Recursive Binary Dilation and Erosion Using Digital Line Structuring Elements in Arbitrary Orientations

Desikachari Nadadur and Robert M. Haralick, *Fellow, IEEE*

***Abstract*—Performing morphological operations such as dilation and erosion of binary images, using very long line structuring elements is computationally expensive when performed brute-force following definitions. In this paper, we present two-pass algorithms that run at constant time for obtaining binary dilations and erosions with all possible length line structuring elements, simultaneously. The algorithms run at constant time for any orientation of the line structuring element. Another contribution of this paper is the use of the concept of orientation error between a continuous line and its discrete counterpart. The orientation error is used in determining the minimum length of the basic digital line structuring element used in obtaining what we call dilation and erosion transforms. The transforms are then thresholded by the length of the desired structuring element to obtain the dilation and erosion results. The algorithms require only *one* maximum operation for erosion transform and only *one* minimum operation for dilation transform, and *one* thresholding step and *one* translation step per result pixel. We tested the algorithms on Sun Sparc Station 10, on a set of 240×250 salt and pepper noise images with probability of a pixel being a 1-pixel set to 0.25, for orientations of the normals of the structuring elements in the range $[\pi/2, 3\pi/2]$ and lengths, in pixels, in the range $[5, 145]$. We achieved a speed up of about 50 (and for special orientations $\theta \in \{(\pi/2), (3\pi/4), \pi, (5\pi/4), (3\pi/2)\}$ a speed up of about 100) when the structuring elements had lengths of 145 pixels, over the brute-force methods in these experiments. We compared the results of our dilation algorithm with those of the algorithm discussed by Soille *et al.* in [1] and showed that for binary dilation (and erosion since it is just the dilation of the background with the reflected structuring element) our algorithm performed better and achieved a speed up of about four when dilation or erosion transform alone is obtained.**

***Index Terms*—Binary morphology, digital lines, dilation, erosion, line structuring elements, recursivity.**

I. Introduction

MATHEMATICAL morphology is based on *Set* theory. It is an algebraic system that provides two basic operations—*dilation* and *erosion*. Combinations of these operations enable the underlying shapes to be identified and optimally reconstructed from their noisy distorted forms. The theory of mathematical morphology has been developed by many researchers [2]–[14] over the years. Sets in mathematical morphology represent shapes that appear in binary or static gray scale images. Sets in Euclidean 2-space represent foreground regions in binary images. Sets in Euclidean 3-space may represent time-varying images, static gray scale images or binary solids. Moreover, sets in higher dimensional spaces may include additional information such as color or multiple perspective imagery. The two operations, dilation and erosion, are closely related to the Minkowski addition and subtraction respectively in Euclidean space using translation, unions and intersections. A combination of these two basic operations of mathematical morphology gives rise to two additional operations of *opening* and *closing*. Morphological algorithms can be developed that incorporate various compositions of dilations and erosions in order to extract shapes from the imagery. Due to its sound mathematical basis and nonlinear nature mathematical morphology has excelled in the field of image analysis. The paper focusses on designing morphological algorithms that execute faster and those which are suitable for real-time or near real-time applications.

Several specialized image processing architectures have been developed and built [15]–[18] that implement local, e.g., 3×3 neighborhood morphological operations directly. Since the general morphological operations are not restricted to 3×3 structuring elements, researchers and practitioners have tried to achieve the speeding up of the morphological operations using arbitrary sized structuring elements through their decomposition into primitive structuring elements [19]–[25]. Recently, researchers [1], [26]–[40] have designed recursive algorithms, for binary as well as grayscale morphology, that executed faster than the brute-force morphological algorithms based on unions and intersections of the images and the structuring elements.

The work discussed in this paper focuses on the development of recursive algorithms for binary dilation and erosion using digital line structuring elements that run at constant time irrespective of the lengths and orientations of the line structuring elements. Here, we discuss two-pass algorithms that we have developed. In most of the applications the exact orientation of the structuring element may not be very rigid. However, the speed of execution of the algorithm may be of importance. The main difference between our approach and the work previously [1] done on this problem is the use of the concept of *orientation error* between the continuous line structuring element and its discrete counterpart. This user specifiable orientation error is used in determining the minimum length of the basic structuring element used in obtaining the dilation and erosion transforms. These transforms are then thresholded by the length of the actual (desired) structuring element and then translated to account for its true origin. This will be the desired dilation or erosion result. Our algorithm takes only *one* max/min operation per re-

Manuscript received January 24, 1997; revised September 1, 1999. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Scott T. Acton.

D. Nadadur is with the Ultrasound Group, Siemens Medical Systems, Inc., Issaquah, WA 98027-7002 USA (e-mail: dnadadur@sqi.com).

R. M. Haralick is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195-2500 USA (e-mail: haralick@ptah.ee.washington.edu).

Publisher Item Identifier S 1057-7149(00)03568-5.

sult pixel for erosion/dilation respectively, whereas Soille's algorithm takes four min/max operations per result pixel for erosion/dilation, respectively.

1) Motivation: Performing morphological operations such as binary dilations and erosions with long digital line structuring elements using brute-force methods of taking unions and intersections is time consuming since for a domain size $k \times k$ of the structuring elements takes k^2 unions/intersections for each 1-pixel in the image. Wu has designed a morphological algorithm [41] for extracting the epicardial boundary of the left ventricle of the heart from the ultrasound images. One third of the morphological operations are performed using long digital line structuring elements on the binary images obtained after preprocessing the gray scale images. Speeding up of the morphological processing of the binary images drastically improves the execution time of the overall algorithm. This motivated us to design recursive morphological algorithms for binary dilation and erosion discussed in the paper [33], [34]. However, after the publication of that paper, we realized that further optimization for speed could be achieved by designing two pass algorithms for binary dilation and erosion instead of the one pass algorithms discussed in that paper.

We compare our results with those of the conventional methods of taking unions/intersections and with those of the algorithms developed [1] by Soille *et al.* In the following section, we briefly discuss Soille's algorithm for dilation. Then, we discuss our recursive procedures for binary dilation and erosion following a tutorial approach and avoid very formal treatment of the subject. Section III states general definitions and notation that will be used in discussing our recursive morphological algorithm. Section IV discusses the type of digital lines that we use and how to generate them. Sections V and VI discuss our binary recursive dilation and erosion transforms and the recursive dilation and erosion algorithms. Section VII discusses the experimental protocol for comparing our results with brute-force as well as Soille's recursive dilation algorithms, and the complexity and timing results of the algorithms.

II. Soille's Algorithm for Dilation

Soille [1] uses two intermediate buffers g and h of the same size as the input image ($n = \#rows \times \#cols$) where # is a number of operator. The output or result image r is also of the same size. Let the length of the line structuring element be k. A digital line of length $\max(\#rows, \#cols)$ is generated from one of the corners of the image depending on the specified orientation as depicted in Fig. 1 which is reproduced here for the sake of clarity. The digital line is stored in an 1-D array p indexed by rows or columns depending on the orientation. The image is scanned by the digital line in the direction shown in Fig. 1 depending on the orientation. As the image is being scanned, at first the line that falls inside the image plane increases, then remains constant and finally decreases. During the scan dilation is obtained by recursively taking the maximums in the *left* (filling buffer h) and *right* (filling buffer g) directions. Then, finally the result is obtained using h and g buffers. The algorithm is reproduced as follows.

For any position x along the line, perform the operations in the equations shown at the bottom of the page, where

$$gh(x) = \begin{cases} g(x+k/2), & \text{if } x = 0, \ldots, n-k/2-1 \\ g(n-1), & \text{if } x = n-k/2, \ldots, k/2 \\ h(x-k/2), & \text{if } x = k/2+1, \ldots, n-1 \end{cases}$$

$$h2(x) = \begin{cases} f(p(x)), & \text{if } x = n-1 \\ \max[h2(x+1), f(p(x))], & \text{otherwise.} \end{cases}$$

In the following section, we discuss the definitions and notations that will be used in the rest of the paper.

III. Definitions and Notation

This section provides some background in Mathematical Morphology using set theoretic notation [2], [4]. Let $Z = \{z \,|\, 0 \le z < \infty\}$ be the set of integers. Let A, B, C, and K be sets in Z^2 and let the O be the origin of Z^2, i.e., $O \in Z^2$.

A. Review of Mathematical Morphology

Definition III.1: The *Translation* of the set A to the point $t \in Z^2$ is defined as $A_t = \{x \,|\, x = a + t, a \in A\}$.

Definition III.2: The *Reflection* of the set K is denoted by $\check{K}$ and is defined by, $\check{K} = \{-x \,|\, x \in K\}$.

Definition III.3: Binary Dilation of a set A by a structuring element K is denoted by $A \oplus K$ and is defined as $A \oplus K = \{x \in Z^2 \,|\, x = a + b,$ for some $a \in A$ and $b \in K\}$.

Geometrically, the dilation can be interpreted as the translation of A by all the points in K and then taking the union, i.e., $A \oplus K = \bigcup \{A_b \,|\, b \in K\}$.

$$g(x) = \begin{cases} f(p(x)), & \text{if } x = 0, k, \ldots, (m-1)k, \\ & mk, \quad \text{and} \quad x < n-1 \\ \max[g(x-1), f(p(x))], & \text{otherwise} \end{cases}$$

$$h(x) = \begin{cases} f(p(x)), & \text{if } x = n-1, mk-1, \\ & (m-1)k-1, \ldots, k-1 \\ \max[h(x+1), f(p(x))], & \text{otherwise} \end{cases}$$

$$r(p(x)) = \begin{cases} g(n-1), & \text{if } n \le k/2 \\ gh(x), & \text{if } k/2 < n \le k \\ g(x+k/2), & \text{if } x = 0, \ldots, k/2-1 \\ h2(x-k/2), & \text{if } x = n-1, \ldots, n-k/2 \\ \max[g(x+k/2), h(x-k/2)], & \text{otherwise} \end{cases}$$

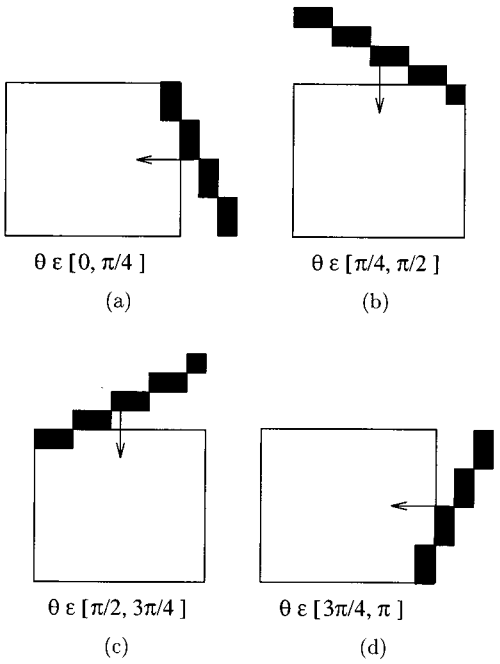

Fig. 1. Digital lines. The arrow indicates the direction of translation of the line over the image.

Definition III.4: Binary Erosion of a set A by a structuring element K is denoted by $A \ominus K$ and is defined as $A \ominus K = \{x \in Z^2 \,|\, x + b \in A \text{ for every } b \in K\}$.

Geometrically, the erosion can be interpreted as the translation of A by all the points in $\check{K}$ and then taking the intersection, i.e., $A \ominus K = \bigcap\{A_b \,|\, b \in \check{K}\}$.

B. Review of Recursive Operators

Recursive operations [29] on binary images are accomplished using a particular scan order of pixels in the image. This scan order can be specified using scanning functions.

Definition III.5: A *scanning function* S is defined as a one-to-one mapping from a finite set $I = \{(x_1, x_2) \in Z^2 \,|\, 0 \leq x_1 < n_1, 0 \leq x_2 < n_2\}$ to the set $\{1, 2, \ldots, n_1 n_2\}$. If $p \in I, q \in I$ and $S(q) < S(p)$, then the output value at q is computed before the output value at p. Fig. 2 illustrates few scanning functions.

Definition III.6: A *recursive operator* on a binary image is an operator whose output depends not only on the input pixels in the domain of its kernel, but also on the values of the previously computed pixels, with respect to a given scanning function S. If $X \in Z^2$ is the input binary image and Y is the output of the recursive operator, then output $Y(P)$ at the pixel P with respect to the given scanning function S is $Y(P) = f(X(1), X(2), \ldots, X(P), Y(1), Y(2), \ldots, Y(P-1))$. Fig. 3 illustrates a recursive operator.

In the next section, we discuss the concept of digital lines, orientation error between a continuous line and its digital counterpart, and motivate the need for the concept of orientation error.

IV. Digital Lines

This section discusses definitions and propositions related to the continuous lines and their discrete counterparts what we call digital lines. These digital lines are only approximations to the continuous lines [42], [43].

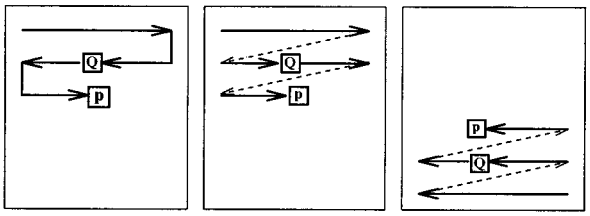

Fig. 2. Examples of scanning functions.

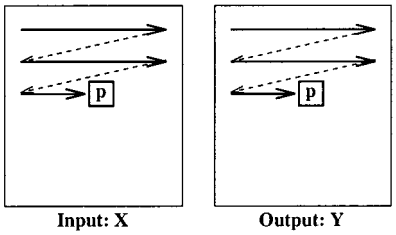

Fig. 3. Example of a recursive operator.

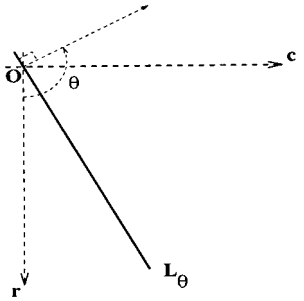

Fig. 4. A continuous line.

Definition IV.1: A *line* in the continuous domain, whose normal is oriented at angle θ measured counter-clockwise from the row axis or r-*axis* in the row-column or r-c coordinate system and passing through the origin $O \in \Re^2$ is denoted by L_θ and is defined as $L_\theta = \{(x, y) \in \Re^2 \,|\, x \cos\theta + y \sin\theta = 0\}$. Fig. 4 illustrates a continuous line.

From now, any reference to "line oriented at angle θ" refers to a "line whose normal is oriented at angle θ". The following defines a digital line closest to the continuous line oriented at angle θ.

Definition IV.2: A digital line closest to the continuous line oriented at some angle θ is denoted by $D_\theta \subseteq Z^2$, and is defined as $D_\theta = \{(r, c) \in Z^2 \mid \text{for some } \theta, |x \cos\theta + y \sin\theta| \leq (1/2)\}$. Fig. 5 illustrates a digital line closest to the given continuous line.

The above definition can be implemented using the discretization operator $\mathcal{D}$ as in the following definition.

Definition IV.3: Let $A = \{(x, y) \in \Re^2\}$ be a given set in $\Re^2$. The discretization operator $\mathcal{D} : \Re^2 \mapsto Z^2$ is defined as $\mathcal{D}(A) = \{(r, c) \in Z^2 \,|\, (x, y) \in A, r = \text{round}(x) \text{ and } c = \text{round}(y)\}$ where round( ) is the operation of rounding off to the nearest integer.

If we replace $A \subseteq \Re^2$ in the above definition by the line $L_\theta \in \Re^2$, the discretization operator $\mathcal{D}$ can be used to obtain

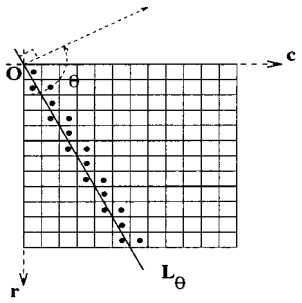

Fig. 5. A digital line closest to the given continuous line.

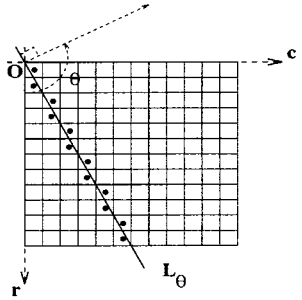

Fig. 6. An example of the digital line generated by an algorithm of the type of Bresenham algorithm, closest to the given continuous line.

the digital line D_θ. However, the digital line thus created (see Fig. 5) gives rise to overlapping patterns when it is used to scan the image during dilation and erosion. Therefore, in our implementation we avoid such patterns. This can be achieved by using algorithms such as Bresenham [42] which by their nature avoids patterns such as shown in Fig. 5. An example of the types of digital lines that are generated (similar to Bresenham) closest to the given continuous lines and the types that we use in our algorithms is given in Fig. 6.

From now, when we say *digital line*, we mean that it is of the type of Bresenham line. The above definitions do not include the length of the digital lines. The following defines a digital line of particular length, say, n pixels.

Definition IV.4: Let n be a positive integer $(n > 1)$. Then a digital line $D_{n,\theta} \subseteq Z_n^2$ of length n pixels closest to the given continuous line $L_\theta \subseteq \Re^2$ oriented at an angle θ, where $Z_n^2 = \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}$ is given by $D_{n,\theta} = \{(r, c) \in Z_n^2 \,|\, (r, c) \in D_\theta\}$.

The orientation or slope of the digital line is different from that of the corresponding continuous line. The following definition states the concept of orientation error or the angular distance between two orientations.

Definition IV.5: Angular distance or angular separation between two orientations $\theta_1, \theta_2 \in [(\pi/2), (3\pi/2)]$, denoted by $d_\theta(\theta_1, \theta_2)$, is defined as $d_\theta(\theta_1, \theta_2) = \min(|\theta_1 - \theta_2|, |\pi + \theta_1 - \theta_2|)$. This concept is illustrated in Fig. 7.

The following defines the angular orientation of the digital line $D_{n,\theta} \subseteq Z_n^2$ that is closest to and obtained by the discretization of the continuous line $L_\theta \subseteq \Re^2$ oriented at a counter-clockwise angle θ from the r-axis and passing through the origin.

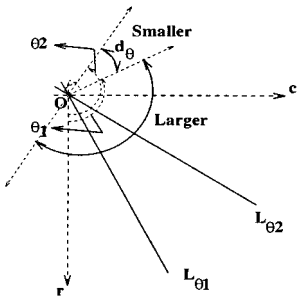

Fig. 7. Angular distance or angular separation between two orientations $\theta_1, \theta_2 \in [(\pi/2), (3\pi/2)]$.

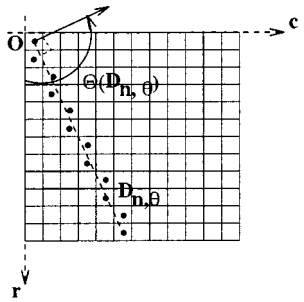

Fig. 8. Angular orientation of the digital line $D_{n,\theta}$.

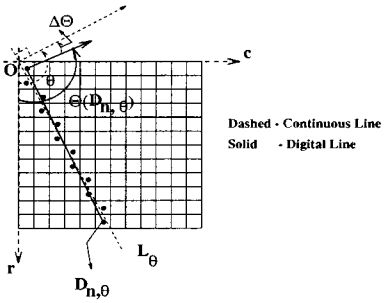

Fig. 9. Orientation error between the continuous line L_θ and its length-n discretization $D(n, \theta)$.

Definition IV.6: The angular orientation of a digital line $D_{n,\theta} \subseteq Z_n^2$ is denoted by $\Theta(D_{n,\theta})$ and is defined as

$$\Theta(D_{n,\theta}) = \frac{\pi}{2} + \arctan\left\{\frac{\max_{(r,c)\in D_{n,\theta}} c}{\max_{(r,c)\in D_{n,\theta}} r}\right\}.$$

This is illustrated in Fig. 8.

The following defines the orientation error between the continuous line $L_\theta \subseteq \Re^2$ oriented at an angle θ and its length-n discretization $D_{n,\theta} \subseteq Z^2$ oriented at angle $\Theta(D_{n,\theta})$ with the aid of Definition IV.5.

Definition IV.7: The Orientation Error between the continuous line $L_\theta \subseteq \Re^2$ oriented at angle θ and passing through the origin and its corresponding length-n digital line $D_{n,\theta}$ oriented at angle $\Theta(D_{n,\theta})$ is denoted by $\Delta\Theta(L_\theta, n)$ and is defined as $\Delta\Theta(L_\theta, n) = d_\theta(\Theta(D_{n,\theta}), \theta)$. Fig. 9 illustrates this concept.

As the length n of the digital line increases the number of possible directions in a square lattice increases. For an odd n, there are a total of $(2n - 2)$ possible orientations. Therefore,

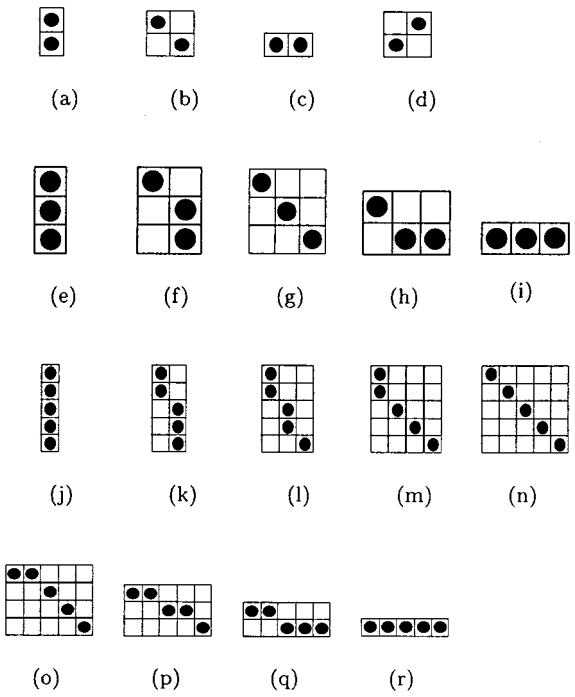

Fig. 10. (a)–(r) Show the increase in the number of angular orientations of the digital line with an increase in its length (or the size of its rectangular domain).

as n increases the orientation error between the digital line of length n and the corresponding continuous line L_θ decreases. A few examples of this concept are illustrated in the Fig. 10. The following definition formally states the same concept.

Definition IV.8: The Orientation error $\Delta\Theta(L_\theta, n)$ between the continuous line L_θ and its length-n discretization $D_{n,\theta}$ decreases as n increases, that is, $\Delta\Theta(L_\theta, n) > \Delta\Theta(L_\theta, n+1)$ where $n > 1$.

A. Motivation for the Use of Orientation Error

If the normal to the continuous line is oriented at an angle, $\theta \in \{(\pi/2), (3\pi/4), \pi, (5\pi/4), (3\pi/2)\}$, the continuous line can be represented exactly in digital domain using only 2 pixels. However, if the normal to the continuous line is oriented at some arbitrary counter-clockwise angle, say, 35.36° from r-axis, it will require more than 200 pixels to represent the line, very accurately, in the digital domain. This by no means is an advantage to us in obtaining the transforms, if the length of the desired structuring element is of length, say, 100. Also, the length of the structuring element is bounded by the maximum dimension of the image. Therefore, for smaller sized images we may not be able to accurately specify the structuring element. Therefore, we require that the *orientation error* between the continuous line and its discrete counterpart be a control parameter in the design of the recursive dilation and erosion discussed in this paper. As stated previously in Definition IV.8, the length of the digital line structuring element increases as the angular error decreases. Also, in most of the applications the constraints on the orientation of the digital line structuring elements may not be rigid; the speed of execution of the algorithm may be crucial, instead. The user should be able to specify the maximum limit on the error in the orientation of the continuous line and its length-n discretization. Using this user specified information it is then possible to determine the value of n that results in this error. This will be the minimum length of the digital line structuring element for the given orientation error. Thus, *orientation error* is a crucial concept in the design of the recursive dilation and erosion algorithms discussed in this paper. The following definition formalizes this concept.

Definition IV.9: If $\Delta\theta$ is the user specified maximum limit (tolerance) in the orientation error between $L_\theta \in \Re^2$ and its length-n discretization $D_{n,\theta}$, then, the smallest n that results in $\Delta\theta$ is obtained as $\Delta\Theta(L_\theta, n) \leq \Delta\Theta$.

In practice, we create a look-up table relating n to the maximum (worst-case) orientation error between the continuous line and the digital line of length n, in an off-line procedure using

$$\Delta\Theta(n; n = 1, \ldots, N) = \begin{cases} \max_{\frac{\pi}{2} \leq \theta \leq \frac{3\pi}{4}} \min_{\substack{j=n \\ i=0,\ldots,j}} \left|\theta - \frac{\pi}{2} - \arctan\left(\frac{i}{j}\right)\right| \\ \max_{\frac{3\pi}{4} \leq \theta \leq \pi} \min_{\substack{i=n \\ j=0,\ldots,i}} \left|\theta - \frac{\pi}{2} - \arctan\left(\frac{i}{j}\right)\right| \\ \max_{\pi \leq \theta \leq \frac{5\pi}{4}} \min_{\substack{i=n \\ j=0,\ldots,i}} \left|\theta - \frac{\pi}{2} + \arctan\left(\frac{i}{j}\right)\right| \\ \max_{\frac{5\pi}{4} \leq \theta \leq \frac{3\pi}{2}} \min_{\substack{j=n \\ i=0,\ldots,j}} \left|\theta - \frac{\pi}{2} + \arctan\left(\frac{i}{j}\right)\right| \end{cases} \quad (1)$$

In constructing the above look-up table, when $j = 0$, we take $\arctan(i/j) = (\pi/2)$.

This look-up table is stored in a one-dimensional (1-D) array indexed by the length, in pixels, of the digital line. The value stored at each location in the array indexed by the length of the digital line is the worst-case orientation error for that length.

In the following section, we describe the way of generating the basic digital line structuring element used in obtaining the dilation transform of the given binary image.

B. Generation of Basic Digital Line Structuring Elements

In this section, we discuss the way of generating the basic digital line structuring element of minimum length using the user specified allowable orientation error. This basic structuring element will be used in obtaining the dilation and erosion transforms of the binary image. The transforms will be thresholded by the actual length of the structuring element to obtain the desired dilation or erosion.

When the user specifies worst-case orientation error that he/she can tolerate, that value is compared to the values in the look-up table. Then, the value from the look-up table corresponding to the closest orientation error value is chosen to be the length, in pixels, of the digital line structuring element to be used as the basic structuring element for obtaining recursive dilation and erosion transforms. The user will also specify the angular orientation of the continuous line structuring element L_θ. This orientation θ and the length n of the digital line structuring element from the look-up table are used in constructing the digital line structuring element.

The digital line structuring element is stored in a 1-D array indexed by the rows or columns of its kernel depending on θ. The values stored in the array are the offsets of the pixels in the digital line with the starting pixel on the line as reference.

The following proposes a way of determining the number of rows and columns in the kernel of the basic digital line structuring element.

Proposition IV.1: If θ is the angular orientation of the continuous line structuring element L_θ, n the length, in pixels, of the basic digital line structuring element (obtained using the Definition IV.9) corresponding to the user specified orientation error tolerance $\Delta\Theta$, then the number of rows and columns in the kernel of the basic digital line structuring element are defined as

$$\left.\begin{array}{l}\#r_n = n \\ \#c_n = \text{round}\left((\#r_n)\tan(\theta - \frac{\pi}{2})\right)\end{array}\right\}, \quad \text{if } \begin{cases}\frac{\pi}{2} < \theta < \frac{3\pi}{4} \\ \text{and} \\ \frac{5\pi}{4} < \theta < \frac{3\pi}{2}\end{cases}$$

and

$$\left.\begin{array}{l}\#c_n = n \\ \#r_n = \text{round}\left(\frac{\#c_n}{\tan(\theta - \frac{\pi}{2})}\right)\end{array}\right\}, \quad \text{if } \begin{cases}\frac{3\pi}{4} < \theta < \pi \\ \text{and} \\ \pi < \theta < \frac{5\pi}{4}\end{cases}$$

where # is a number of operator.

Notice that in the above proposition, angles $(\pi/2), (3\pi/4), \pi, (5\pi/4)$ and $(3\pi/2)$ are not included. This is because for these orientations the basic digital line structuring element is of length, only $n = 2$ pixels. Notice that in this case $\Delta\Theta = 0$.

After the number of rows and columns in the basic structuring element are determined, the digital line is then drawn and the offsets of the pixels from the starting pixel on the line are determined and stored in a 1-D array. The following proposition conveys a way of computing these offsets.

Proposition IV.2: Let $\#r_n$ and $\#c_n$ be the total number of rows and columns in the kernel of the digital line structuring element. Also, $n = \max(\#r_n, \#c_n)$. Let $s(i); i = 0, \ldots, (n-1)$ be the 1-D array in which the pixel offsets are stored. Let θ be the angular orientation of the continuous line L_θ. Then, the offsets are computed as

$$s(r; r = 0, 1, \ldots, \#r_n - 1) = \text{round}\left((r)\tan\left(\theta - \frac{\pi}{2}\right)\right), \quad \text{if } \begin{cases}\frac{\pi}{2} < \theta < \frac{3\pi}{4} \\ \frac{5\pi}{4} < \theta < \frac{3\pi}{2}\end{cases}$$

$$s(c; c = 0, 1, \ldots, \#c_n - 1) = \text{round}\left(\frac{c}{\tan(\theta - \frac{\pi}{2})}\right), \quad \text{if } \begin{cases}\frac{3\pi}{4} < \theta < \pi \\ \pi < \theta < \frac{5\pi}{4}\end{cases}$$

The following section describes the recursive dilation transform algorithm.

V. Recursive Dilation Transform and Recursive Dilation

The *dilation transform* of a binary image is based on the successive morphological dilations of the input image with the basic structuring element. If the set $I \subseteq Z^2$ is the binary image and the set $A \subseteq I$ is the foreground or the 1-pixels in I, then the dilation transform with respect to the basic digital line structuring element $D_{n,\theta} \subseteq Z^2$ is a gray scale image where the gray level of each pixel $x \in I$ is a generalized distance of x to the set A, i.e., the generalized distance of x to the foreground pixels according to the given scanning functions $S(I)$ over the image. That is, they depict the smallest positive integers n, such that $x \in A \oplus D_{n,\theta}$. If no such n exists, where $x \notin A \oplus D_{n,\theta}$ for all n, then the dilation transform at $x \in I$ is designated zero. The following definition states the above concept formally.

Definition V.1: Let $I \subseteq Z^2$ be the binary image and $A \subseteq I$ be the set of foreground or 1-pixels in I. Let $D_{n,\theta} \subseteq Z^2$ be the basic digital line structuring element containing the origin $O \in D_{n,\theta}$ at one end. The dilation transform of the set A is then denoted by

$$F_d(A, D_{n,\theta})(x) = \begin{cases}\min\{n \mid x \in A \oplus D_{n,\theta}\}, & \text{if } \exists n, \quad x \in A \oplus D_{n,\theta} \\ 0, & \text{if } \forall n, \quad x \notin A \oplus D_{n,\theta}\end{cases} \tag{2}$$

A. Dilation Transform Algorithm

In this section we describe the two-pass algorithm for obtaining the recursive dilation transform of a binary image with the basic digital line structuring element generated in Section IV-B.

The recursive dilation transform algorithm comprises of two passes. In the first pass, a *two point* transform is obtained by considering only the end points or end pixels of the n pixel long digital line structuring element. In the second pass an $(n-2)$ *point* transform is generated by filling in the rest of the pixels. The following defines the scanning function used to scan the image, i.e., it defines the order in which the input binary image is scanned by the digital line structuring element.

Definition V.2: Let $I \subseteq Z^2$ be the given binary image and θ be the orientation of the continuous line L_θ. Then the order in which the image is scanned by the digital line structuring element in defined by the scanning function $S(I)$ over image as

$$S(I) = \begin{cases}\text{Left-Right-Top-Down Scan}, & \text{if } \frac{\pi}{2} \le \theta \le \pi \\ \text{Left-Right-Bottom-Up Scan}, & \text{if } \pi < \theta \le \frac{3\pi}{2}\end{cases}$$

Algorithm V.1: Let $s(i); i = 0, \ldots, (n-1)$ be the array of pixel offsets of the digital line, of length n pixels, from its starting pixel (corresponds to index 0 which is also its origin). Let $I \subseteq Z^2$ be the input binary image for which the dilation transform image is to be obtained, $F_d \subseteq \Re^2$ be the corresponding dilation transform image. Let $S(I)$ be the scanning function over the image as in definition V.2. Let $x = (r, c)$ denote an arbitrary pixel on the image. Then

1) **Initialization**

$$F_d(r, c) = \begin{cases}0, & \text{if } I(r, c) = 1 \\ \infty, & \text{if } I(r, c) = 0\end{cases}$$

2) **2-Point Transform along the line from the pixel** (r, c)

$$F_d(r + s(n-1), c + n - 1) = F_d(r, c) + n - 1 \quad \text{if } \begin{cases}\frac{\pi}{2} < \theta < \frac{3\pi}{4} \quad \text{or} \quad \frac{5\pi}{4} < \theta < \frac{3\pi}{2} \quad \text{and} \\ F_d(r, c) \neq \infty \quad \text{and} \\ F_d(r + s(n-1), c + n - 1) \neq 0\end{cases}$$

$$F_d(r + n - 1, c + s(n-1)) = F_d(r, c) + n - 1 \quad \text{if } \begin{cases}\frac{3\pi}{4} < \theta < \frac{5\pi}{4} \quad \text{and} \\ \theta \neq \pi \quad \text{and} \\ F_d(r, c) \neq \infty \quad \text{and} \\ F_d(r + n - 1, c + s(n-1)) \neq 0\end{cases}$$

3) **$(n-2)$-Point Transform along the line from the pixel (r,c)**

$$F_d(r+s(i),c+i)=\min\{F_d(r+s(i),c+i),F_d(r,c)+i\}$$
$$\text{if } \begin{cases} \lfloor F_d(r,c)/(n-1)\rfloor *(n-1)=F_d(r,c) & \text{and} \\ \frac{\pi}{2}<\theta<\frac{3\pi}{4} & \text{or} \\ \frac{5\pi}{4}<\theta<\frac{3\pi}{2} \end{cases}$$

$$F_d(r+i,c+s(i))=\min\{F_d(r+i,c+s(i),F_d(r,c)+i\}$$
$$\text{if } \begin{cases} \lfloor F_d(r,c)/(n-1)\rfloor *(n-1)=F_d(r,c) & \text{and} \\ \frac{3\pi}{4}<\theta<\frac{5\pi}{4} & \text{and} \\ \theta\neq\pi \end{cases}$$

where $i=1,\ldots,(n-2)$ and $\lfloor\bullet\rfloor$ denotes an integer division.

Since the length n of the structuring element is only 2 for the cases when θ is equal to $(\pi/2)$, $(3\pi/4)$, π, $(5\pi/4)$, $(3\pi/2)$ we need to carry out only the first two steps—Initialization and the 2-Point transform. This explains the shorter execution time of the algorithm for these orientations compared to other orientations (see Section VII).

B. Recursive Dilation

The following definition states that the dilation of the set A by a digital line structuring element of length m can be obtained by thresholding, by m, the dilation transform of A obtained using the basic structuring element $D_{n,\theta}$.

Assertion V.1: Let m be a positive integer. If $A\subseteq Z^2$ is a set and $D_{n,\theta}$ be the basic digital line structuring element containing the origin $O\in D_{n,\theta}$ at one end, $F_d(A,D_{n,\theta})$ is the dilation transform image and $B_m=\{x\in Z^2\,|\,0\le F_d(A,D_{n,\theta})(x)\le m-1\}$, then $A\oplus D_{m,\theta}=B_m$.

The following definition states that if the origin of the actual structuring element of length m is different from that of the basic structuring element of the length n used to obtain the dilation transform and the dilation B_m, then the final dilation can be obtained by a simple translation of B_m.

Assertion V.2: Let $t\in Z^2$ be the required translation. Then, the final dilation result, B is obtained by translating the image B_m by t as $B=\{x+t\,|\,x\in B_m \text{ and } t\in z^2\}$.

Fig. 11(a)–(h) illustrate all the steps discussed in Algorithm V.1 for obtaining the dilation transform and Assertions V.1 and V.2 for obtaining the final dilation through thresholding and translation of the dilation transform image.

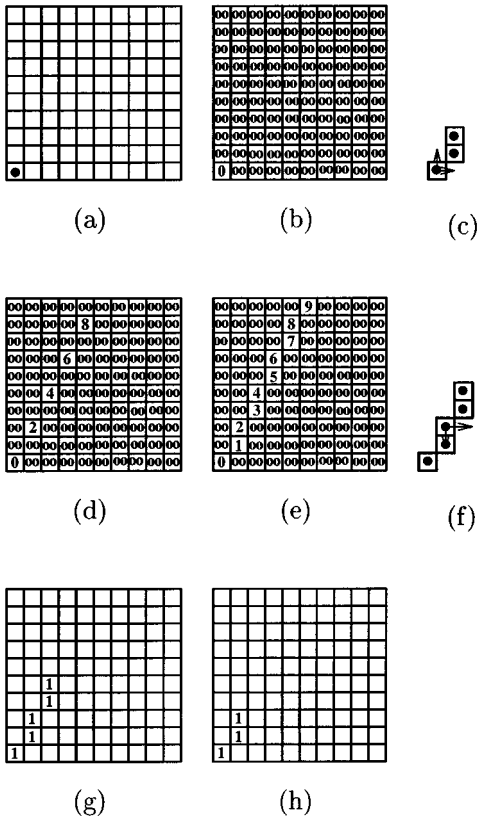

Fig. 11. Illustration of recursive dilation algorithm. (a) Input binary image, (b) initialization, (c) basic SE of length $n=3$, (d) 2-point transform, (e)$(n-2)$ point transform, (f) actual structuring element of length $m=5$, (g) transform thresholded by $m=5$, and (h) translate the thresholded image to account for the true origin. This is the final dilation result.

VI. Recursive Erosion Transform and Recursive Erosion

The *recursive erosion transform* is based on the successive morphological erosions of the binary image. It is a generalization of the distance transform commonly known in the literature [31].

Given a binary image $I\subseteq Z^2$ and the set $A\subseteq I$ of all the one or the foreground pixels, the erosion transform of A with respect to the digital line structuring element $D_{n,\theta}\subseteq Z^2$ is 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 $x\in A\ominus D_{n,\theta}$. The generalized distance at a pixel x indicates the maximum number of consecutive erosions of A by $D_{n,\theta}$ such that x is still contained in the eroded image foreground. The support for erosion is the foreground set A. This is, in other words, saying that the dilation of the background by a structuring element is the same as the erosion of the foreground by the reflected structuring element. The following definition formally states the above concept.

Definition VI.1: Let $I\subseteq Z^2$ be a binary image and $A\subseteq I$ be the set of foreground or one pixels in I. Let $D_{n,\theta}\subset Z^2$ be the digital line containing the origin $O\in D_{n,\theta}$. The erosion transform of the set A with respect to $D_{n,\theta}$ is denoted by $F_e(A,D_{n,\theta})$ and is defined as

$$F_e(A,D_{n,\theta})(x)=\begin{cases}\max\{n\,|\,x\in A\ominus D_{n,\theta}\}, & \text{if } x\in A\\ 0, & \text{if } x\notin A\end{cases}$$

A. Erosion Transform Algorithm

Since we have already established the recursive dilation in a rigorous manner, we believe that such rigorous treatment is not necessary in the recursive erosion transform. We simply note the changes that are required in the structuring element offset array and the scanning function over the image and proceed to describe the algorithm. The required final translation of the image after the thresholding step is same as that we used for dilation.

The changes required in the structuring element and scanning function are as follows.

- Use $-s(i)$ as the array of offsets, instead of $s(i)$.
- The scanning function $S(I)$ is

$$S(I) = \begin{cases} \text{Right-Left-Bottom-Up Scan,} & \text{if } \frac{\pi}{2} \le \theta \le \pi \\ \text{Right-Left-Top-Down Scan,} & \text{if } \pi < \theta \le \frac{3\pi}{2} \end{cases}$$

Algorithm VI.1: Let $-s(i); i = 0, \ldots, (n-1)$ be the array of offsets in the reflected digital line, of length n pixels, from its starting pixel (corresponds to index 0 which is also its origin). Let $S(I)$ be the scanning function over the image as in (3). Let $I \subseteq Z^2$ be the input binary image for which the erosion transform image is to be computed, $F_e \subseteq \Re^2$ be the corresponding erosion transform image. Let $x = (r, c)$ denote an arbitrary pixel on the image. Then

1) **Initialization**

$$F_e(r,c) = \begin{cases} 0, & \text{if } I(r,c) = 1 \\ -\infty, & \text{if } I(r,c) = 0 \end{cases}$$

2) **2-Point Transform along the line from the pixel** (r, c)

$$F_e(r - s(n-1), c - n + 1) = F_e(r,c) + n - 1$$
$$\text{if } \begin{cases} \frac{\pi}{2} < \theta < \frac{3\pi}{4} \quad \text{or} \quad \frac{5\pi}{4} < \theta < \frac{3\pi}{2} \\ F_e(r,c) \neq -\infty \quad \text{and} \\ F_e(r - s(n-1), c - n + 1) \neq -\infty \end{cases}$$
$$F_e(r - n + 1, c - s(n-1)) = F_e(r,c) + n - 1$$
$$\text{if } \begin{cases} \frac{3\pi}{4} < \theta < \frac{5\pi}{4} \quad \text{and} \\ \theta \neq \pi \\ F_e(r,c) \neq -\infty \quad \text{and} \\ F_e(r - n + 1, c - s(n-1)) \neq -\infty \end{cases}$$

3) **$(n-2)$-Point Transform along the line from the pixel** (r, c)

$$F_e(r,c) = \infty$$
$$\text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \quad \text{and} \\ F_e(r - s(1), c - 1) = -\infty \end{cases}$$

$$F_e(r - s(i), c - i)$$
$$= \begin{cases} \infty \quad \text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \\ \text{and} \\ F_e(r - s(i+1), c - i - 1) = -\infty \\ \text{and} \\ \frac{\pi}{2} < \theta < \frac{3\pi}{4} \quad \text{or} \quad \frac{5\pi}{4} < \theta < \frac{3\pi}{2} \end{cases} \\ \max\{F_e(r - s(i), c - i), F_e(r,c) + i\} \\ \quad \text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \\ \text{and} \\ F_e(r - s(i+1), c - i - 1) \neq -\infty \\ \text{and} \\ \frac{\pi}{2} < \theta < \frac{3\pi}{4} \quad \text{or} \quad \frac{5\pi}{4} < \theta < \frac{3\pi}{2} \end{cases} \end{cases}$$

$$F_e(r,c) = \infty$$
$$\text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \quad \text{and} \\ F_e(r - 1, c - s(1)) = -\infty \end{cases}$$

$$F_e(r - i, c - s(i))$$
$$= \begin{cases} \infty \quad \text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \\ \text{and} \\ F_e(r - i - 1, c - s(i+1)) = -\infty \quad \text{and} \\ \frac{3\pi}{4} < \theta < \frac{5\pi}{4} \quad \text{and} \quad \theta \neq \pi \end{cases} \\ \max\{F_e(r - i, c - s(i)), F_e(r,c) + i\} \\ \quad \text{if } \begin{cases} \lfloor F_e(r,c)/(n-1) \rfloor * (n-1) = F_e(r,c) \\ \text{and} \\ F_e(r - i - 1, c - s(i+1)) \neq -\infty \quad \text{and} \\ \frac{3\pi}{4} < \theta < \frac{5\pi}{4} \quad \text{and} \quad \theta \neq \pi \end{cases} \end{cases}$$

where $i = 1, \ldots, (n-2)$ and $\lfloor \bullet \rfloor$ denotes an integer division.

Special cases in the orientation of the digital lines discussed for recursive dilation apply to recursive erosion, also.

B. Recursive Erosion

The Recursive erosion result is then obtained by thresholding and then translating the recursive erosion transform according to Assertions V.1 and V.2. Fig. 12(i)–(p) illustrate all the steps discussed in the Algorithm VI.1 for obtaining the erosion transform and Assertions V.1 and V.2 for obtaining the final erosion through thresholding and translation of the erosion transform image.

VII. Performance Evaluation

In this section we briefly discuss the experimental protocol that we followed in evaluating the execution times of our recursive algorithms against those of the raw or conventional algorithms of taking unions/intersections as well as Soille's algorithm. We then present the timing results.

1) Experimental Protocol: We used salt and pepper noise image of size 240×256 with the probability of a pixel becoming a 1-pixel set to 0.25. Then our recursive algorithms as well as the conventional algorithms were run on this image and the execution times are noted. We also compared our recursive dilation algorithm with that discussed in [1]. Due to the nonavailability of the software from the authors of that paper, we reimplemented the dilation algorithm discussed in that paper. All the software was written in General Image Processing System (GIPSY) environment. The algorithms were run under GIPSY environment on SUN SPARCstation 10 and the execution times were noted. The length of the structuring element is varied in the range $[5, 145]$ pixels and its orientation in the range $[0°, 180°]$. The user specifiable tolerance in orientation error is set at 2° in these experiments. Then the graphs are plotted with the lengths of the structuring elements, in pixels, on the *X-axis* and the elapsed CPU time in seconds on the *Y-axis*.

2) Results and Discussion: The comparison of execution times of the our recursive dilation, conventional, and Soille's algorithms is given in Fig. 13. Fig. 14 shows similar results for the recursive erosion and conventional algorithms.

The results show that we achieved a speed up of about 50 over the conventional algorithms when the actual structuring element is 145 pixels long and $\theta \notin \{(\pi/2), (3\pi/4), \pi, (5\pi/4), (3\pi/2)\}$, under the conditions described in Section VII. Under these condition the recursive algorithm ran at a constant time of about 200 ms. When $\theta \in \{(\pi/2), (3\pi/4), \pi, (5\pi/4), (3\pi/2)\}$, we

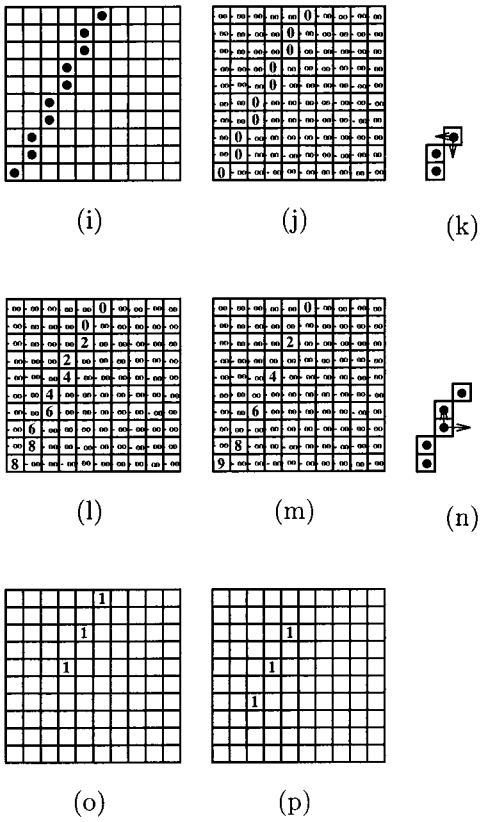

Fig. 12. Illustration of recursive erosion algorithm. (i) Input binary image, (j) initialization, (k) basic SE of length $n = 3$, (l) 2-point transform, (m) $(n-2)$ point transform, (n) actual structuring element of length $m = 5$, (o) transform thresholded by $m = 5$, and (p) translate the thresholded image to account for the true origin. This is the final erosion result.

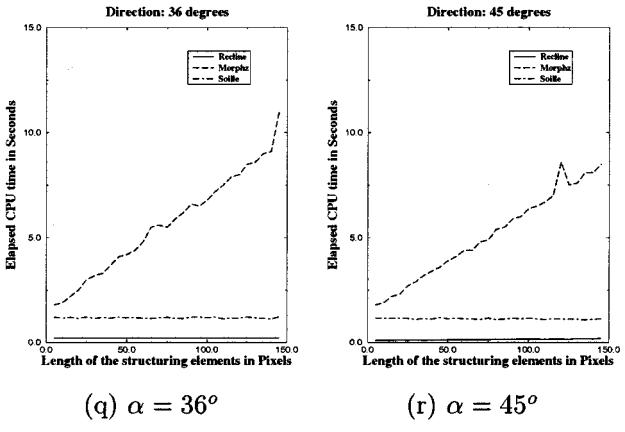

Fig. 13. Comparison of the elapsed times of the recursive (recline), conventional (Morphz) and Soille's dilation algorithms. The angles shown in the figure are actually the orientations of the line. To get the orientation θ of their normals we need to compute $\theta = \alpha + (\pi/2)$.

achieved a speed up of about 100, when the structuring element is 145 pixels long. In this case the algorithm ran at a constant time of about 100 ms. The reason for different speed ups are due to the special treatment of some orientations as explained in

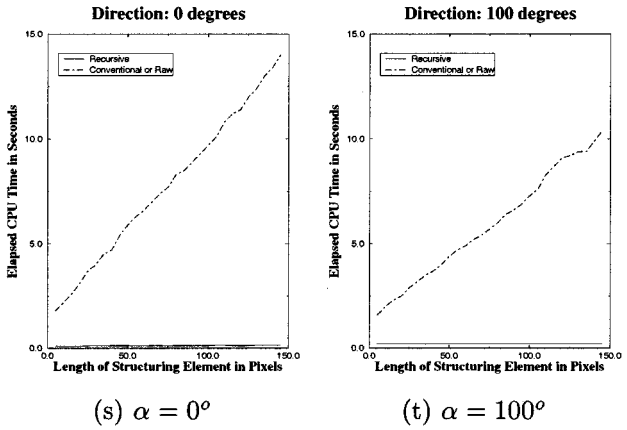

Fig. 14. Comparison of the elapsed times of the recursive and conventional erosion algorithms. The angles shown in the figure are actually the orientations of the line. To get the orientation θ of their normals we need to compute $\theta = \alpha + (\pi/2)$.

Algorithm V.1. We also showed that we performed better over Soille's algorithm when used in binary morphology.

3) Computational Complexity: The conventional algorithm of taking unions and intersections of the structuring element whose domain size is $k \times k$, with the image, requires k^2 unions and intersections, $2k^2$ comparisons, and$3k^2$ memory accesses (a total of $6k^2$ operations) per pixel to obtain dilations and erosions, respectively. Soille's algorithm is independent of domain size (i.e., length of the structuring element) and uses four min/max operations, 19 comparisons, 23 additions and subtractions, 14 divisions, 30 memory accesses (total of 90 operations) per result pixel. Our algorithm, which is only for *binary images*, is also independent of the length of the line structuring element and requires 14 comparisons, one min/max operation, 15 memory accesses, two divisions and multiplications, 15 additions and subtractions, one thresholding, and one translation operation (a total of 48 operations) per result pixel. Considering only the min/max operations used for obtaining the dilation/erosion transform images, we have a speed up of about four over Soille's algorithm.

VIII. Conclusion

In this paper, we presented two-pass algorithms that ran at constant time for obtaining dilations and erosions of binary images, irrespective of the length and orientation of the line structuring elements. One of the main contributions of this paper has been the use of the concept of orientation error between a continuous line and its discrete counterpart. This user-specified orientation error was used in determining the minimum length of the basic digital line structuring element used in obtaining the transforms. The transforms were then thresholded by the actual length of the structuring elements to obtain the binary dilation and erosion results. The algorithms required only 48 operations per result pixel, as opposed to Soille's algorithm which used 90 operations per result pixel. We tested the algorithms on a set of 240×250 salt and pepper noise images with probability of a

pixel being a 1-pixel set to 0.25, on Sun Sparc Station 10 for orientations in the range $[0°, 180°]$ and lengths, in pixels, in the range $[5, 145]$. We achieved a speed up of about 50 (and for special orientations discussed in Algorithm V.1 a speed up of about 100) when the structuring elements had lengths of 145 pixels, over the conventional methods in these experiments. We also compared the results of our dilation algorithm with that discussed by Soille, *et al.*, in [1] and showed that for binary dilation or erosion our algorithm performed much better and achieved a speed up of about four when dilation or erosion transform alone is obtained.

REFERENCES

[1] P. Soille, E. J. Breen, and R. Jones, "Recursive implementation of erosions and dilations along discrete lines at arbitrary angles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 562–567, May 1996.

[2] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.

[3] J. Serra, Ed., *Image Analysis and Mathematical Morphology, Part II: Theoretical Advances*, New York: Academic, 1988.

[4] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 532–550, Apr. 1987.

[5] P. Maragos, "Pattern spectrum and multiscale shape representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 701–716, July 1989.

[6] S.R. Sternberg, "Morphology for graytone functions," *Comput. Vis., Graph., Image Process.*, vol. 35, pp. 333–355, 1986.

[7] P. Maragos and R. W. Schafer, "Morphological filters I: Their set theoretic analysis and relations to linear shift invariant filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1153–1169, 1987.

[8] ——, "Morphological filters II: Their relation to median, order-statistics and stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1170–1184, 1987.

[9] T. R. Crimmins and W. M. Brown, "Image algebra and automatic shape recognition," *IEEE Trans. Aerosp., Electron. Syst.*, vol. 21, pp. 60–69.

[10] R. C. Vogt, *Automatic Generation of Morphological Set Recognition Algorithms*. New York: Springer-Verlag, 1989.

[11] G. X. Ritter, J. N. Wilson, and J. L. Davidson, Image algebra: An overview, in Computer Vision, Graphics and Image Processing. to appear.

[12] M. Van Droogenbroeck, "On the implementation of morphological operations," in *Mathematical Morphology and Its Applications to Image Processing*, J. Serra and P. Soille, Eds. Dordrecht, The Netherlands: Kluwer, 1994, pp. 241–248.

[13] J. Serra and L. Vincent, "An overview of morphological filtering," *Circuits, Syst., Signal Process.*, vol. 11, no. 1, pp. 47–108, 1992.

[14] I. Young and R. Pevereni *et al.*, "A new implementation for the binary and Minkowski operators," *Comput. Graph. Image Process.*, vol. 17, pp. 189–210, 1981.

[15] R. M. Lougheed, "Architectures and algorithms for digital image processing II," in *Proc. Soc. Photo-Optic Instrumentation Engineering*, vol. 534, 1985, pp. 22–33.

[16] S. R. Sternberg, "Cellular computers and biomedical image processing," in *Multicomputers and Image Processing—Algorithms and Programs*, L. Uhr, Ed. New York: Academic, 1982, vol. 35, pp. 291–305.

[17] J. L. Potter, "Image processing on the massively parallel processor," *Comput.*, vol. 16, pp. 62–67, 1983.

[18] J. N. Wilson, G. R. Fischer, and G. X. Ritter, "Implementation and use of an image processing algebra for programming massively parallel machines," *Frontiers Mass. Parallel Comput.*, Oct. 1988.

[19] G. X. Ritter and P. D. Gader, "Image algebra techniques for parallel image processing," *J. Parallel Distrib. Comput.*, vol. 4, pp. 7–44, 1987.

[20] P. D. Gader, "Image algebra techniques for parallel computation of discrete Fourier transforms and general linear transforms," Ph.D. dissertation, Univ. Florida, Gainesville, FL, 1986.

[21] J. L. Davidson, "Lattice structures in the image algebra and applications to image algebra," Ph.D. dissertation, Univ. Florida, Gainesville, FL, 1986.

[22] X. Zhuang and R. M. Haralick, "Morphological structuring element decomposition," *Comput. Vis., Graph., Image Process.*, vol. 35, pp. 370–382, 1986.

[23] F. Y. Shih and O. R. Mitchell, "Threshold decomposition of grayscale morphology into binary morphology," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 31–42, 1989.

[24] ——, "Decomposition of grayscale morphological structuring elements," in *IEEE Workshop Comput. Vis.*, Miami Beach, FL, Dec. 1987, pp. 304–306.

[25] G. X. Ritter and D. Li, "Template decomposition in image algebra," *preprint.*

[26] P. Soille, E. Breen, and R. Jones, "A fast algorithm for min/max filters along lines arbitrary orientation," in *Proc. 1995 IEEE Workshop Nonlinear Signal Image Processing*, June 1995, pp. 987–990.

[27] M. Van Herk, "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernals," *Pattern Recognit. Lett.*, vol. 13, pp. 517–521, 1992.

[28] X. Wang and G. Bertrand, "Some sequential algorithms for a generalized distance transformation based on Minskowski operations," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 1114–1121, Nov. 1992.

[29] S. Chen and R. M. Haralick, "Recursive erosion, dilation, opening, and closing transforms," *IEEE Trans. Image Processing*, vol. 4, pp. 335–345, Mar. 1995.

[30] I. Pitas, "Fast algorithms for running ordering and max/min calculations," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 795–804, June 1989.

[31] A. Rosenfeld and J. L. Pfaltz, "Distance functions in digital pictures," *Pattern Recognit.*, vol. 1, pp. 33–61, 1968.

[32] G. Bertrand and X. Wang, "An algorithm for a generalized distance transformation based on Minskowski operations," in *Proc. 9th ICPR*, Nov. 1988, pp. 1163–1167.

[33] D. Nadadur, A recursive algorithm for binary dilation using line structuring elements, ISL Tech. Rep. ISL-TR-94-08, 1994.

[34] D. Nadadur and R. M. Haralick, "Recursive morphology using line structuring elements," in *Proc. ISMM'96*, Atlanta, GA, May 11–13, 1996.

[35] S. Crabtree, L.-P. Yuan, and R. Ehrlich, "A fast and accurate erosion-dilation method suitable for microcomputers," *Comput. Vis., Graph., Image Process.*, vol. 53, pp. 283–290, May 1991.

[36] B. Laÿ, "Recursive algorithms in mathematical morphology," *ACM Trans. Graph.*, vol. 6, pp. 691–696, 1987.

[37] J. Pecht, "Speeding up successive Minskowski operations," *Pattern Recognit. Lett.*, vol. 3, no. 2, pp. 113–117, 1985.

[38] I. Ragnemalm, "Fast erosion and dilation by contour processing and thresholding of distance maps," *Pattern Recognit. Lett.*, vol. 13, pp. 161–166, 1985.

[39] R. van den Boomgaard and R. van Balen, "Methods for fast morphological image transforms using bitmapped images," *Comput. Vis., Graph., Image Process.*, vol. 54, pp. 252–258, May 1992.

[40] L. Vincent, "Morphological transformations of binary images with arbitrary structuring elements," *Signal Process.*, vol. 22, pp. 3–33, Jan. 1991.

[41] W. Wu, "Automated boundary detection for ultrasound images of the left ventricle," M.S. thesis, Univ. Washington, Seattle, WA, 1993.

[42] J. Bresenham, "Algorithm for computer control of digital plotter," *IBM Syst. J.*, vol. 4, pp. 25–30, 1965.

[43] A. Rosenfeld, "Digital straight line segments," *IEEE Trans. Comput.*, vol. C-23, pp. 1264–1269, 1974.

[44] R. M. Haralick and L. G. Shapiro, *Comput. Robot Vis.*. Reading, MA: Addison-Wesley, 1992.

[45] A. Gillies, "Machine learning procedures for generating image domain feature detectors," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1985.

[46] J. S. Wiejak, H. Buxton, and B. F. Buxton, "Convolution with separable masks for early image processing," *Comput. Vis., Graph., Image Process.*, vol. 32, pp. 279–290, 1985.

[47] G. Strang, *Linear Algebra and Its Applications*. New York: Academic, 1976.

[48] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins Press, 1983.

[49] B. Chaudhuri, "An efficient algorithm for running window pel gray level ranking in 2D images," *Pattern Recognit. Lett.*, vol. 11, no. 2, pp. 77–80, 1990.

Desikachari Nadadur was born in Cuddapah, India, in 1970. He received a B.Tech. degree in electronics and communications engineering from Sri Venkateswara University, Tirupati, India, in 1990, and the M.S. degree in electrical engineering from Gonzaga University, Spokane, WA, in 1993. He is currently pursuing the Ph.D. degree in the Electrical Engineering Department, University of Washington, Seattle, specializing in image processing and computer vision.

He joined the Intelligent Systems Laboratory (ISL), University of Washington, Seattle, in 1993. From 1993 to 1997, he was a Predoctoral Research Associate at ISL. From 1997 to 1999, he was a Software Engineer at Electronics for Imaging, Inc., Foster City, CA. He is currently a Senior Scientist with the Ultrasound Group, Siemens Medical Systems, Inc., Issaquah, WA. His other research interests include mathematical morphology, performance characterization of computer vision algorithms, Bayesian image analysis, application of deformable, and finite element models to cardiac ultrasound image segmentation.

Robert M. Haralick (F'84) is the Boeing Clairmont Egtvedt Professor with the Department of Electrical Engineering, University of Washington, Seattle. He is responsible for developing the gray scale co-occurrence texture analysis technique and the facet model technique for image processing. He has worked on robust methods for photogrammetry and developed fast algorithms for solving the consistent labeling problem in high-level computer vision. He has developed shape analysis and extraction techniques using mathematical morphology, the morphological sampling theorem, and fast recursive morphology algorithms. Together with Prof. I. Phillips, he has developed a comprehensive ground-truthed set of some 1600 document image pages (most in English) and some 200 pages in Japanese in the area of document image understanding. He has also developed algorithms for document image skew angle estimation, zone delineation, and word and text line bounding box delineation. His most recent research is in the area of computer vision performance characterization and covariance propagation. He has published more than 490 papers.

Dr. Haralick is a Fellow of IEEE for his contributions in computer vision and image processing and a Fellow of the International Association for Pattern Recogntion (IAPR) for his contributions in pattern recognition, image processing, and for service to IAPR. He has just completed his term as the President of the IAPR.