Recursive Opening Transform

Robert M. Haralick, Su Chen and Tapas Kanungo
Department of Electrical Engineering, FT-10
University of Washington
Seattle, Washington 98195

Abstract

The opening transformation on N-dimensional discrete space Z^N is discussed. The transform is efficient to compute the binary opening (closing) with any sized structuring element. It also provides a quick way to calculate the pattern spectrum of an image. The pattern spectrum is found to be nothing more than a histogram of the opening transform. An efficient two-pass recursive opening transform algorithm is developed and implemented. The correctness of the algorithm is proven and some experimental results are given. The results have shown that the execution time of the algorithm is a linear function of n, where n is the product of the number of binary one pixels in the input binary image and the number of points in the structuring element. When the input binary image size is 256 × 256 and 50% of the image is covered by the binary one pixels, it takes approximately 250 milliseconds to do an arbitrary sized line opening and it takes approximately 500 milliseconds to do an arbitrary sized box opening on the Sun/Sparc II workstation (with C compiler optimization flag on).

1 Introduction

The mathematical morphology has drawn much attention in the computer vision community since the initial work by Serra [1]. The technique is proven to be a very powerful tool in shape analysis. There is a large body of literature addressing the theoretical aspects of the morphological operators [2] as well as their various applications [3].

However, one of the challenging problems remaining in this area is to develop efficient algorithms to perform the morphological operations. This kind of development will have a great impact on many real-time vision systems where the morphological operations are computationally intensive, especially when the size of the structuring elements becomes large.

One way out of this dilemma is to develop recursive morphological filters. The recursive morphological operator is one type of morphological operator whose output depends not only on the input pixels which are covered by the domain of the structuring element, but also on one or more previously computed output values. The recursive filters are generally computationally more efficient than their non-recursive counterparts. Haralick [5] and Bertrand [6] described one such type of a filter, the generalized distance transform (GDT) which is a generalization of the distance transform first developed by Rosenfeld and Pfaltz [4]. The GDT is very efficient in performing the binary erosion with an arbitrary sized structuring element. For a N-point structuring element, the required maximum number of operations per pixel is $N+2$.

In this paper, we will first review some of the morphological operations and the GDT. Then we will introduce the concept of the opening transform (OT) and show how it can be used to calculate the binary opening with an arbitrary sized structuring element. The opening transform also provides a quick way to compute the pattern spectrum of an image. It is found that the pattern spectrum is nothing more than a histogram of the opening transform [3]. An efficient two-pass recursive opening transform algorithm requiring about $14N$ operations per pixel for an N-point structuring element is described in detail. The theoretical proof of the alogrithm is not given due to the lack of space. Finally, some experimental results are provided.

2 Definitions and Notations

In this section, some of the morphological operations and the generalized distance transform are reviewed. Let A, K are sets in Z^N.

Definition 1: The **dilation** of A by a structuring element K is denoted by $A \oplus K$ and is defined by $A \oplus K =$

$\{c \in Z^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in K\}$.

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

Definition 3: The **opening** of a set A by a structuring element K is denoted by $A \circ K$ and is defined as $A \circ K = (A \ominus K) \oplus K$.

Definition 4: The **closing** of a set A by a structuring element K is denoted by $A \bullet K$ and is defined as $A \bullet K = (A \oplus K) \ominus K$.

Definition 5: The **n-fold dilation** of a set K is denoted by $(\oplus_n K)$ and is defined as

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

Definition 6: The **translation** of a set K by an element $t \in Z^N$ is denoted by $(K)_t$ and is defined as $(K)_t = K \oplus \{t\}$.

The generalized distance transform is based on successive morphological erosions of an image. The GDT 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 $x \in A \ominus_{n-1} K$. The largest integer n satisfies the constraints that $x \in A \ominus_{n-1} K$, but $x \notin A \ominus_n K$.

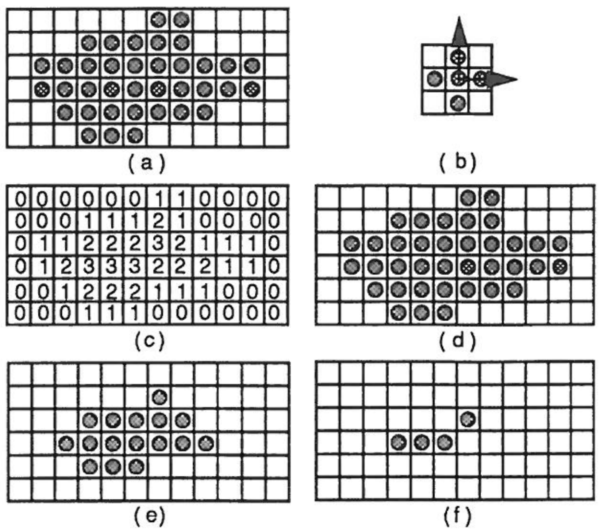
Definition 7: The **generalized distance transform** of a set $A \subseteq Z^N$ with respect to a structuring element $K \subseteq Z^N$ is denoted by $GDT[A, K]$ and is defined as

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

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

The following proposition shows that once the GDT has been computed, it only requires a simple thresholding to compute a binary erosion with any sized structuring element. The GDT and its subsequent thresholding processes are illustrated in Figure 1.

Proposition 1: Let n be a non-negative integer. If $A \subseteq Z^N$ is a set, $K \subseteq Z^N$ is a structuring element, and $B_n = \{x \in A | GDT[A, K](x) > n\}$, then $B_n = A \ominus (\oplus_n K)$.



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

Figure 1: illustrates an instance of the generalized distance transform. (a) a 2-D binary image; (b) a diamond shaped structuring element; (c) the GDT of the binary image; (d) the thresholded GDT image ($n = 0$); (e) the thresholded GDT image ($n = 1$); (f) the thresholded GDT image ($n = 2$).

An efficient two-pass recursive GDT algorithm is described in [5][6]. By careful accounting, it shows that the algorithm requires $N + 2$ operations on each binary one pixel and 2 operations on each binary zero pixel.

Our implementation of the recursive GDT algorithm on the Sun/Sparc II workstation shows that when the C compiler optimization flag is set and the 2 × 2 box structuring element is applied, it takes approximately 60 milliseconds to perform the GDT on a 256 × 256 binary image (assuming 50% of the image is covered by binary one pixels).

3 Opening Transformation

The definition of the opening transform is given in an analogous way to the GDT. 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 the $(\oplus_{n-1} K)$ can be translated so that it covers x and is contained in A.

Definition 8: The **opening transform** of a set $A \subseteq Z^N$ by a structuring element $K \subseteq Z^N$ is denoted by

$OT[A, K]$ and is defined as

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

Based upon the above definition, it is easy to prove the following properties of the opening transform. 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 upon the origin of the K. The opening transform and its subsequent thresholding processes are illustrated in Figure 2.

Proposition 2: Let n be a non-negative integer. If $A \subseteq Z^N$ is a set, $K \subseteq Z^N$ is a structuring element, and $B_n = \{x \in A | OT[A, K](x) > n\}$, then $B_n = A \circ (\oplus_n K)$.

Proposition 3: If $A \subseteq Z^N$ is a set, $K \subseteq Z^N$ is a structuring element, $K' = (K)_t$ is a translation of K by $t \in Z^N$, then $OT[A, K'](x) = OT[A, K](x)$, for all x $\in A$.

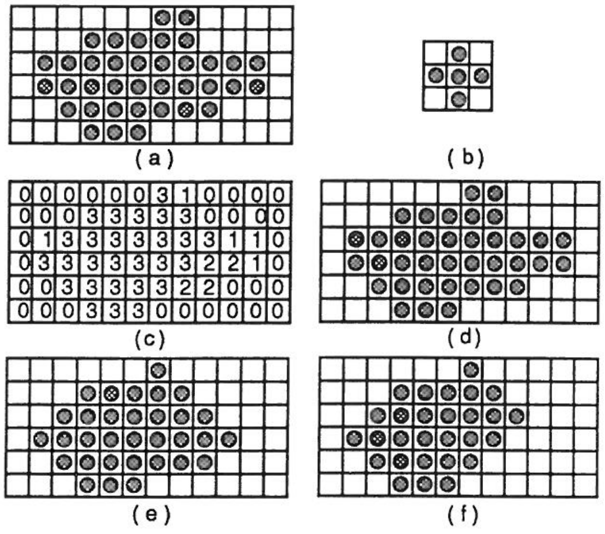


Figure 2: illustrates an instance of the opening transform. (a) a 2-D binary image; (b) a diamond shaped structuring element; (c) the OT of the binary image; (d) the thresholded OT image ($n = 0$); (e) the thresholded OT image ($n = 1$); (f) the thresholded OT image ($n = 2$).

The OT of an image can be computed via a brute force iterative OT algorithm, but the large amount of computation involved poses a serious problem to the application of the algorithm. Suppose the input image I is $M \times M$ and K is a N-point structuring element. The algorithm requires first successively eroding the input image I (in the worst case, the number of erosions is $[M/2]$) and then successively dilating each intermediate eroded image $I_e^{(i)}$ for i number of times ($i = 0, 1, 2, \cdots$). Let $I_e^{(0)} = I$ and $I^{(i)} = I_e^{(i)} \oplus_i K$. The OT image $OT[I, K]$ can be obtained as:

$$OT[I, K](x) = max\{(i + 1) I^{(i)}(x) | i = 0, 1, \cdots, [M/2]\}.$$

The number of operations per pixel required to perform this calculation is $M(MN + 6N + 8)/8$ since we assume each erosion and dilation involves N operations per pixel. In comparison, the following two-pass recursive algorithm takes about $14N$ operations per pixel on the basis of our experimental results.

4 A Two-pass Recursive Algorithm for Opening Transform

The binary opening operation can be accomplished via successively eroding and dilating a set by a structuring element. It is reasonable to conjuncture that the OT algorithm will also consist of such similar processes suitably done recursively.

The GDT provides the transformation from the binary image space to the GDT image space. The GDT value at a given image point is defined as the generalized distance of the point to the image background. Accordingly, the value also defines the extent that one can dilate the given image point subject to the constraint that the dilated pattern still contains in the image. From the definition of the OT, it is not difficult to see that the process of the OT requires each point in the GDT image space to be dilated to its maximum limit and then combines the dilated patterns in the same way as the brute force algorithm does. This kind of process can be well described in terms of the label propagation process. The OT algorithm we develop is thus composed of the following two sequential modules:

- The GDT distance transform module;
- The OT propagation module.

To facilitate the discussion of the propagation algorithm, we need first introduce some basic concepts.

4.1 Basic Definitions and Concepts

Definition 9: A **propagator** on Z^N is defined as a triple, denoted by $\varphi = (l, f, x)$, where l is a label that φ propagates (called the propagating label), f is a positive integer number with the number $(f-1)$ specifying the number of times left that φ can still propagate (called the propagating factor), and x is a point in Z^N that indicates the location of φ (called the propagating point).

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

Definition 10: The **propagation** P of a propagator $\varphi = (l, f, x)$ by a structuring element K is denoted by $P[\varphi, K]$ and is defined as $P[\varphi, K](z) =$

$$\begin{cases} (l, GDT[(\oplus_{f-1}K)_x, K](z), z) & \textit{if } z \in (\oplus_{f-1}K)_x \\ (0,0,z) & \textit{otherwise,} \end{cases}$$

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

Figure 3 illustrates the process of the propagation. The propagation $P[\varphi, K]$ generates a set of non-dummy propagators in the domain of $(\oplus_{f-1}K)_x$. The original propagating label l of φ is propagated all the way to the extent of $(\oplus_{f-1}K)_x$; while the propagating factor at z is $GDT[(\oplus_{f-1}K)_x, K](z)$.

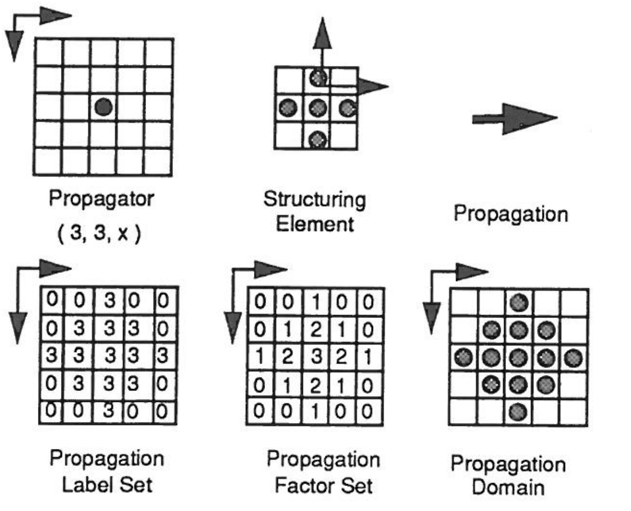


Figure 3: illustrates an instance of the propagation process. x is (2, 2) in this case. The propagation $P[\varphi, K]$ generates a set of non-dummy propagators in the domain of $(\oplus_{f-1}K)_x$.

The following proposition directly relates the OT to the GDT and the above propagation operation.

Proposition 4: Let $A \subseteq Z^N$ be a set, $O \in K \subseteq Z^N$ be a structuring element containing the origin, the GDT of the set A with respect to K be denoted by $GDT[A, K]$. If we define:

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

and

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

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

An explanation of the Proposition 4 is that when given the GDT 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 of φ_z in the domain of A and stack these propagations on top of each other according to their own origin z. This will generate multiple propagators at each point $x \in A$. In the last step, we choose $B(x)$ to be the maximum propagating label among all the propagators at x. The relationship $B(x) = OT[A, K](x)$ will hold for all $x \in A$.

The following Proposition 5 shows the relationship between the propagations of two or more propagators located at the same point.

Proposition 5: If $\varphi_1 = (l_1, f_1, x)$ and $\varphi_2 = (l_2, f_2, x)$ are two propagators at point x, $l_1 \geq l_2$ and $f_1 \geq f_2$, then for all z $\in$ A,

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

The proposition shows that to calculate the OT, it is not necessary to propagate all the propagators of the same point. It is, therefore, useful to introduce the following concept of the propagator list.

Definition 11: The **propagator list** at a propagating point x is defined as an ordered list of propagators that satisfies the following constraints:

Suppose $\Phi(x)$ is a propagator list, $\Phi(x) = \{(l_0, f_0, x), (l_1, f_1, x), ..., (l_m, f_m, x)\}$

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

4.2 A Recursive Propagation Algorithm

In this section, we shall show that the propagation operation $P[\varphi, K]$ can be computed recursively.

We shall consider one special case that the origin O of the structuring element $K \subseteq Z^N$ 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: $O = (x_0, y_0)$, where $y_0 = min\{y|(x,y)\in K\}$ and $x_0 = min\{x|(x, y_0)\in K\}$.

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

Proposition 6: Let $\varphi = (l, f, x)$ be a propagator on Z^N, K be a structuring element with its origin O being the top-most left point of K and $\check{K} = \{-x \mid x \in$ K and x $\neq O\}$ be a reflection of K. If $R[\varphi, K]$ is defined as:

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

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

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

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

The proposition states that the propagation $P[\varphi, K]$ can be computed by scanning a propagator array which at first contains only one propagator φ. At each point 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 non-zero, we assign the propagating label to be l; otherwise assign the propagating label to be zero. Figure 4 shows an example of the recursive propagation process.

4.3 Two-pass Recursive OT Algorithm

Suppose $A \subseteq Z^2$ is a set, and $K \subseteq Z^2$ is a structuring element. Choose the scanning S to be left-to-right and top-to-bottom.

Algorithm: *Recursive Openning Transform–2D Case*

1. Set the origin O of K to be the top-most left point in the domain of K.

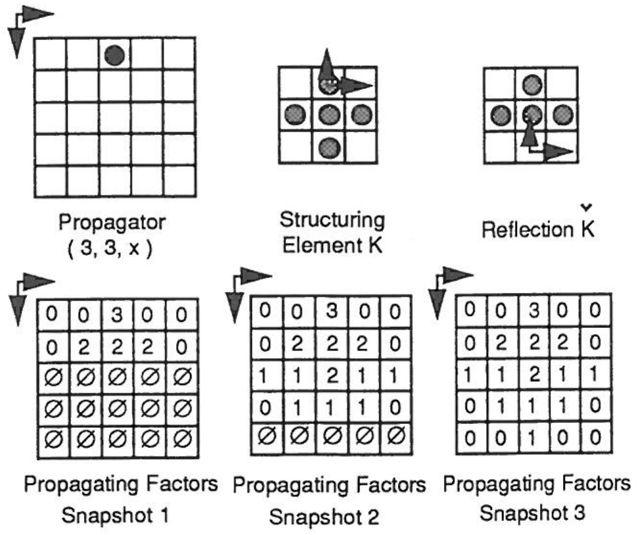


Figure 4: illustrates an instance of the recursive propagation process. x is (0, 2) in this case. ϕ is used to identify that the current point has not been scanned.

2. Decompose the K with respect to the scanning S: the forward filtering structuring element K^f will be the empty set ϕ (origin excluded); the backward filtering structuring element will be K^b (origin excluded).

3. Boundary initialization: assume binary zero pixel replication on the input binary image boundary.

4. The first pass of OT (backward scanning S^*): execute the GDT algorithm described in Haralick [5] and Bertrand [6] on the input binary image.

5. Flip the backward filtering structuring element K^b, let it be $\check{K}^b$.

6. The second pass of OT (forward scanning S): for each point x on the scanning path, do the followings,
   - Let $\Phi(x)$ be the propagator list at point x;
   - If $GDT[A,K](x) = 0$, set $\Phi(x) = \{(0,0,x)\}$;
   - If $GDT[A,K](x) \neq 0$, perform the following steps:
     (a) Let S_0, S_b be propagator set;
     (b) $S_0 = \{(l_0, f_0, x)| l_0 = GDT[A,K](x), f_0 = GDT[A,K](x)\}$;
     (c) $S_b = \{(l, f-1, x) \mid (l, f, x+b) \in \Phi(x+b), b \in \check{K}^b$ and $f > 1\}$;
     (d) $\Phi(x) = MakePropagatorList(S_0 \cup S_b)$; 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 linked list data structure.

- $OT[A, K](x) = L\{TopPropagator[\Phi(x)]\}$; The L is the label function and the $TopPropagator$ is a function to get the top propagator in the propagator list.
- Go to the next point $x \in A$.

5 Results and Discussions

The correctness of the recursive OT algorithm was also confirmed by the experimental results. To do this, an experimental system was setup in such a way that it will be able to generate random test images, compute the OT images via the recursive OT algorithm as well as the iterative brute force algorithm, and compare their results. More than 2,000 test images have passed through such verification.

The timing performance of the recursive OT algorithm was also measured on the Sun/Sparc II workstation (Figure 5). The OT modules were compiled with the optimization flag on. The image sizes were chosen to be 256 × 256.

The algorithm's execution time is proportional to the product of the number of binary one pixels in the input binary image and the number of points in the structuring element. If we perform linear regressions to the experimental data, the ratio of the slopes of the two fitted timing curves of the OT and GDT is approximately 14:1 in the 4-point box structuring element case. It is known that the GDT requires $N + 2$ operations per binary pixel. Thus, the OT will require on average $14N$ operations per binary pixel.

6 Conclusion

In this paper, we define the opening transform of a binary image. A two-pass recursive algorithm was developed to compute the OT. With the recursive GDT and OT algorithms, we can compute the binary erosion, dilation, opening and closing with any sized structuring element in a very efficient way. This will have great significance to real-time vision systems. The opening tranform offers a solution to some vision tasks needing to perform an opening operation but where the structuring element size of that opening has to be determined after an analysis of the opening transform. The opening transform also provides a quick way to compute the pattern spectrum of an

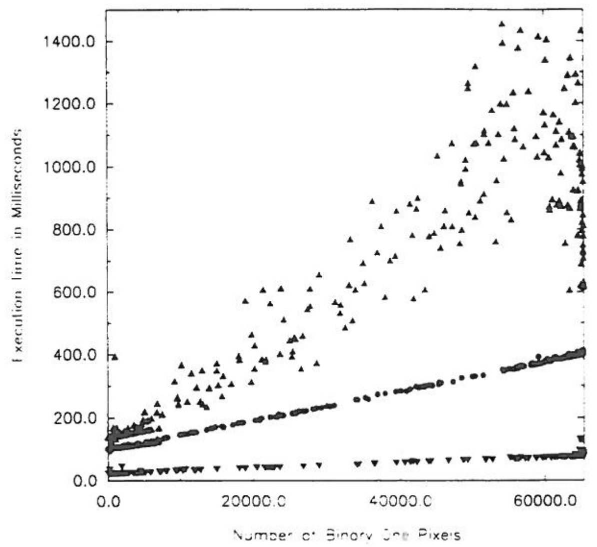


Figure 5: summarizes the timing performance of the two-pass recursive OT algorithm. '●' is for 2-point diagonal line structuring element; '△' is for 4-point box structuring element. The timing of GDT for 4-point box structuring element is also plotted for comparison ('▽').

image for the pattern spectrum is nothing more than a histogram of the opening transform [3].

References

[1] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

[2] R.M. Haralick, S.R. Sternberg and X.Zhuang, "Image Analysis Using Mathematical Morphology", *IEEE Transactions on PAMI*, Vol. 9, No. 4, July 1987, pp. 532-550.

[3] P. Maragos, "Pattern Spectrum and Multiscale Shape Representation", *IEEE Transactions on PAMI*, Vol. 11, No. 7, July 1989, pp. 701-716.

[4] A. Rosenfeld and J.L. Pfaltz, "Distance Functions in Digital Pictures", *Pattern Recognition*, Vol. 1, 1968, pp. 33-61.

[5] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1991

[6] G. Bertrand and X. Wang, "An Algorithm for a Generalized Distance Transformation based on Minkowski Operations", *9th ICPR, Rome*, November 1988, pp. 1163-1167.