Efficient 2-D Grayscale Morphological Transformations With Arbitrary Flat Structuring Elements

Erik R. Urbach, *Associate Member, IEEE*, and Michael H. F. Wilkinson, *Senior Member, IEEE*

***Abstract*—An efficient algorithm is presented for the computation of grayscale morphological operations with arbitrary 2-D flat structuring elements (S.E.). The required computing time is independent of the image content and of the number of gray levels used. It always outperforms the only existing comparable method, which was proposed in the work by Van Droogenbroeck and Talbot, by a factor between 3.5 and 35.1, depending on the image type and shape of S.E. So far, filtering using multiple S.E.s is always done by performing the operator for each size and shape of the S.E. separately. With our method, filtering with multiple S.E.s can be performed by a single operator for a slightly reduced computational cost per size or shape, which makes this method more suitable for use in granulometries, dilation-erosion scale spaces, and template matching using the hit-or-miss transform. The discussion focuses on erosions and dilations, from which other transformations can be derived.**

***Index Terms*—Dilation, dilation-erosion scale spaces, erosion, fast algorithm, hit-or-miss transform, mathematical morphology, multiscale analysis.**

I. Introduction

MORPHOLOGICAL operators [2] like dilation and erosion with structuring elements (S.E.) are the most fundamental operators in mathematical morphology and have become common tools for both image filtering and analysis [3], [4] of binary and grayscale images, especially since the development of efficient algorithms [5]–[14]. Usually, these efficient algorithms can only be used for binary images [5]–[7], [9], [12], [13], or they are limited to shapes that can (efficiently) be decomposed into a series of linear S.E.s [8], [11], [14]. Efficient implementations for specialized hardware have also been studied extensively, such as the decomposition of arbitrary shapes into 3×3 blocks [10]. A recent overview of efficient algorithms for morphological operators with linear S.E. and 2-D S.E. decompositions can be found in [15]. All methods based on decomposition of 2-D S.E.s into linear S.E.s share the same limitation: many shapes either cannot be decomposed efficiently or they cannot be decomposed at all. In the binary case, efficient algorithms for some 2-D shapes like circles do exist, but these cannot efficiently be extended to the grayscale case, for which polygonal approximations [11], [16] of circles usually are used instead. For larger circles these approximations tend be either too coarse or too computationally intensive, since the number or linear S.E.s required is proportional to the diameter of the circle.

Algorithms that efficiently perform morphological operators with arbitrary S.E.s not only are important for those cases where the S.E. cannot be decomposed, but also wherever a generic algorithm is desired such as in image processing libraries, which often have a number of specialized routines for specific cases, and a direct implementation for arbitrary S.E. Furthermore, for many applications the benefits of using the fastest specialized algorithm available instead of using one slightly less efficient generic algorithm does not outweigh the costs involved in adapting the methods used. S.E. shape decompositions require some design and programming efforts that can be avoided if a generic algorithm is used. Our algorithm is efficient for any S.E. and only significantly outperformed when large linear S.E.s or compositions of linear S.E.s are used with a dedicated algorithm such as proposed by Gil and Kimmel [14].

Commercial and open source image processing software for performing morphological operations was found to be either quite slow, being based on an processor optimized version of the direct algorithm (like openCV or Matlab) or fast but limited to rectangular S.E.s (like Adobe Photoshop CS2). Olena [17], which uses the algorithm by Van Droogenbroeck and Talbot [1], is one of the few exceptions that is faster and can handle arbitrary S.E.s. Van Droogenbroeck and Talbot [1] proposed an efficient algorithm for computing morphological operations with arbitrary 2-D shapes using a histogram, which makes the computing time of their algorithm dependent on the number of gray levels used. Their idea is to compute for one pixel p of the image the complete histogram based on the intensities of the pixels around p corresponding to elements of the S.E. The value of p after erosion is the minimum intensity in the histogram which has a value >0. For all succeeding pixels of the image (by moving around the S.E. over the image), the histogram is efficiently updated and the position of its minimum intensity changes only if i) a new minimum value is shifted into the histogram, which can be kept track when the histogram is updated, or ii) when the current minimum is shifted out of the histogram, in which case the algorithm searches for the first following (brighter) intensity which is now represented in the histogram.

The C source code of the following algorithms are available on request: i) our proposed ("Urbach-Wilkinson" or UW) algorithm, ii) Van Droogenbroeck and Talbot (DT) [1] for arbitrary 2-D S.E.s, and iii) Gil and Kimmel (GK) for linear S.E.s.

In a preliminary version [18], we presented a new method for performing morphological operators with *any* 2-D flat structuring element that always outperforms existing algorithms for

Manuscript received January 16, 2007; revised October 2, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Philippe Salembier.

The authors are with the Institute of Mathematics and Computing Science, University of Groningen, 9700 AV, Groningen, The Netherlands.

Digital Object Identifier 10.1109/TIP.2007.912582

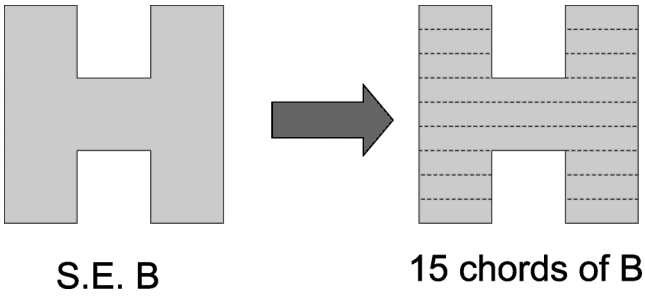

Fig. 1. Decomposition of S.E. B into chords.

arbitrary structuring elements, and which has two advantages: i) it is independent of both image content and the number of gray levels used, and ii) application of a single operator using many different S.E.s can be computed somewhat more efficiently, which may be useful for granulometries [2], [3] and erosion-dilation scale spaces [19]. Compared to Van Droogenbroeck and Talbot's method, it has the further advantage that it also works on floating point data, which is common for images originating from a Fourier transformation, such as images in radio astronomy [20]. This is also useful for generic implementation for any pixel type for which a total order exists [17].

This paper has the following improvements over the preliminary version [18]: i) improvements of the algorithm resulting in a further reduction of the number comparisons and, hence, in speed gains of a factor between 2.0 and 7.7 compared with the earlier version, especially for thin structuring elements, ii) a more detailed and rewritten discussion of the algorithm, and iii) the comparison with existing methods has been extended.

II. Algorithm

For the sake of simplicity and clarity, we limit our discussion here to discrete 2-D grayscale images f and erosions with 2-D flat S.E.s B. It should be noted that our method can be easily adapted for 3-D images and other morphological operations. It is assumed that all images f have their origin top-left and that images are processed in scan-line order.

Like existing methods, our approach to improve the computational efficiency of erosions is by reducing the number of redundant comparisons performed by a direct implementation of its definition

$$(f \ominus B)(x,y) = \min_{(u,v)\in B} f(x+u, y+v). \tag{1}$$

A. Data Structure

Much in the same way as in the binary case in [6], our algorithm decomposes an arbitrary S.E. into a series of chords, i.e., runs of foreground pixels of maximum extent, as demonstrated in Fig. 1 for a letter H. Each chord can be considered as a single horizontal linear S.E. and is represented by a triple containing: i) its y-offset with respect to the origin of the S.E., (ii) its minimal x-position, and iii) its length l. For a S.E. B consisting of N_C chords, we store the number of chords N_C, the set C of N_C chords, the minimum and maximum y-offsets, $y_{\min}$ and $y_{\max}$, the minimum and maximum x-values $x_{\min}$ and $x_{\max}$, and the maximum chord length $l_{\max}$ occurring in B.

B. Computation

Let us consider the computation of the erosion with a flat S.E. B at coordinate (x,y) of a 2-D grayscale image f. Furthermore, let C be the set of chords representing B. We can now compute the minimum intensity value $v_c^{\min}(x,y)$ of the pixels for a chord $c \in C$ translated by (x,y), i.e., the minimum intensity of the $l(c)$ pixels of f from $f(x+x(c), y+y(c))$ to $f(x+x(c)+l(c), y+y(c))$

$$v_c^{\min}(x,y) = \min_{i=0}^{l(c)-1} f\left(x+x(c)+i, y+y(c)\right). \tag{2}$$

Thus, the direct approach for computing an erosion $f \ominus B(x,y)$ of (1) can now also be computed by

$$(f \ominus B)(x,y) = \min_{c\in C} v_c^{\min}(x,y) \tag{3}$$

with C being the set of chords representing B. Algorithms based on either (1) or (3) both have a computational complexity of $O(W \times H \times N)$ for a $W \times H$ image f and a S.E. B consisting of N elements.

Our approach is to split the computation of the erosion $f \ominus B$ for each line y of f in two parts. First, a lookup table is computed in which the minimum values for each chord length is stored for all the pixels belonging to a number of lines, thus basically storing the $v_c^{\min}(x,y)$. After this, the erosion is computed for each pixel (x,y) of f by taking the minimum of the computed values $v_c^{\min}(x,y)$ (stored in lookup table) corresponding to the chords of C translated by (x,y). This will now be discussed in detail.

C. Lookup Table

Let us assume a $W \times H$ image f and a S.E. B of height M. Furthermore, let $L_{\max}(C)$ be the maximum chord length of the set of N_C chords C representing B. Often, only a limited number of different chord lengths exists in a structuring element. Therefore, let us introduce an array $R(i)$ storing for each chord length index i its corresponding chord length. For many structuring elements, the maximum value of i will be considerably less than the number of the chord lengths N_C present in C.

Assume we are eroding line y of image f. This means that for a S.E. B with minimum y-offset $y_{\min}$ and maximum y-offset $y_{\max}$ all $M = y_{\max} - y_{\min} + 1$ lines of f belonging to the rectangle $(0, y+y_{\min}) - (W-1, y+y_{\max})$ are needed.

The lookup table $T_y(i,x,r)$ can now be defined for image f at line y and y offset r as

$$T_y(i,x,r) = \min_{a=0}^{R(i)-1} f(x+a, y+r). \tag{4}$$

Thus, $T_y(i,x,r)$ stores the minimum intensity value of the pixels belonging to the chord with length index i that starts at $(x, y+r)$. Algorithm II.1 illustrates these computations. Note that we avoided implementation issues like handling the boundary of f. In our implementation, we solved this by padding the lookup table for $x < 0$ and $x \geq W$. Furthermore, it is assumed that for each chord length $R(i) > 1$ a smaller chord length $R(i-1)$ is present such that $2 \times R(i-1) \geq R(i)$. For structuring elements that do not satisfy that criterion, we add the necessary intermediate chord lengths to the computation of the lookup table.

A full computation of the lookup table is only necessary for the first line $y = 0$ of the image; for all following lines $y > 0$ the table is updated as follows:

$$T_y(i,x,r) = \begin{cases} T_{y-1}(i,x,r+1), & \text{if } y_{\min} \le r < y_{\max} \\ \min_{a=0}^{R(i)-1} f(x+a, y+r), & \text{if } r = y_{\max}. \end{cases} \quad (5)$$

If we reuse the array of $T_{y-1}(i,x,r)$ for the computation of $T_y(i,x,r)$, then we perform two steps to accomplish (5): i) we copy for each $y_{\min} \le r < y_{\max}$: $T_y(:,:,r) = T_{y-1}(:,:,r+1)$ with the semicolons denoting here the copying of all elements along that dimension and ii) the computation for the elements belonging to $T_y(:,:,y_{\max})$. The copying step can be performed efficiently by implementing the lookup table as a multidimensional array so that this copying can be performed by swapping of pointers. This computation step is a special and limited case of the full computation as described in Algorithm II.1 in that we only need to perform the outermost FOR-loop once, rather than $y_{\max} - y_{\min} + 1$ times.

Algorithm II.1 Algorithm for full computation of the lookup table $T_y(i,x,r)$ for line y of image f.

ComputeTable(f, $y_{\min}$, $y_{\max}$).

INPUTS: A $W \times H$ image f, minimum and maximum y-offsets $y_{\min}$ and $y_{\max}$.

OUTPUT: A fully computed lookup table $T_y(i,x,r)$.

FOR $r = y_{\min}$ TO $y_{\max}$ DO.

  FOR $x = 0$ TO W DO

    $T_y(0,x,r) = f(x, y+r)$.

FOR $i = 1$ TO $L_{\mathtt{num}}(C)$ DO.

  FOR $x = 0$ TO W DO

    $d = R(i) - R(i-1)$

    $T_y(i,x,r) = \min[T_y(i-1,x,r), T_y(i-1, x+d, r)]$.

D. Chord Representation

Constructing the set of chords representing a S.E. B is straightforward: we scan all elements $(x,y) \in B$ and record the the beginning (x-offset and y-offset) and length of each run-length of elements (x,y) and add a chord to C describing that run length.

In the preliminary version [18], for a S.E. B with maximum chord length $l_{\max}$ a lookup table was computed storing the minimum values for the chord lengths $R(i) = 2^i$ with $R(i) < l_{\max}$. Thus, the minimum value for a chord c translated to (x,y) in image f is computed by taking the minimum of the two chords that together cover chord c.

The present algorithm computes the lookup table for the following chord lengths for a S.E. B:

- all chord lengths present in B;
- those chord lengths that are necessary to meet the requirement that the minimum value for any chord length beginning at (x,y) can be computed using only one comparison, i.e., if B contains a chord of length a than the lookup table should also contain the minimum value for a chord of length $a/2 \le b < a$.

E. Performing the Erosion

For each line y, we compute or update the lookup table $T_y(i,x,r)$, after which we erode line y of f and store the result in the output image g. Eroding line y using $T_y(i,x,r)$ is illustrated in Algorithm II.2. Algorithm II.3 shows how to compute the erosion of an image f using a S.E. B.

Algorithm II.2 Algorithm for eroding line y of an image f using lookup table $T_y(i,x,r)$ and storing the result in output image g. Range of gray values for images f and g is $[0 \ldots f_{\max}]$.

LineErode(g, $T_y(i,x,r)$, C).

INPUTS: A $W \times H$ output image g, lookup table $T_y(i,x,r)$, and a set of chords C.

FOREACH $(x,y) \in g$ DO

  $g(x,y) = f_{\max}$.

FOR $x = 0$ TO W DO.

  FOREACH $c \in C$

    $v = T_y(I(c), x + x(c), y(c))$

    IF $v < g(x,y)$ THEN $g(x,y) = v$.

Algorithm II.3 Algorithm efficient computation of erosion $g = f \ominus B$

UWErode(f, B).

INPUTS: A $W \times H$ input image f and a S.E. B.

RESULT: Output image $g = f \ominus B$.

Construct set of chords C representing B.

Compute $T_y(i,x,r)$ for $y = 0$ (Algorithm II.1).

LineErode(g, $T_y(i,x,r)$, C).

FOR $y = 1$ TO H DO

  Update $T_y(i,x,r)$.

  LineErode(g, $T_y(i,x,r)$, C).

The total computation time required for performing an erosion using our proposed algorithm depends strongly on the number of chords. Since some S.E.s have considerably fewer vertical run-lengths than horizontal ones, erosions using those can be computed more efficiently if C would be a list of vertical chords. As the time needed to compute a chord representation C is negligible compared with the total time needed, we compute both a horizontal and a vertical run-length decomposition and use the decomposition with the minimum

number of chords as the chord representation C. The modified chord table $\tilde{T}_y(i,x,r)$ contains in entry (i,x,r) the minimum value for the chord starting at x-coordinate x and table row r with chord length i toward the previous image lines. Updating or a full computation of this table is similar to case of horizontal chords as described earlier. This avoids the need for costly image rotation over 90°.

If the S.E. has the shape of a letter H of width and height 49 and with its legs 1 pixel thick, then using horizontal run-lengths a set of $2 \times 48 + 1 = 97$ chords, whereas using vertical run-lengths the same S.E. is represented using only $2 + 47 = 49$ chords. Experiments showed the same relative difference in the required computation time.

F. Computational Complexity

It is obvious that the algorithm described here is independent in its time complexity of the image content, unlike the method of Van Droogenbroeck and Talbot [1], which uses a histogram for each pixel of the image. If the histogram update leads to a new minimum intensity value present in the histogram, this new minimum is found by a linear search in the histogram. This is efficient for compact S.E.s on relatively smooth images, but becomes very computationally expensive for contrast-rich images, such as noisy images, with thin S.E.s, and with high bit depths leading to large histogram tables. Our method computes the minimum of each chord and the minimum of all chords independent of image content.

Let f be a $W \times H$ grayscale image and B be a $U \times V$ (binary) S.E. Furthermore, let the chord representation C of B contain N_C chords with a maximum chord length of $L_{\max}(C)$. The construction of the chord representation C is done by a linear scan of B and is, thus, $O(U \times V)$. After this, a full computation of the lookup table $T_y(i,x,r)$ is performed once, which is either $O(W \times V \times \log_2(L_{\max}(C)))$ for S.E.s with relatively limited variation in chord lengths (such as a square or the letter H) and $O(W \times V \times L_{\max}(C))$ otherwise (such as a circle). Our algorithm performs for each line y of f the LineErode algorithm and updates the lookup table $H - 1$ times. Updating the table computes only one new line for the buffer and has, thus, a complexity of either $O(W \times \log_2(L_{\max}(C)))$ or $O(W \times L_{\max}(C))$ similar to the full computation of the table for $V = 1$. As the LineErode algorithm is $O(W \times N_C)$, the computational complexity of the eroding f with B is depending on the variation in the chord lengths for $N_f = W \times H$ either $O(U \times V + W \times V \times \log_2(L_{\max}(C)) + H \times W \times N_C + (H - 1) \times W \times \log_2(L_{\max}(C))) = O(N_f \times (N_C + \log_2(L_{\max}(C))))$ or $O(N_f \times (N_C + L_{\max}(C)))$; assuming in both cases that f is significantly larger than B.

The memory requirements are determined by the buffer that is used to store the minimum values, which is either $O(W \times V \times \log_2(L_{\max}(C)))$ or $O(W \times V \times L_{\max}(C))$.

G. Extensions

So far our discussion has been limited to erosions for 2-D images with a single S.E. The algorithm for computing dilations is obtained if we replace all min operators by max operators in the algorithm. For extension to 3-D images we need to augment the S.E. with a $z_{\min}$ and $z_{\max}$, each chord with a z-offset, and the lookup table $T_y(i,x,r)$ need to be replaced by $T_{y,z}i,x,r,d$, with d being the z-offset. Otherwise, the extension is straightforward.

If we want to perform the same operator with multiple structuring elements using our algorithm a small extra speed gain can be achieved if we realize that the buffer for the minimum (or maximum) values can be reused if we compute it for union of chord lengths present in all the S.E.s we want to use. Below, we will measure this small gain experimentally.

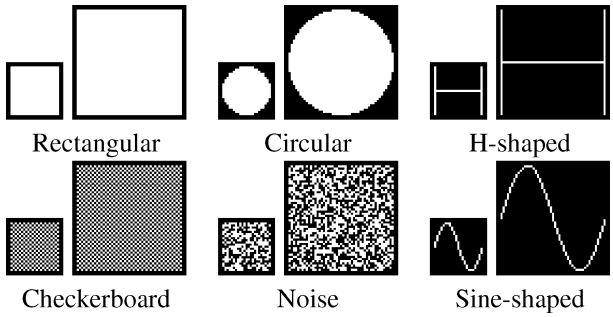

Fig. 2. S.E.s used in experiments; shapes shown are 23 and 49 pixels wide. Black borders were added to these illustrations here for clarity; these borders were not used in the experiments.

III. Experiments

We compared the computation time of our method (UW) with an optimized direct implementation, where only the foreground pixels of the S.E. are considered, and with the algorithm of DT [1] using rectangular, circular, H-shaped, checkerboard, noise, and sine-shaped structuring elements on two 8-bit 2160×1440 grayscale images: a natural image and a generated image with uniformly distributed noise. Examples of these S.E.s are shown in Fig. 2. All time necessary to compute a morphological operator (including eventual inverting or transposing the image but excluding reading and writing images) was considered part of the computation time of the method. As our and the DT method are intended as generic algorithms that should perform morphological operators with any arbitrary S.E. efficiently, all experiments discussed below were performed for these two algorithms with the same implementations, optimizations, and settings. Furthermore, no other processing (like transposing the image) were used. The other methods discussed here are specialized algorithms for which for example separately optimized implementations for horizontal and vertical linear S.E.s were used.

In order to measure the influence of the number of gray levels on the computation time, 16-bit versions of these two images were computed by scaling the gray values in both images. In the figures, DT8 and DT16 refer to respectively the 8-bit and the 16-bit implementation of the algorithm by Van Droogenbroeck and Talbot.

Figs. 3 and 4 show the computation times of these methods for a 2.8-GHz Pentium 4 processor based PC, with 1 GB of RAM. All methods were implemented in ANSI C without multithreading and compiled using gcc with its $-$O3 optimization flag set. A thin letter H, rectangular, circular, checkerboard, noise, and sine-shaped S.E.s of increasing width were used as

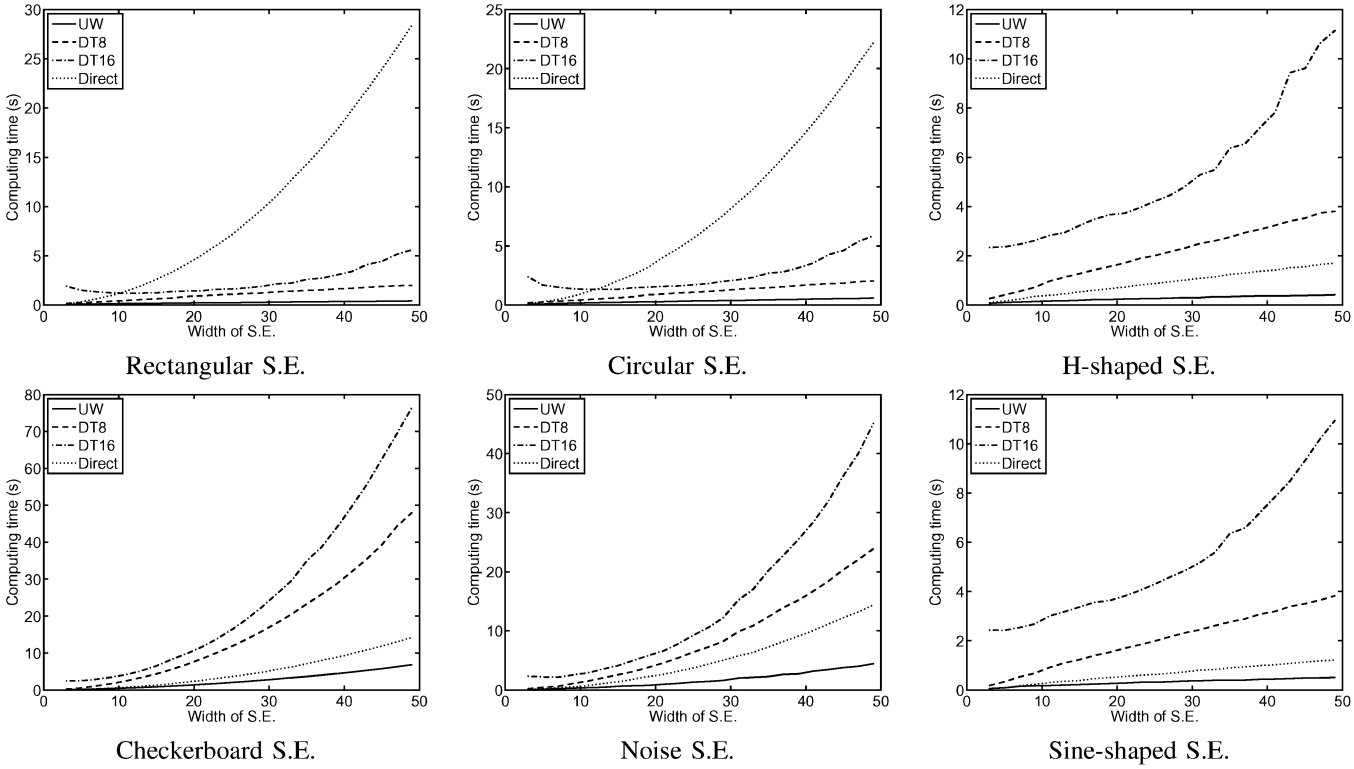

Fig. 3. Required computing time for erosions with width $l = k$. using the naive (Direct), the Van Droogenbroeck–Talbot (DT8 and DT16), and our proposed (UW) algorithm on a natural image.

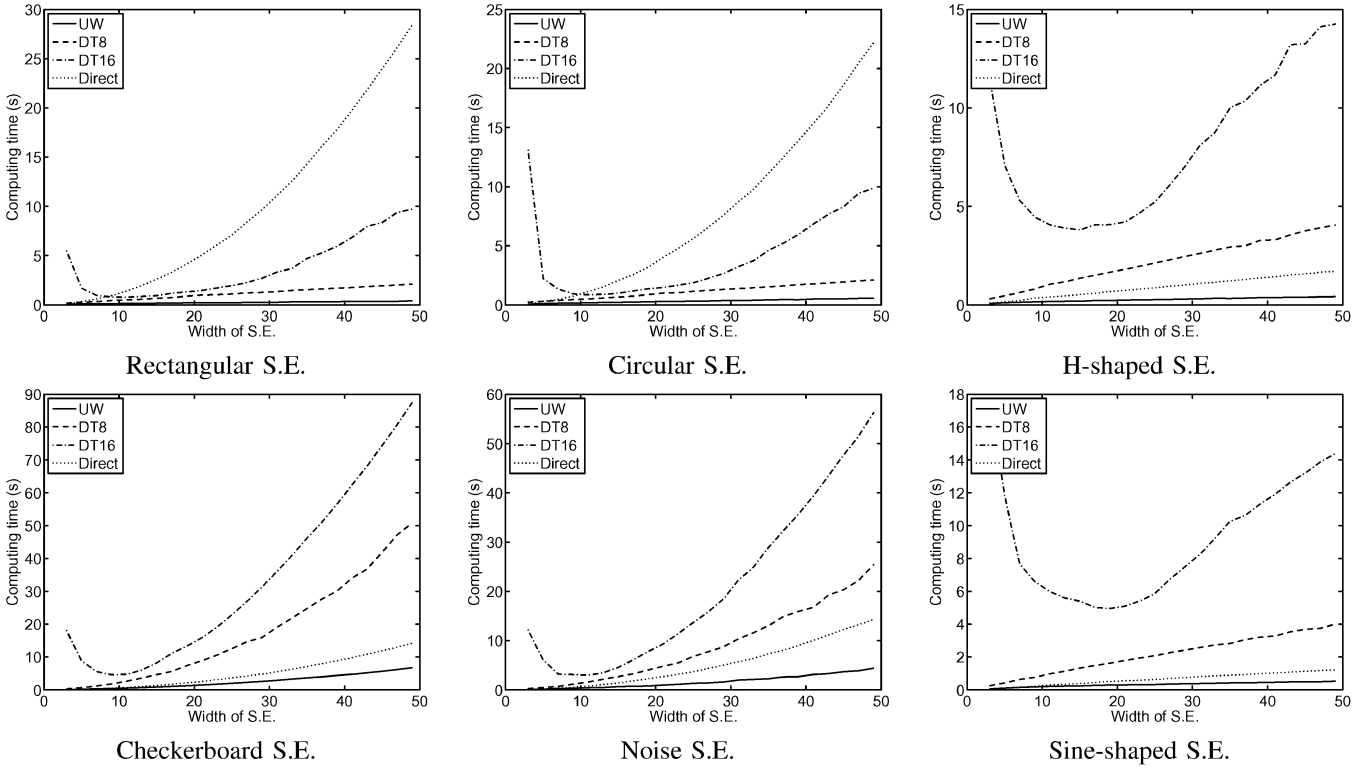

Fig. 4. Required computing time for erosions with width $l = k$, using the naive (direct), the DT (DT8 and DT16), and our proposed (UW) algorithm on a noise image.

S.E. shapes. The line segments of the H- and sine-shaped S.E.s were always 1 pixel thick. Note that the influence of image content (natural versus noise image) on the DT method is absent in our method.

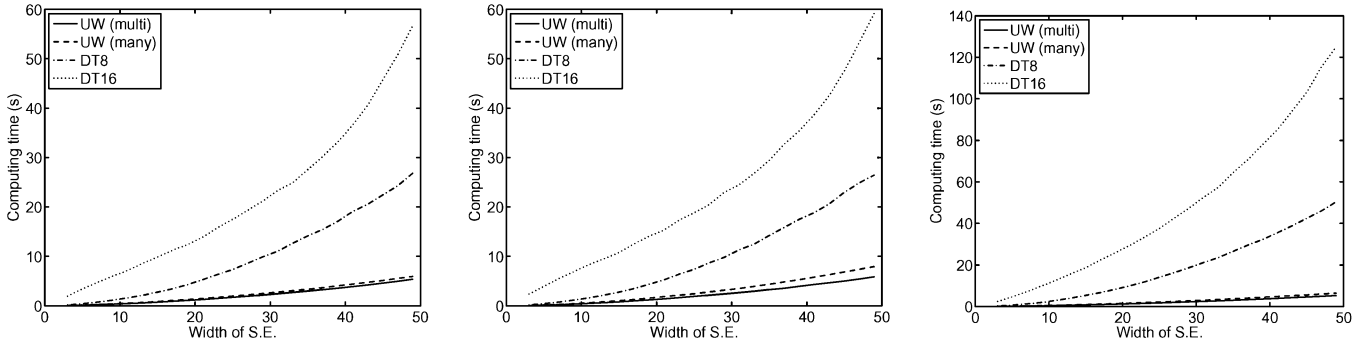

Fig. 5. Required computing time for multiple erosions with width $l = 3, 5, 7, \ldots, k$ using the DT (DT8 and DT16), and our proposed (UW) algorithm on a natural image.

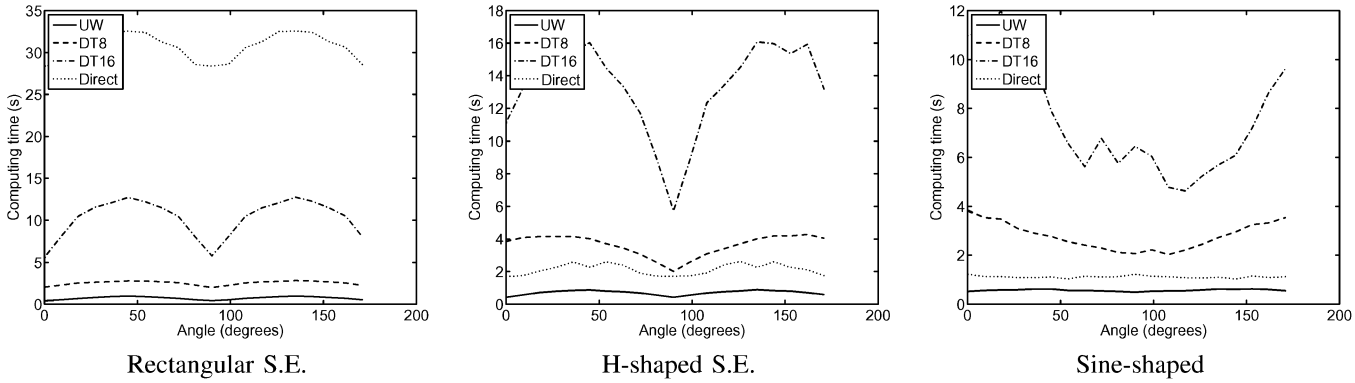

Fig. 6. Required computing time for erosions with S.E.s rotated at different angles using the naive (Direct), the DT (DT8 and DT16), and our proposed (UW) algorithm on a natural image.

TABLE I
COMPUTATION TIME IN SECONDS FOR ERODING A NATURAL AND A NOISE IMAGE USING A CIRCULAR S.E. OF DIAMETER 49, A 49 × 49 RECTANGULAR S.E., AND A 1-PIXEL-THICK LETTER H OF WIDTH AND HEIGHT 49

	circle		rectangle		H-shape	
	natural	noise	natural	noise	natural	noise
Direct	22.43	22.46	28.70	28.72	1.72	1.71
DT8	2.02	2.13	2.08	2.11	3.90	4.06
DT16	5.19	9.99	5.67	9.86	11.23	14.40
UW(ICIP)	1.15	1.15	1.06	1.05	3.31	3.50
UW(int)	0.58	0.55	0.43	0.41	0.43	0.41
UW(float)	0.90	0.87	0.67	0.64	0.68	0.66

From the figures, it is clear that the proposed method is always faster than the existing methods. The computation times for a circular, a rectangular, and a H-shaped S.E. of width 49 is shown in Table I. The computation times of our present algorithm is included for both the integer "UW(int)" and floating point "UW(float)" version and are compared with our preliminary version "UW(ICIP)" [18]. As can be seen, our present version has a factor 7.7 speed gain over our preliminary version for an erosion with an H-shaped S.E. of width 49. The computation time needed by our algorithm is independent of the bit depth of the image, but when floating point images are used instead of integer data, the computation time on the natural image with the same circular S.E. was 0.90 s: a speed penalty of 55% caused by the computational cost of using floating point operations instead of their integer versions. The Matlab image processing toolbox needed 165 s to erode the natural image with a circular S.E. of width 49 which is even much slower than the 22.43 s of our optimized direct implementation. Experiments have shown that the computation time of Matlab's erosion operator for linear S.E.s increases linearly with the length of the S.E.

As noted in the previous section, when an image is eroded with multiple S.E.s, our approach provides a way to compute these in less time than would be required when each of them would be computed separately. To measure the speed gain of this strategy [called "UW (multi)"], it was compared in Fig. 5 with the total time needed for computing the same erosions using the existing [1] (referred to as "DT8" and "DT16") and our method ["UW (many)"].

In Fig. 6, the computational cost of erosions with rectangular and H-shaped S.E.s of width 49 rotated to different orientations is shown. The variation in computation time can be explained by the fact that the number of chords and the chord lengths depend on the orientation. The computation time of performing an erosion with multiple S.E.s consisting of one shape rotated to many angles is shown in Fig. 7.

Soille *et al.* [11], [16] have shown how polygonal approximations of erosions with circular S.E.s can be computed efficiently. The computation times of their approach, which will refer to as the Soille algorithm, and our method are shown in Fig. 8. Surprisingly, even for circles of diameter 100 our method outperforms Soille's polygon approximation using 8 linear S.E.s. Furthermore, if d is the diameter of the circle, then the $O(d + \log d)$ behavior can be noticed for our method. Note that while

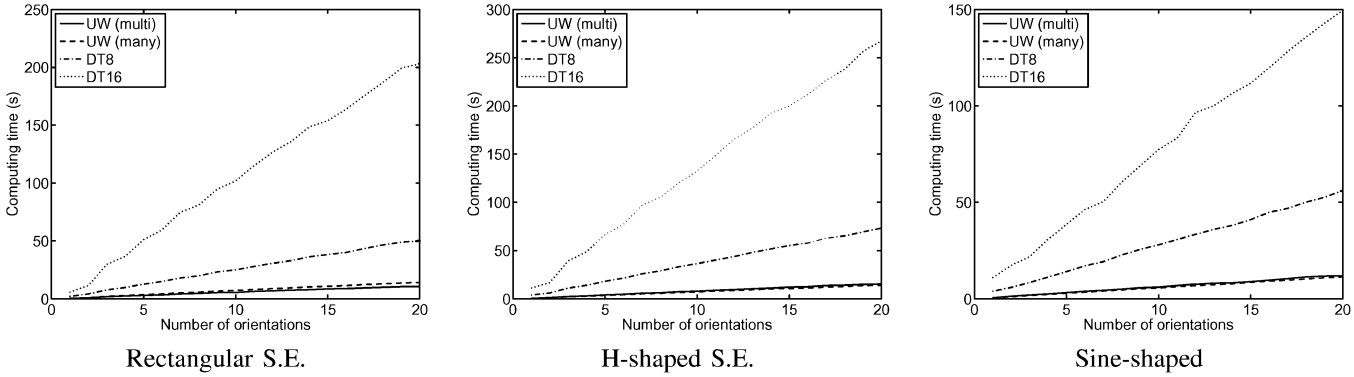

Fig. 7. Required computing time for multiple erosions with S.E.s rotated at k angles using the DT (DT8 and DT16), and our proposed (UW) algorithm on a natural image.

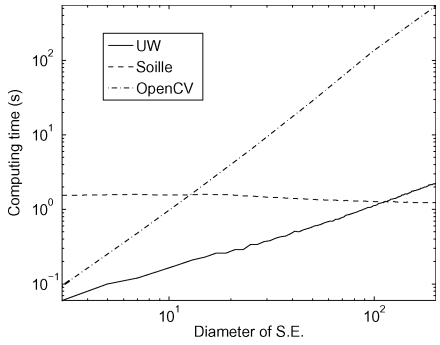

Fig. 8. Required computing time for erosions with circular S.E.s using polygonal approximations (Soille), openCV, and our proposed (UW) algorithm on the natural image.

a polygon approximation of a circle was used for Soille's approach, a "perfect" digital circle was used for our method. While we maintained the same number of linear S.E.s for each diameter of the polygonal approximation, one would desire to increase the number of lines as the diameter of the polygon increases, which in turn would increase the computation time of Soille's approach. The third method shown in the figure is the corresponding method from the openCV image processing library from Intel which is processor optimized. Note that here a logarithmic scale is used for both axes of the plot.

In Fig. 9, the computation time of our method is compared for linear S.E.s with the fastest existing algorithm dedicated for horizontal linear S.E.s by Gil and Kimmel [14]. The time needed to perform erosions with rectangular S.E.s is also shown. As the GK approach is to decompose a rectangular S.E. into a horizontal and a vertical linear S.E. we compared this with our proposed method using it both with square S.E.s ("UW 2-D") and the same squares decomposed in horizontal and vertical S.E.s ["UW (decomposed)"]. As is to be expected, our generalized approach is outperformed by this specialized algorithm, but only for horizontal and vertical S.E.s longer than, respectively, 31 and 63 pixels. The noticeable jumps in computation times for our method are caused whenever the number of chord lengths computed for lookup table is increased due to the increase in the length of the S.E. Note that this happens whenever the length of the maximum chord exceeds a new power of two.

Finally, our algorithm showed no change in computing time between different images of the same size. The apparent differences between the plots of the figures are due to different scales being used due to differences in computing time for the Droogenbroeck and Talbot algorithm.

IV. Conclusion

A new method for computing morphological operations with arbitrary 2-D flat structuring elements was proposed that is for S.E.s of width 49 between 2.0 and 7.7 times faster than our preliminary version. It has a computational complexity that is independent of the number of gray levels in the image. The proposed method has a clear computational performance advantage over existing methods when S.E.s are used that cannot be easily decomposed into linear structuring elements. Remarkably, our method is even faster for erosions with shorter linear S.E.s than dedicated algorithms (upto a length of 63 pixels for vertical S.E.s). A further, minor improvement is achieved when multiple S.E.s are used with a single operator, such as the computation of granulometries and dilation or erosion scale spaces [19], since the results of many comparisons are computed once and stored in an auxiliary array, which can be reused for filtering with all succeeding S.E.s. Note that for linear S.E.s the opening trees of Vincent are more suitable [21]

Like the existing method of DT [1] arbitrary S.E.s can be used. However, our method always outperforms the existing method, especially when images with higher bit depth, as common in applications such as medical imaging and astronomy, are used. Besides, our method can handle floating point images easily, which is impossible in the DT method. The DT method can, however, be easily adapted to other percentiles, which is impossible in ours.

Several existing open source and commercial image processing software, such as Matlab and openCV, were evaluated and it was found that most of them are based on a processor-optimized version of the naive algorithm. This probably explains why the old argument of morphological operators being slow still persists with many people within the image processing community despite the development of very efficient algorithms over the years.

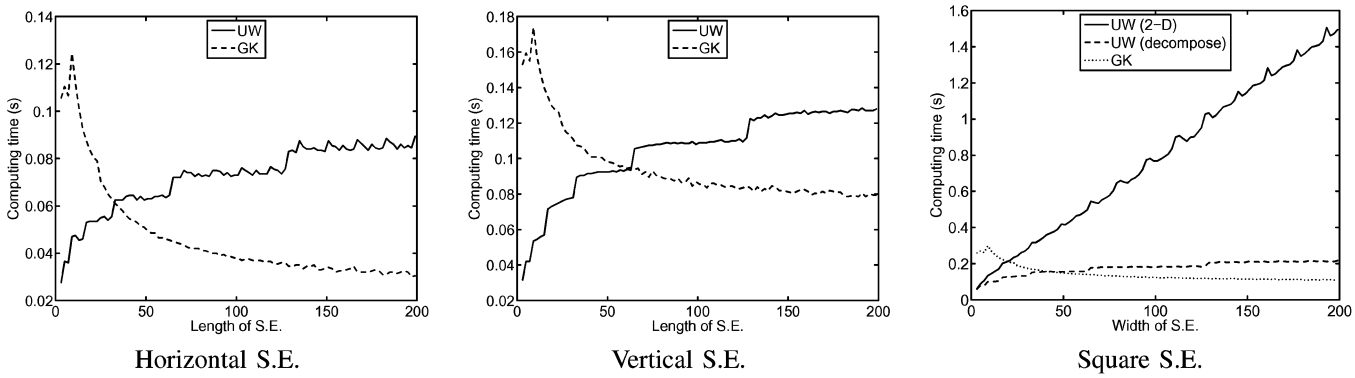

Fig. 9. Required computing time for erosions with linear S.E.s using the GK algorithm and our proposed (UW) algorithm on a natural image.

We are currently working on further speed improvements, such as the inclusion of the algorithm of GK for the computation of the chords for those cases where that would lead to a speed gain. Furthermore, we are studying its use for preprocessing 2-D and 3-D images and the application of template matching using the hit-or-miss transform with large numbers of S.E.s at many scales and orientations which can now be computed within practical time constraints.

REFERENCES

[1] M. Van Droogenbroeck and H. Talbot, "Fast computation of morphological operations with arbitrary structuring elements," *Pattern Recognit. Lett.*, vol. 17, pp. 1451–1460, 1996.

[2] J. Serra, *Image Analysis and Mathematical Morphology*, 2nd ed. New York: Academic, 1982, vol. 1.

[3] S. Batman and E. R. Dougherty, "Size distributions for multivariate morphological granulometries: Texture classification and statistical properties," *Opt. Eng.*, vol. 36, no. 5, pp. 1518–1529, May 1997.

[4] J. A. Moore, K. A. Pimbblet, and M. J. Drinkwater, "Mathematical morphology: Star/galaxy differentiation and galaxy morphology classification," Publications of the Astronomical Society of Australia, preprint.

[5] L. J. van Vliet and B. J. Verwer, "A contour processing method for fast binary neighborhood operations," *Pattern Recognit. Lett.*, no. 7, pp. 27–36, Jan. 1988.

[6] L. Ji, J. Piper, and J.-Y. Tang, "Erosion and dilation of binary images by arbitrary structuring elements using interval coding," *Pattern Recognit. Lett.*, vol. 9, no. 3, pp. 201–209, 1989.

[7] L. Vincent, "Morphological transformations of binary images with arbitrary structuring elements," *Signal Process.*, vol. 22, no. 1, pp. 3–23, 1991.

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

[9] E.-H. Liang and E. K. Wong, "Hierarchical algorithms for morphological image processing," *Pattern Recognit.*, vol. 26, no. 4, pp. 511–529, 1993.

[10] H. Park and R. T. Chin, "Decomposition of arbitrarily shaped morphological structuring elements," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 1, pp. 2–15, Jan. 1995.

[11] P. Soille, E. Breen, and R. Jones, "Recursive implementation of erosions and dilations along discrete lines at arbitrary angles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 5, pp. 562–567, May 1996.

[12] G. Anelli, A. Broggi, and G. Destri, "Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 2, pp. 217–224, Feb. 1998.

[13] N. Nikopoulos and I. Pitas, "A fast implementation of 3-d binary morphological transformations," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 283–286, Feb. 2000.

[14] J. Gil and R. Kimmel, "Efficient dilation, erosion, opening and closing algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1606–1617, Dec. 2002.

[15] M. Van Droogenbroeck and M. Buckley, "Morphological erosions and openings: Fast algorithms based on anchors," *J. Math. Imag. Vis.*, vol. 22, pp. 121–142, 2005.

[16] P. Soille and H. Talbot, "Directional morphological filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1313–1329, Nov. 2001.

[17] J. Darbon, T. Géraud, and A. Duret-Lutz, "Generic implementation of morphological image operators," in *Proc. Int. Symp. Math. Morphology*, 2002, pp. 175–184.

[18] E. R. Urbach and M. H. Wilkinson, "Efficient 2-d grayscale dilations and erosions with arbitrary flat structuring elements," in *Proc. Int. Conf. Image Processing*, 2006, pp. 1573–1576.

[19] P. T. Jackway and M. Deriche, "Scale-space properties of the multiscale morphological dilation-erosion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 1, pp. 38–51, Jan. 1996.

[20] A. R. Thompson, J. M. Moran, and G. W. Swenson, Jr., *Interferometry and Synthesis in Radio Astronomy, 2nd Edition*, 2nd ed. New York: Wiley, 2001.

[21] L. Vincent, "Granulometries and opening trees," *Fundam. Inf.*, vol. 41, pp. 57–90, 2000.

Erik R. Urbach (S'06–A'07) received the M.Sc. degree in computer science from the Institute of Mathematics and Computing Science, University of Groningen (RUG), Groningen, The Netherlands, in 2002, where he worked on the connected morphological operators for scale and shape spaces (C-MOSSS) project for the Ph.D. degree.

He is currently with the Lunar and Planetary Institute, Houston,TX, where he works on developing methods for identification and characterization of craters in images of Mars. The prime areas of his research are connected filters, multiscale and multishape analysis, image classification, and texture analysis.

Michael H. F. Wilkinson (M'99–SM'06) received the M.Sc. degree in astronomy from the Kapteyn Laboratory, University of Groningen (RUG), Groningen, The Netherlands, in 1993, after which he worked on the image analysis of intestinal bacteria at the Department of Medical Microbiology, RUG, and received the Ph.D. degree from the Institute of Mathematics and Computing Science (IWI), RUG, in 1995.

He was appointed as a Researcher at the Centre for High Performance Computing, RUG, where he worked on simulating the intestinal microbial ecosystem on parallel computers. During that time, he edited the book *Digital Image Analysis of Microbes* (Wiley, 1998) together with F. Schut. After this, he worked as a Researcher at the IWI on image analysis of diatoms. He is currently an Assistant Professor at the IWI.