A NEW OPTIMIZATION ALGORITHM FOR DILATION AND EROSION

Kuang Yin

Machine Intelligence Laboratory
College of Computer Science, Sichuan University
Chengdu 610065, P.R. China

Abstract

Effectively optimizing dilation and erosion is an extensively studied but not completely resolved problem. In this paper, a new optimization algorithm is proposed to improve the efficiency of dilation and erosion. Four notions are given to define the edges for any simply connected structuring element (SE). An assistant algorithm is proposed to detect these edges. Based on these notions, three iteration equations can be derived, which redefine dilation and erosion as iteration calculation. Time complexity of the new algorithm is reduced to $O(n^3)$. In addition, the new algorithm is suitable for online applications without the decomposition of SE. Simulation shows that with the same parameters, the performance of the new algorithm is better than that of Yang's algorithm.

Keywords: dilation, erosion, optimization, iteration algorithm, time complexity analysis

1 Introduction

Dilation and erosion are the basis of open, close and other operations for Mathematical Morphology. They have been widely used for digital image processing, pattern recognition, and computer vision, etc. Because time complexity of dilation and erosion is $O(n^4)$, in recent years, optimization algorithms of them have been extensively studied [1, 2, 15, 16].

According to the open literature, there are two kinds of methods for optimization. The first kind of method based on chain rules is decomposition of a large structuring element (SE) into a set of smaller elements. A number of researchers have proposed many different optimization algorithms. Zhuang and Haralick [3] presented a tree-search algorithm for decomposition of an arbitrary SE into two-pixel elements. Xu [4] proposed an optimal algorithm for the decomposition of convex SE with a 3x3 region of support. Park and Chin [5] proposed an optimal algorithm for the decomposition of convex SEs

for 4-connected parallel array processors. Li and Ritter [6] developed the decomposition of separable and symmetric convex templates. Hashimoto, Barrera and Ferreira [7] developed a combinatorial optimization technique for the sequential decomposition. If the decomposition exists, the performance can be equivalent or better than [3, 4], and [5]. Several others [8, 9, 12, 13] proposed their own decomposition algorithms, but these algorithms were limited to convex or other restrictive shapes. Park and Chin [11] proposed the decomposition of arbitrarily shaped SEs into 3x3 elements. However, it cannot be suitable for all SEs. Generally speaking, not all SEs have sequential decomposition [4, 7, 11]. Moreover, the general morphological template decomposition problem is NP-complete [10]. Therefore, decomposition is suitable for off-line applications and small pictures. Especially when the picture is smaller than SE, decomposition is inevitable.

The second method is non-decomposition of SE [14]. Compared with the first method, it is suitable for any simply connected SE and online applications.

In this paper, a new optimization algorithm is proposed based on non-decomposition. Through defining the edges of SE, a simply connected SE can be regarded as a set including edge pixels and inner pixels. Then, three iteration equations are derived to redefine dilation and erosion as iteration calculation. By iteration, time complexity is reduced to $O(n^3)$. As the edges of SE are the foundation to form the iteration calculation, an assistant algorithm is proposed to detect these edges.

The rest of this paper is organized in such a manner that in Section 2, an assistant algorithm for the new optimization algorithm is proposed to detect the edges of any simply connected SE. In the next section, the new optimization algorithm is introduced and discussed in detail. In Section 4, the experiment and result are presented and the performance of the optimization algorithm is analyzed minutely. Finally, in Section 5, some conclusions are drawn according to the result in Section 4.

2 An assistant algorithm for detecting edges of SE

2.1 Definition of related notions

Let P and S be binary picture and SE respectively, and (x,y) denotes a pixel. Assume the top-left pixel of S is (0,0), then these following notions can be defined:

Definition 1. *S_{up} is the upper edge of S if and only if*

$$S_{up}=\{(x,y) \mid (x,y) \in S \cap (x,y-1) \notin S\}.$$

Definition 2. *S_{low} is the lower edge of S if and only if*

$$S_{low}=\{(x,y) \mid (x,y) \in S \cap (x,y+1) \notin S\}.$$

Definition 3. *S_{left} is the left edge of S if and only if*

$$S_{left}=\{(x,y) \mid (x,y) \in S \cap (x-1,y) \notin S\}.$$

Definition 4. *S_{right} is the right edge of S if and only if*

$$S_{right}=\{(x,y) \mid (x,y) \in S \cap (x+1,y) \notin S\}.$$

Definition 5. *S_{inner} is the set of inner pixels of S.*

$$S_{inner}=\{(x,y)|(x,y) \in S \cap ((x,y) \notin (S_{up} \cup S_{low}) \cup (x,y) \notin (S_{left} \cup S_{right}))\}.$$

Note that in above definitions, Def. 5 is defined by either Def. 1 and Def. 2, or Def. 3 and Def. 4. That is to say, Def. 5 is a dependent notion. Even for the same S, Sinner determined by S_{up} and S_{low} may be different with Sinner determined by S_{left} and S_{right}.

Fig. 1 is an example of above definitions. Fig. 1(a) is an arbitrarily shaped S and the pixel labelled with ⊙ belongs to Sinner according to Fig. 1(b) and (c). However, it does not belong to Sinner according to Fig. 1(d) and (e). This example confirms that Def. 5 is a dependent notion determined by different kinds of edges. Moreover, in accordance with Fig. 1, some pixels may belong to both Sup and Slow, or both S_{left} and S_{right}, at the same time. (See Fig. 1(d) and (e).) In the new optimization algorithm, these pixels must be included in both S_{up} and S_{low}, or both S_{left} and S_{right}, for iteration.

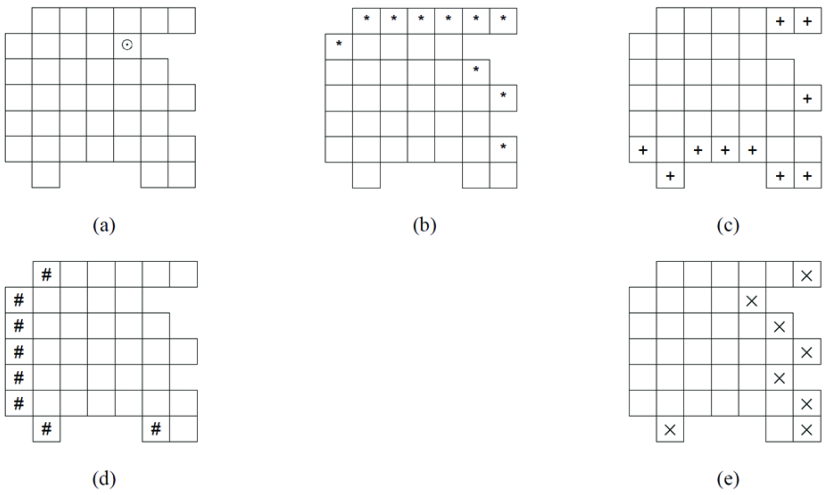

Figure 1. Edges and inner pixels of S: (a) A structuring element, S. (b) S_{up} of S. (c) S_{low} of S. (d) S_{left} of S. (e) S_{right} of S.

2.2 An assistant algorithm and time complexity analysis

After defining the above notions, the next problem is how to detect these edges for the new optimization algorithm. Since Sinner is a dependent notion, clearly, S_{up} and S_{low}, or S_{left} and S_{right}, should be detected at the same time in the assistant algorithm. Take Sup and Slow, for instance, and the assistant algorithm for detecting these edges is summarized as follows:

1. Choose one unprocessed column from S;
2. Scan the current column and record all pixel pairs whose gray scales vary, such as (x,y-1) and (x,y);
3. If the gray scale of (x,y) is 1, then (x,y) $\in S_{up}$;
4. If the gray scale of (x,y-1) is 1, then (x,y-1) $\in S_{low}$;
5. If there are still unprocessed columns in S, then go to step 1);
6. End.

Note that time complexity of the assistant algorithm is mainly determined by step 1), 2) and 5), and the algorithm corresponds to a duplex loop. So the time complexity of the assistant algorithm is $O(n^2)$ at most. If S is scanned row by row, S_{left} and S_{right} can be detected similarly. In applications, the assistant algorithm should be called before calling the optimization algorithm so that the edges can be directly accessed in the optimization algorithm to improve efficiency.

3 A new optimization algorithm for dilation and erosion

3.1 Standard algorithm of dilation and erosion and time complexity analysis

In nature, dilation and erosion are spatial filtering and S corresponds to the filter. Suppose P is $v \times w$ and S is m×n. For *(x,y)*, the output of filter, denoted by *f (x,y)*, is shown as:

$$f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} S(s,t)P(x+s, y+t) \tag{1}$$

where $a=(m-1)/2$, $b=(n-1)/2$, $S(x,y)$ and $P(x,y)$ are (x,y) in S and P, respectively. If *P(x,y)* is updated by

$$P(x,y) = \begin{cases} 0 & f(x,y) = 0, \\ 1 & otherwise. \end{cases} \tag{2}$$

then the calculation defined by equation (1) and (2) is dilation. If *P(x,y)* is updated by

$$P(x,y) = \begin{cases} 0 & f(x,y) = \| S \|, \\ 1 & otherwise. \end{cases} \quad (3)$$

where ‖ S ‖ is the number of 1 in S, then the calculation defined by equation (1) and (3) is erosion.

In order to get a completely filtered *P*, for *x =0,1,2,…,v-1* and *y=0,1,2,…,w-1*, calculations must be repeated according to equation (1), (2) and (3). This is known as the standard algorithm for dilation and erosion, and the time complexity of it is $O(n^4)$, where *n=max(v, w, m, n).*

3.2 A new optimization algorithm and time complexity analysis

Now a conclusion can be drawn that the time complexity of dilation and erosion is mainly caused by calculation in equation (1) because it must be repeated pixel by pixel for P and S. At the same time, it can also be found that in equation (1), there are many calculations that are unnecessary. For example, consider the relationship between *f(x,y)* and *f (x+1,y),* which are the outputs of filter at time t and t+1. Assume $P_{left}(t)$ is the set of P(x,y), which overlaps S_{left} at time t. Similarly we can define $P_{right}(t)$, $P_{up}(t)$, and $P_{low}(t)$. See Fig. 2 and we can get

$$f(x + 1, y) = f(x, y) + ||P_{right}(t + 1)|| - ||P_{left}(t)|| \quad (4)$$

On the contrary, if S horizontally moves from right to left, *f (x,y)* and *f (x -1,y)* are the outputs of filter at time t and t+1. Similarly we can get

$$f(x - 1, y) = f(x, y) + ||P_{left}(t + 1)|| - ||P_{right}(t)|| \quad (5)$$

Finally, if S vertically moves from top to bottom, *f(x,y)* and *f(x,y+1)* are the outputs of filter at time t and t+1. We can get

$$f(x, y + 1) = f(x,y) + ||P_{low}(t + 1)|| - ||P_{up}(t)|| \quad (6)$$

Equation (4), (5), and (6) mean that if S continually moves above P pixel by pixel, it is not always necessary to calculate *f(x,y)* according to equation (1). Furthermore, if S moves along the route shown in Fig. 3, in equation (4), (5), and (6), there will always be only one equation suitable for calculation, except at *f(0,0).* But it doesn't matter because *f(0,0)* can be initialized according to equation (1). Obviously, the initialization of *f(0,0)* will not increase time complexity of the optimization algorithm because only one pixel needs to do so. Thus, dilation and erosion are redefined as the iteration calcu-

lation and the edges of S can be detected by the assistant algorithm introduced in Section 2.

Finally, the description of the optimization algorithm can be summarized as follows:

1. Initialize *f(0,0)* and other basic variables, such as calling the assistant algorithm to detect edges of S;
2. If processing *P(x,y)* is finished, then go to step 9);
3. If S moves horizontally from left to right, then calculate *f(x,y)* according to equation (4);
4. If S moves horizontally from right to left, then calculate *f(x,y)* according to equation (5);
5. If S moves vertically from top to bottom, then calculate *f(x,y)* according to equation (6);
6. Update *P(x,y)* according to *f(x,y);*
7. Update necessary variables and prepare for the next iteration;
8. Go to step 2);
9. End.

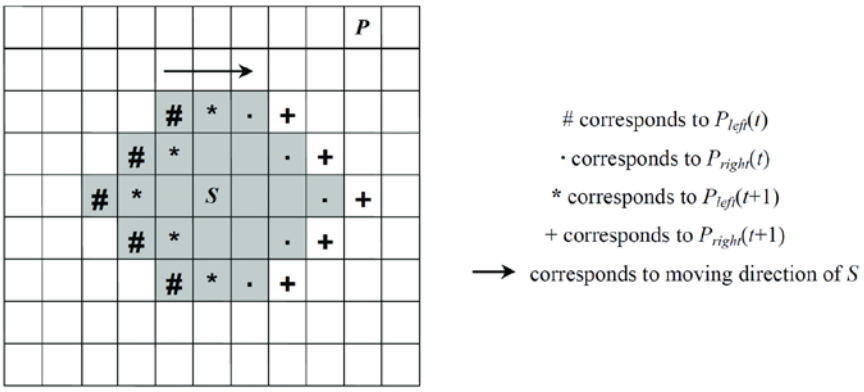

Figure 2. Variation of $P_{left}(t)$, $P_{left}(t+1)$, $P_{right}(t)$, and $P_{right}(t+1)$. Shadow area is P(x,y) covered by S at time t. Clearly, at time t, $P_{left}(t)$, $P_{left}(t+1)$, and $P_{right}(t)$ are in shadow area; at time t+1, $P_{left}(t+1)$, $P_{right}(t)$, and $P_{right}t(t+1)$ are in shadow area. In other words, during t and t+1, $P_{left}(t)$ moves out of shadow area and $P_{right}(t+1)$ moves into shadow area.

Note that step 2) means a duplex loop and steps 3), 4) and 5) include accessing the result of the assistant algorithm. So from step 2) to step 8), it is a triple loop and time complexity is $O(n^4)$ at most.

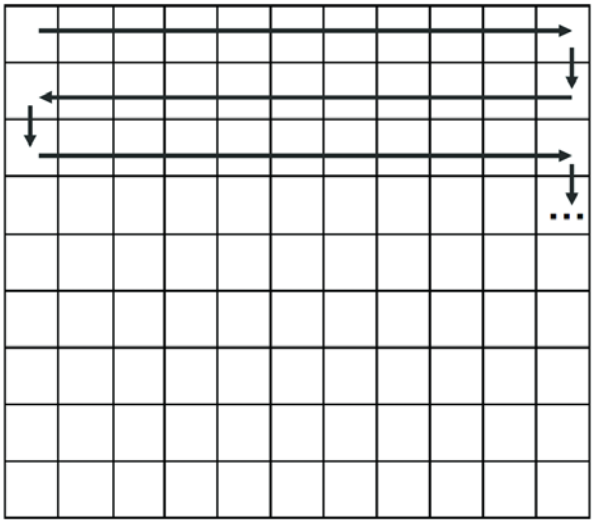

Figure 3. Moving route of S above P.

4 Simulation and result

4.1 Design of experiment

During the experiment, P and S were both randomly initialized as binary matrixes. For the sake of simplicity of parameters, redefine P as w×w and S as m×m. In order to compare the performance of the optimization algorithm with that of the standard algorithm and Yang's algorithm [14], we recorded running time for the three algorithms with different parameters. Running time was measured by the clocks consumed in each experiment. Because in C programming language, 1 second includes 1000 clocks, the accuracy of result is 1 millisecond. At the same time, all experiments with the same parameters were repeated for 100 times and averages were recorded.

4.2 Comparison and analysis of result

The comparison of the result is shown in Fig. 4 and Fig. 5. Fig. 4 shows the relationship between running time and parameter w. Equation (1) implies that for the standard algorithm, the relationship between running time and *w* should be a quadratic function. Fig. 4(a) confirms this conclusion. Similarly, the conclusion is also true for Yang's algorithm and the new optimization algorithm in terms of Fig. 4(b) and (c). In addition, it can be seen in Fig.4 that for different parameter *m*, curves of the new optimization algorithm are more convergent than those of the standard algorithm and Yang's algorithm. It

means that for the new optimization algorithm, m has the least effect on running time. This is exactly what we want.

In Fig. 5, the relationship between running time and m can be seen more clearly. Comparing Fig. 4(a) with Fig. 5(a), m has less effect on running time than w. However, time complexity is usually measured by asymptotic function of performance on the worst condition. So on the basis of Fig. 4(a) and Fig. 5(a), time complexity of the standard algorithm can be estimated as $O(n^4)$, tallying with the conclusion from equation (1). Note that in Fig. 5(c), performance curves are nearly horizontal lines. It suggests that m affects performance almost linearly. Therefore, time complexity of the new optimization algorithm is estimated as $O(n^3)$ according to Fig. 4(c) and Fig. 5(c), being consistent with the analysis in Section 3.

Finally, Fig. 6 shows the comparison of the performance between Yang's algorithm and the new optimization algorithm. It can be inferred that the performance of the new algorithm is better. However, we also find that, the superiority of the new algorithm can be stable only when $w \geq 300$. Otherwise, the superiority is unstable and sometimes the performance of the optimization algorithm is even worse than that of the standard algorithm. This conclusion is also true for Yang's algorithm. The reason for the instability is that if w is not large enough, the improved performance can-not match the additionally consumed time caused by the more complex structure of the optimization algorithm. In fact, according to the No Free Lunch (NFL) theorem, the conclusion will fit all optimization algorithms.

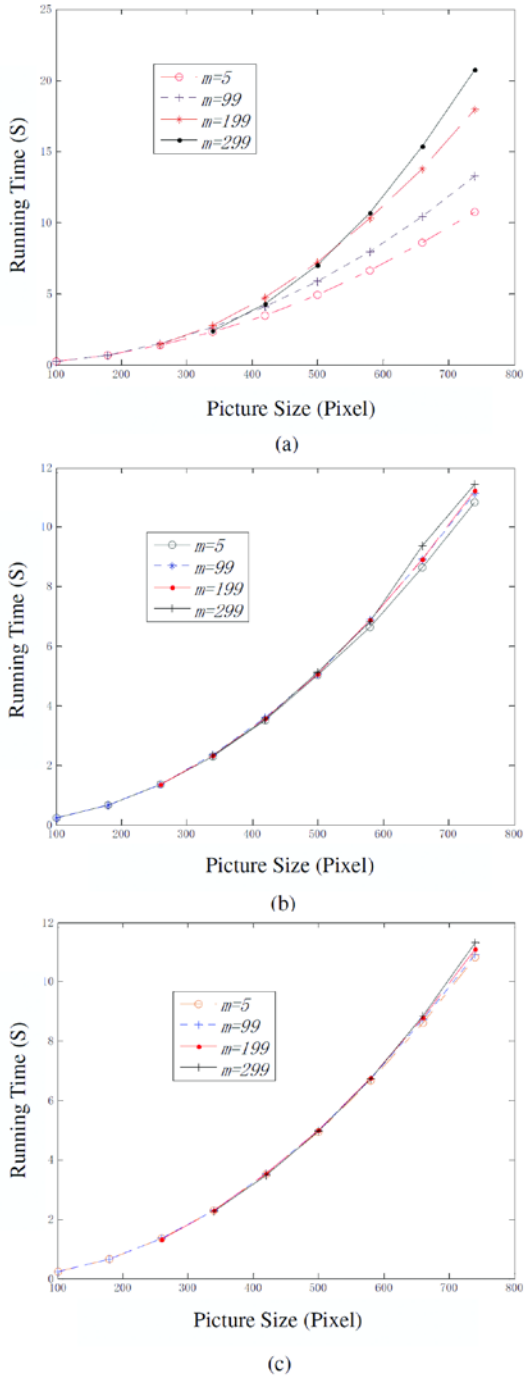

Figure 4. Comparison of performance effected by w: (a) Standard algorithm. (b) Yang's algorithm. (c) The new optimization algorithm.

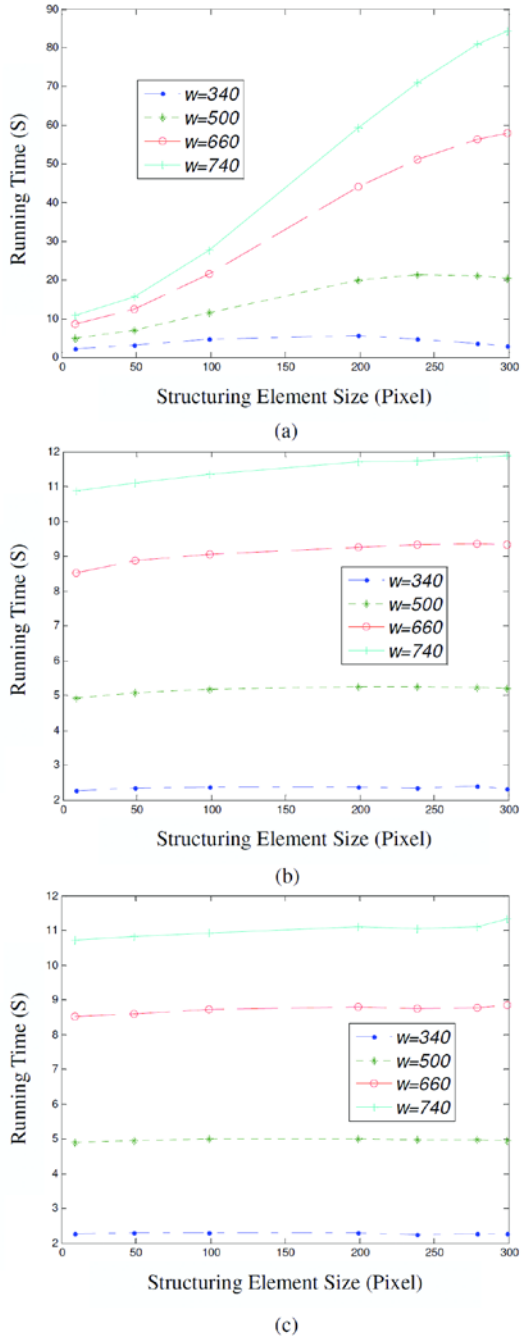

Figure 5. Comparison of performance effected by m: (a) Standard algorithm. (b) Yang's algorithm. (c) The new optimization algorithm.

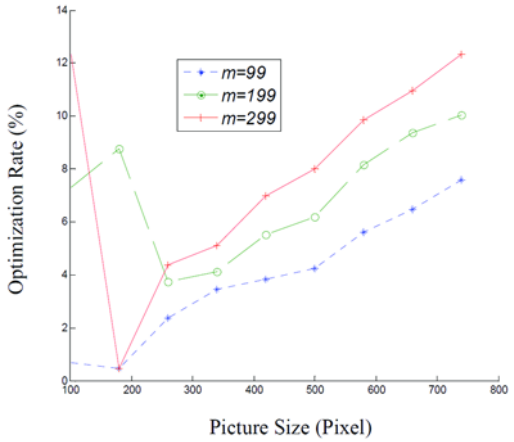

Figure 6. Optimization rate of the new optimization algorithm compared with Yang's algorithm. Optimization rate is defined as the improved performance on the base of the performance of Yang's algorithm, with the same parameters.

5 Conclusions

Optimization for dilation and erosion has been an active research area for a long time. Earlier efforts mainly focused on the decomposition of SE. In this paper, a new optimization algorithm is presented without decomposition. By iteration, the time complexity is reduced to $O(n^3)$. Simulation confirms that the performance of the new algorithm is better than that of Yang's algorithm when $w \geq 300$.

References

1. I. Pitas and A. N. Venetsanpoulos, Jan.1990, *Morphological shape decomposition,* IEEE Trans. Pattern Anal. Machine Intell., Vol. 12, No. 1, pp. 38-45.
2. R. M. Haralick, S. R. Stemberg, and X. Zhuang, July 1987, *Image analysis using mathematical morphology,* IEEE Trans. Pattern Anal. Machine Intell., Vol. 9, No. 4, pp. 532-550.
3 X. Zhuang and R. M. Haralick, Sept. 1986, *Morphological structuring element decomposition*, Computer Vision, Graphics, Image Processing, Vol. 35, pp. 370-382.
4. J. Xu, Feb. 1991, *Decomposition of convex polygonal morphological structuring elements into neighborhood subsets,* IEEE Trans. Pattern Anal. Machine Intell., Vol. 13, No. 2, pp. 153-162.

5. H. Park and R. T. Chin, Mar. 1994, *Optimal decomposition of convex morphological structuring elements for 4-connected parallel array processors,* IEEE Trans. Pattern Anal. Machine Intell., Vol. 16, No. 3, pp. 304-313.
6. D. Li and G. X. Ritter, 1990, *Decomposition of separable and symmetric convex templates,* Image Algebra and Morphological Image Processing, P. D. Gader, ed., Proc. SPIE 1350, pp. 408-418.
7. R. F. Hashimoto, J. Barrera, and C.E. Ferreira, August 2000, *A combinatorial optimization technique for the sequential decomposition of erosions and dilations,* Journal of Mathematical Imaging and Vision, Vol. 13, No. 1, pp. 17-33.
8. X. Zhuang, Jan. 1994, *Decomposition of morphological structuring elements,* Journal of Mathematical Imaging and Vision, Vol. 4, No. 1, pp. 5-18.
9. T. Kanungo, R. M. Haralick, and X. Zhuang, 1990, *B-code dilation and structuring element decomposition for restricted convex shapes,* Image Algebra and Morphological Image Processing, P. D. Gader, ed., Proc. SPIE 1350, pp. 419-429.
10. P. M. Pardalos, P. Sussner, and G. X. Ritter, June 1997, *On integer programming approaches for morphological template decomposition problems in computer vision,* Journal of Combinatorial Optimization, Vol. 1, No. 2, pp. 165-178.
11. H. Park and R. T. Chin, Jan. 1995, *Decomposition of arbitrarily shaped morphological structuring elements,* IEEE Trans. Pattern Anal. Machine Intell., Vol. 17, No. 1, pp. 2-15.
12. S. Y. Ohn, 2005, *Decomposition of 3D Convex Structuring Element in Morphological Operation for Parallel Processing Architectures,* Image analysis and recognition, Springer, ICIAR 2005, pp. 676-685.
13. S. Y. Ohn, 2006, *Neighborhood decomposition of convex structuring elements for mathematical morphology on hexagonal grid,* Computer and Information Sciences, Springer, ISCIS 2006, pp. 511-521.
14. K. Yang, L. B. Zeng, and D. C. Wang, Dec. 2005, *A Fast Arithmetic for the Erosion and Dilation Operations of Mathematical Morphology,* Computer engineering and application, Vol. 41, No. 34, pp. 54-56.
15. G. Anelli, A. Broggi, and G. Destri, Feb. 1998, *Decomposition of arbitrarily shaped binary morphological structruing elements using genetic algorithm,* IEEE Trans. Pattern Anal. Machine Intell., Vol.20, No.2, pp. 217-224.
16. R. F. Hashimoto and J. Barrera, Jan. 2002, *A note on Park and Chin's algorithm,* IEEE Trans. Pattern Anal. Machine Intell., Vol.24, No.1, pp. 139-144.