, is an associate editor for the IEEE Transactions on Image processing, and is an associate editor for *Pattern Recognition*.