

Dilation-run wavelet image coding

Zheng Wu · Ming-Yi He · Kai-Kuang Ma

Received: 12 June 2007 / Revised: 1 January 2008 / Accepted: 2 January 2008
© Springer-Verlag London Limited 2008

Abstract The run-length coding and the morphological representation are two classical schemes for wavelet image coding. The run-length coders have the advantage of simplicity by recording the lengths of zero-runs between significant wavelet coefficients but at the expense of yielding an inferior rate-distortion performance. The morphology-based coders, on the other hand, utilize the morphological dilation operation to delineate the clusters of significant coefficients for improving coding performance. In this paper, a novel *dilation-run* image coding algorithm is developed by taking the advantages of both schemes, in which the clustered significant coefficients are extracted by using the morphological dilation operation and the insignificant coefficients between the extracted clusters are coded by using the run-length coding method. The proposed dilation-run image coder is implemented in the framework of bitplane coding for producing embedded bitstreams. Compared with several state-of-the-art wavelet image coding methods, the proposed dilation-run image coding method achieves comparable rate-distortion coding performance, especially more attractive for fingerprint type of imageries.

Keywords Wavelet image coding · Run-length coding · Morphological dilation operation · Bitplane coding

1 Introduction

Since the re-introduction of wavelets in the mid-1980s, impressive advances in wavelet coding have been achieved. Some top-ranked wavelet image coders are Shapiro's embedded zerotree wavelet coder (EZW) [1], Said and Pearlman's set partitioning in hierarchical trees (SPIHT) [2], stack-run algorithm of Tsai et al. [3], morphological representation of wavelet data (MRWD) of Servetto et al. [4], significance-linked connected component analysis (SLCCA) of Chai et al. [5], and especially Taubman's embedded block coding with optimized truncation (EBCOT) [6]—the framework of the JPEG2000 image compression standard [7]. These coders have been proven to be successful in the field of image and video compression.

Among the wavelet coders mentioned earlier, the stack-run algorithm [3] is known for its simplicity by recording the lengths of *zero* (or *insignificant coefficient*) runs between adjacent *significant* coefficients; thus, relatively speaking, it requires less amount of computational time. The same general idea is introduced by Tian and Wells in their wavelet difference reduction (WDR) algorithm [8,9]. Other run-length coding schemes, such as adaptively-scanned wavelet difference reduction (ASWDR) of Walker et al. [10], context-modeled wavelet difference reduction (CM-WDR) of Yuan et al. [11] and cluster-based run-length (CRL) coder of Berghorn et al. [12], are also proposed to achieve higher coding efficiency. However, at the price of simplicity, the performances of these run-length coders are inferior to that of the state-of-the-art wavelet image coder, SPIHT [2].

Z. Wu · K.-K. Ma (✉)
School of Electrical and Electronic Engineering,
Nanyang Technological University,
Singapore 639798, Singapore
e-mail: ekkma@ntu.edu.sg

Z. Wu
e-mail: wwuuzz@hotmail.com

M.-Y. He
Department of Electronic and Information Engineering,
Northwestern Polytechnical University,
Xi'an 710072, People's Republic of China
e-mail: myhe@nwpu.edu.cn

Another kind of coding scheme worthy of notice is that based on the morphological representation, such as previously mentioned MRWD, SLCCA and other schemes including the energy clustering and zero-quadtree representation (ECZQR) algorithm by Zhong et al. [13] and the embedded morphological dilation coding (EMDC) algorithm by Lazzaroni et al. [14]. With the consideration that it might be expensive for zerotree coders to represent the arbitrarily shaped zero regions by the union of a highly constrained set of tree-structured regions, the morphology-based coders are designed to directly form the irregularly shaped non-zero regions (clusters of significant coefficients) through the morphological dilation operation [15,16] and their performances are nearly the same as or superior to SPIHT. For the morphology-based coders, a key problem is how to identify the seed for each cluster, i.e., the pixel from which a cluster is originated. Different solutions to this problem result in various morphology-based coding schemes [4,5,13,14].

In this paper, a novel *dilation-run* algorithm is proposed for wavelet image coding by taking the advantages of both the run-length coding and the morphological representation, in which the dilation operation is used to extract the clusters of significant coefficients, and the run-length coding is utilized to record the lengths of zero-runs between the extracted clusters. With the dilation operation used, most of the clustered significant coefficients are separated from the run-length coding process, and the efficiency of the run-length coding is thus improved. Furthermore, the run-length coding also implicitly provides the positional information of clusters. In summary, the key building blocks of the proposed dilation-run algorithm are discrete wavelet transform, bit-plane coding, morphological dilation operation (for extracting significant-coefficient clusters), run-length coding (for encoding insignificant coefficients between the extracted clusters) and arithmetic coding of symbols.

The remaining paper is organized as follows. In Sect. 2, after the run-length coding scheme is briefly introduced, some promising ways for improving the performance of the run-length coding are discussed. The morphological representation scheme for describing the clusters of significant wavelet coefficients is reviewed in Sect. 3. In Sect. 4, the proposed dilation-run idea is presented, and the details of the coding algorithm are described in Sect. 5. In Sect. 6, a set of performance comparisons among the proposed coding algorithm and other state-of-the-art wavelet image coders are given. The final section concludes the paper.

2 Run-length coding

A distinct feature of wavelet-transformed image is that most wavelet coefficients are either fairly small or zero in magnitude. After quantization, large numbers of zero-magnitude

coefficients are yielded. Thus, when designing a wavelet image coder, it is an important task to describe these zero coefficients compactly. The most well-known description scheme for zero wavelet coefficients is given by the zero-tree coders [1,2], in which the zero-valued coefficients associated with the same spatial region across all the subbands are organized in a tree structure, called the *zerotree*. Each zerotree can be compactly represented by a single symbol. In contrast with the zerotrees which are algorithmically complex to handle, a much simpler data structure for recording these zeros is provided by the *run-length coding* [3,8,9].

2.1 A review of relevant methods

The fundamental principle of the run-length coding is to record the lengths of zero-runs between two adjacent significant coefficients by using a compact representation. In the stack-run coding [3], the quantized wavelet coefficients in each subband are scanned in the raster order. Whenever a significant coefficient w is encountered, a message in the form of (a, b) is output: where a represents the number of zeros counted between the previous significant coefficient and the current significant coefficient w , and b consists of the magnitude m and the sign s of w . Both a and m are represented in binary and ordered from the least significant bit (LSB) to the most significant bit (MSB). Because of the need to distinguish between the runs of zeros and the values of significant coefficients, an alphabet set with four symbols, {"0", "1", "+" and "-"}, is used. The message (a, b) is then output as follows. First, the symbols "+" and "-" are sent for the binary representation of the zero-run length a , followed by the symbols "1" and "0" for the binary representation of the magnitude m , and finally, a symbol "+" or "-" for the sign s . This representation can be made even more compact, by exploiting the fact that the MSB of a nonzero binary number is always 1. Therefore, when sending the zero-run length a and the magnitude m in their binary notations, their MSBs can be omitted.

The similar idea of run-length coding is also exploited in the WDR algorithm [8,9], but in contrast, the latter provides an embedded coder, in which the zero-run length is transmitted by its binary representation, followed by the sign information of the significant coefficient, while the coefficient's magnitude is left to be gradually refined in later processes. Both the stack-run and the WDR coders are easy to implement and have low computational cost. However, their performances are inferior to the benchmark coder SPIHT as shown in Table 1.

2.2 Discussions and improvements

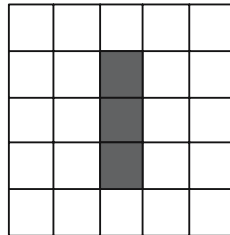
Intuitively, the efficiency of run-length coding will be affected by the presence of significant coefficients, for they interrupt

Table 1 A performance comparison (in PSNR [dB]) of three run-length coding schemes against the SPIHT using a 512×512 Lena image coded at three different bit rates (bits per pixel, bpp)

Algorithm/bit rate (bpp)	0.25	0.50	1.00
Stack-run [3]	33.63	36.79	–
WDR [8,9]	33.44	36.59	39.84
CRL [12]	34.01	37.09	40.28
SPIHT [2]	34.11	37.21	40.41

Note that the results of three run-length coders are quoted from the respective references

Fig. 1 A 5×5 subband contains *white* and *dark-gray* pixels, respectively, denoting the insignificant and significant coefficients



the consecutive runs of zeros. Given a subband as illustrated in Fig. 1, for example, the three significant coefficients (as shaded in dark gray) will affect the resultant run lengths, depending on the way of scanning operation. If the subband is raster scanned row-by-row, the zero-runs are frequently interrupted by the three significant coefficients, resulting in the run lengths 7, 4, 4 and 7. On the other hand, with a column-by-column raster scanning, the zero-run lengths are 11, 0, 0 and 11. Clearly, the latter requires lesser number of bits for representing the zero-run lengths in this particular example. This simple example shows that the run-length coder will be more efficient if longer zero-runs can be generated in the scan, which is also verified by the analysis and experiments documented in [12]. Motivated by this fact, various ways are proposed to improve the run-length coding as follows.

Instead of applying raster scan, a Hilbert-scanned path motivated by the property of the Hilbert curve is designed in the CRL algorithm [12] to exploit the correlations of direct neighbors in the wavelet domain. Along this path, more significant coefficients are clustered than those along a raster-scanned path, which leads to longer zero-runs. With further exploitation of the coefficient dependencies by using a 2×2 cluster structure of coefficients, the CRL coder achieves the performance close to that of SPIHT, as shown in Table 1.

In ASWDR [10] and CM-WDR [11], the scan procedure is adaptively adjusted by placing the coefficients with significant neighbors or significant parent (i.e., the coefficient located in the same spatial position at the last coarser scale under the similar orientation) in the head part of the scan order. By doing so, the “skewness” of the distribution of significant coefficients in the scan is increased, resulting in longer zero-runs and thus higher run-length coding efficiency.

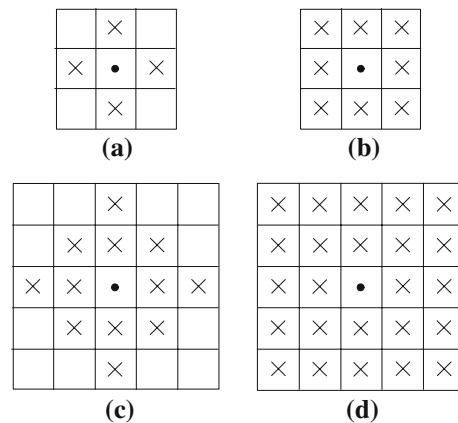


Fig. 2 Four possible structuring elements under consideration, which are denoted as **a** SE4, **b** SE8, **c** SE12, and **d** SE24, respectively. Note that the number “x” of SE x denotes how many pixels, except the central one, are included in the window mask

In this paper, a morphological approach is considered for improving the run-length coding.

3 Morphological representation of significant-coefficient clusters

The morphology-based coding methods [4,5,13,14] utilize the *morphological dilation operation* [15,16] to extract and represent the clusters of significant coefficients. In this study, the morphological dilation technique is considered for improving the run-length coding efficiency, based on the fact that large-magnitude wavelet coefficients from each subband tend to cluster at the spatial locations associated with the edges or textures in the original image. These clustered significant coefficients should be dealt with carefully in the run-length coding to achieve higher coding efficiency, while the morphological dilation operation has been proven as a good way for searching and extracting the clustered significant coefficients. In the following, the morphological dilation operation is briefly reviewed first. Then the cluster-representing procedure through the dilation operation is illustrated by an example.

Consider a set S to which the dilation operation will be applied, and let B represent a structuring element (such as those depicted in Fig. 2). Let \oplus denote the morphological dilation operator. The dilated set $S \oplus B$ is defined to be the union of all points falling within the support of the structuring element B , when this structuring element is centered at each point in S . Further, denote the set of the points obtained by the dilation operation and not in S as $S \oplus B \setminus S$, where the symbol \setminus means set-theoretic difference.

If S is a subset of a cluster of significant wavelet coefficients within a subband, then its neighboring coefficients

in the set $S \oplus B \setminus S$ will have a high probability of being significant, according to the clustering tendency of large-magnitude wavelet coefficients. Hence, by seeking significant coefficients in the set $S \oplus B \setminus S$, a new set containing S and its neighboring significant coefficients can be formed. If such region-growing operation is recursively applied, the whole cluster can be progressively constructed from the original S . In this process, the only insignificant coefficients involved are those located at the boundary of the cluster.

A graphical illustration for the afore mentioned cluster-growing procedure is given in Fig. 3, in which the 8-connected structuring element SE8 in Fig. 2b is used. Figure 3a shows the significant coefficients (in black) clustered within a subband. Assume that the significant coefficient shown in Fig. 3b is already known and thus identified as a *seed*, from which the whole cluster will be constructed through the *recursive dilation operation* (corresponds to the pseudocode algorithm, **Recursive-dilation-encode**, at the end of Table 8). In Fig. 3c, the dilation operation is first applied to the seed. Then in the remaining steps as shown in Figs. 3d–h, the dilation operations are applied to *each* of the new significant coefficients found in the previous steps. The significance states of the coefficients involved in each step are determined and output (the magnitudes and signs of the coefficients are not considered in this example for simplicity). Finally, the recursive dilation operation stops when no more new significant coefficients are found, i.e., the entire significant-coefficient cluster has been extracted.

As shown in the example, a seed should be first identified for extracting each significant-coefficient cluster located within a wavelet-transformed image. Therefore, how to record the positional information of the seeds is a key step for the morphology-based coders. In MRWD [4], all the wavelet coefficients are coded regardless of their states of significance. The seed of each cluster is specified by transmitting a special symbol. In SLCCA [5], a seed position is encoded by its spatial coordinates or inferred from its significant parent by a LINK symbol which signifies that one child is the seed of a cluster. ECZQR [13] utilizes zero-quadtree decomposition method to locate the seeds. EMDC [14] not only exploits explicit position coding and parent–child prediction to identify the seeds, but also introduces a boundary morphological dilation operation to search the seeds along the outer boundaries of existing clusters.

4 Dilation-run coding

4.1 Intuition and basic idea

The proposed *dilation-run* coding procedure is developed by combining the morphological dilation operation with the run-length coding. For each subband of a wavelet-transformed

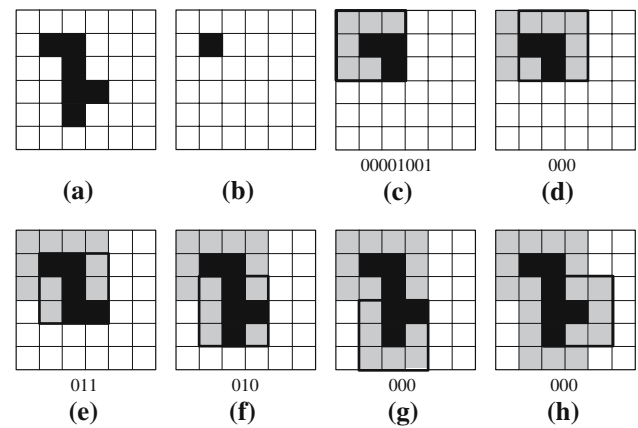


Fig. 3 A demonstration of the significant-coefficient cluster formation by recursively applying the dilation operation on a 6×6 subband. **a** is the significance map, where the *black* and *white* pixels denote significant and insignificant coefficients, respectively. **b** shows the seed. **c–h** show how the cluster is grown by applying the dilation operator with the structuring element SE8 being centered at each significant coefficient. In these steps, the *white pixels* denote the coefficients that have not been visited, and the *black* (symbol “1”) and *gray* (symbol “0”) pixels separately denote the encoded significant and insignificant coefficients. The *symbols* below each figure are the bits output at that step. Note that in each step only those bits corresponding to the new coefficients falling within the current SE8 window are output (in the raster-scan order over the SE8). Therefore, only the first step in **c** outputs 8 bits, while additional 3 bits for each remaining one

image, coefficients are scanned in a predefined order (such as raster scan) for encoding. When a significant coefficient is encountered, the number of insignificant coefficients unencoded before it is counted and recorded as the way conducted in the run-length coding; consequently, the position of a seed is precisely identified. The dilation operation is then recursively applied to the seed and to each significant coefficient newly found during the dilation process (if any), just as the steps demonstrated in Fig. 3. Once all the significant coefficients of the cluster associated with the current seed are encircled, the recursive dilation process stops and the same scanning process is resumed on those unscanned coefficients, until the entire subband coefficients are exhaustively encoded.

Up to this point, most of the clustered significant coefficients are extracted by using the morphological representation, such that longer zero-runs can be generated for benefiting the run-length coding process. On the other hand, the run-length coding also provides an efficient way to identify the seed’s position for each significant-coefficient cluster. The advantages of the dilation-run coding idea are further illustrated by the following experiment.

4.2 Experimental studies of the dilation-run coding

A five-level wavelet decomposition using Daubechies 9/7 filter [17] is applied to a test image Lena (512×512). Then

Table 2 A detailed comparison of the coding efficiency of the run-length coding scheme versus the dilation-run coding scheme using Lena (512×512)

Index	Attribute	Run-length coding	Dilation-run coding			
			SE4	SE8	SE12	SE24
1	Total number of bytes	8127	7414	7531	7670	7895
2	Number of coefficients encoded by the run-length coding	262144	236534	229448	223132	213112
3	Number of zero-runs	15638	3011	2054	1555	1085
4	Number of run-length symbols	33999	12610	9570	7607	5326
5	Run-length representation efficiency, I_r	7.71	18.76	23.98	29.33	40.01
6	Entropy H_r	1.9853	1.7901	1.7478	1.7260	1.7231
7	Number of dilation symbols	–	25610	32696	39012	49032
8	Entropy H_d	–	1.4928	1.3947	1.3044	1.1741

The dilation-run coding scheme is tested with four structuring elements

a signed significance map is generated for the transformed Lena image with a threshold value 16. That is, for the wavelet coefficients with magnitudes less than 16, they are set to 0, while the others are set to 1 or -1 according to their signs. In the following experimental studies, three coding schemes—(1) the run-length coding, (2) the morphological representation, and (3) the dilation-run coding, are, respectively, conducted to code the signed significance map. Here, only the significance map and the signs of significant coefficients are coded, because the proposed dilation-run coding is mainly for providing an efficient way to identify the positions of significant coefficients.

1. In the run-length coding scheme, each subband is scanned in the raster order. When a significant coefficient is encountered, the length of the zero-run before it is output in binary representation (using the symbols “0” and “1”), followed by its sign symbol (either “+” or “-”). For the zero-run length, the MSB of its binary representation is omitted, while for the run of length whose binary representation is composed by all 1 bits (i.e., the length equals to $2^k - 1$ and k is a positive integer), it is necessary to retain the MSB. This is due to the fact that the runs with length one would not be representable, if all the MSB symbols are eliminated.
2. In the morphological representation scheme which exploits the basic idea of MRWD [4], each subband is also scanned in the raster order, and the coefficients are encoded one by one by using the symbols “0”, “+” and “-”, which correspond to insignificant coefficient, significant positive coefficient and significant negative coefficient, respectively. When a significant coefficient is found, the recursive dilation process is applied, and for each coefficient involved in the process it is also encoded by using the symbol “0”, “+” or “-”. After the current recursive dilation process terminates, the same scanning

process is resumed on those unscanned coefficients until the entire subband is encoded.

3. In the proposed dilation-run coding scheme, a combination of the run-length coding and the morphological representation is exploited, such that when a significant coefficient is found, the un-encoded zeros before it and its sign are encoded in the same way as that in the run-length coding scheme, followed by invoking the recursive dilation process to fulfill the morphological representation of the clustered significant coefficients.

In all the three coding schemes afore mentioned, the *adaptive arithmetic coder* [18] is adopted for conducting the entropy coding. In the run-length coding, one probability table of the symbols is established for driving the arithmetic coder, while two probability tables are used for the morphological representation scheme—one for the symbols generated during the scan and the other for the symbols produced during the dilation process. In the dilation-run coding scheme, two probability models are used, one for the symbols generated during the run-length coding (called the *run-length symbols*), and the other for the symbols yielded during the dilation process (called the *dilation symbols*). The morphological representation scheme and the dilation-run coding scheme were experimented with all the four structuring elements in Fig. 2. The experimental results are documented in Tables 2 and 3.

For a detailed comparison between the run-length coding scheme and the dilation-run coding scheme, the results are reported in Table 2 on several aspects—the total number of bytes output by each coder, the numbers of coefficients encoded by the run-length coding, the numbers of zero-runs generated during the scan, and the numbers of the run-length symbols and the dilation symbols with their respective entropies H_r and H_d . In Table 2, the factor I_r is the measurement of the representation efficiency for each run-length symbol,

Table 3 A comparison of the coding efficiency of the morphological representation scheme versus the dilation-run coding scheme using Lena (512×512)

Method	SE4	SE8	SE12	SE24
Morphological representation	7525	7565	7687	7924
Dilation-run coding	7414	7531	7670	7895

Both schemes are tested with four structuring elements and the numbers of bytes yielded in each case are listed

which is defined as the average number of the coefficients represented by each run-length symbol and is computed as

$$I_r = \frac{\text{No. of coeffs. coded by the run-length coding}}{\text{No. of the run-length symbols}}$$

A detailed comparison between the morphological representation scheme and the dilation-run coding scheme is given in Table 3, where the number of the bytes yielded in each case is listed.

4.3 Analysis and comments

In this sub-section, insightful remarks are made regarding the simulation results documented in Tables 2 and 3, so that the proposed dilation-run coding scheme can be better appreciated.

First, Table 2 shows that the dilation-run coding scheme always outperforms the run-length coding scheme by yielding less total number of bytes (given in row 1) to encode the generated significance map (consisting of 262,144 pixels for the 512×512 Lena image), which can be explained as follows. In the dilation-run coding scheme, a large portion of clustered significant coefficients are extracted by using the dilation operation. Consequently, with about 10 ~ 20% coefficients being separated from the run-length coding process (as shown by row 2), much less number of zero-runs (given in row 3) is generated, thus leading to longer zero-run lengths and less run-length symbols (given in row 4). Figure 4 verifies that with the dilation operation exploited, the number of the zero-runs with longer length is increased. A longer length means a more compact representation, because with each additional bit added in the binary representation, the representation range will be doubled. Therefore, each run-length symbol in the dilation-run coding scheme represents more coefficients than that in the run-length coding scheme, which can be verified through the measurement of I_r (given in row 5, which is obtained by dividing each entry in row 2 by the corresponding entry in row 4). Furthermore, the table shows that the entropy of the run-length symbol H_r in the dilation-run coding (given in row 6) is much less than that yielded in the run-length coding. This is also due to the longer zero-runs generated by the dilation-run coding scheme, which results

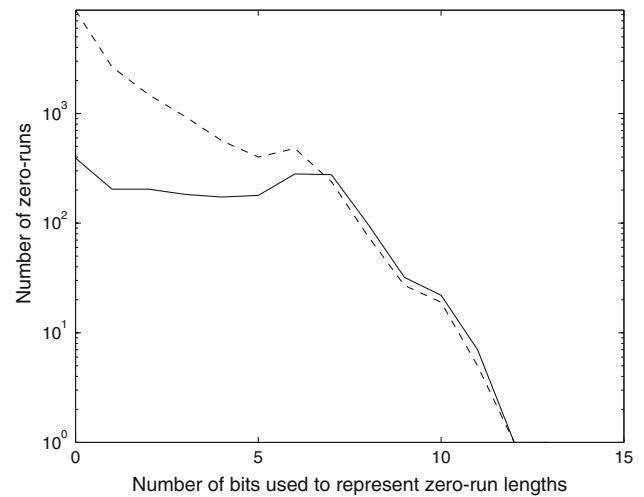


Fig. 4 The distributions of zero-runs with respect to zero-run lengths, which are measured by the number of bits needed for representing their binary notations. (The vertical axis is in the logarithm scale.) For the zero-run length not equal to $2^k - 1$ (k is a positive integer), the MSB will be dropped when calculating the number of bits needed for representing it. The *solid curve* represents the dilation-run coding scheme using the structure element SE8, whereas the *dashed curve* denotes the run-length coding scheme. Note that the *solid curve* is above the *dashed curve* for the zero-run lengths with numbers of bits ≥ 7 , which means that the dilation-run coding scheme yields more long zero-runs than the run-length coding scheme

in that the distribution of the run-length symbols is much more skewed to the symbols “0” and “1” for representation of the zero-run lengths, and thus smaller entropy.

In the dilation-run coding scheme, the price paid for improving the run-length coding efficiency is that the coefficients which have high probability of being significant are coded through the dilation operations, and a dilation symbol is output for each involved coefficient (so the sum of the entry in row 2 and the entry in row 7 for each column is 262,144). However, the entropy H_d of the dilation symbols (given in row 8) is always smaller than that of the run-length ones, which is mainly due to the fact that the dilation symbols are 3-valued (“0”, “+” and “-”) and the run-length symbols are 4-valued (“0”, “1”, “+” and “-”). Hence, the dilation-run coding scheme finally outputs less number of bytes than the run-length coding scheme.

Table 2 also shows that the dilation-run coding scheme generates less bytes by using a smaller structuring element (as listed in row 1), which is explained as follows. When applying the dilation operation to a significant coefficient, the coder indeed makes the prediction that the coefficients in its neighborhood may be significant too and thus code them through the dilation operation. By using a small structuring element, only the coefficients which are very close to the significant coefficient are considered to be possibly significant and thus small amount of coefficients are involved in the dilation process. On the other hand, with a large structuring element,

Table 4 Algorithm of the bitplane coding**Bitplane-encode()**

1. Output $n_{\max} = \lfloor \log_2(\max_{(i,j)} |w_{i,j}|) \rfloor$, i.e., the index of the most significant bitplane. Set $n = n_{\max}$.
2. Encode the n th bitplane. Scan through the coefficients.
 - (a) **Dominant pass:** For the coefficient with $|w_{i,j}| < 2^{n+1}$, evaluate its significance state for the current bitplane: if $|w_{i,j}| \geq 2^n$, output “1” and its sign; otherwise, output “0”.
 - (b) **Refinement pass:** For the coefficient with $|w_{i,j}| \geq 2^{n+1}$, output its n th most significant bit.
3. Decrease n by one for encoding the next bitplane, and go to Step 2.

the coefficients in a large neighborhood are predicted to be significant and thus large amount of coefficients are involved in the dilation process. As the dependency between two wavelet coefficients declines when the distance of their locations increases, choosing a larger structuring element means that a worse prediction could be made in the dilation operation, that is, more zero coefficients will be coded in the dilation process. Although run-length coding efficiency is improved by using a larger structuring element (shown by row 4 to row 6), the price paid is that many more dilation symbols are output (given in row 7). Thus the total numbers of the coding bytes show that the coding performance decreases as the size of the structuring element becomes large.

In comparison with the morphological representation scheme such as MRWD [4], the dilation-run coding scheme has the following advantages. First, the coding complexity is much reduced by introducing the run-length coding. Table 2 clearly shows that in the dilation-run coding scheme, the amount of the run-length symbols is much lesser than the number of the coefficients encoded in the scan. In contrast, one symbol is needed for encoding each coefficient in the morphological representation scheme. Secondly, the exploitation of the run-length coding tends to slightly improve the coding efficiency, as indicated in Table 3. When compared with the seed identification methods used by other morphology-based coders [5, 13, 14], the run-length coding method adopted by the dilation-run coding scheme is also advantageous on efficiency and simplicity. For example, the quad-tree decomposition method used by ECZQR [13] is complex, whereas the coordinate recording method used by SLCCA [5] and EMDC [14] is simple, but inefficient in representing seed positions.

5 Coding algorithm

Based on the dilation-run idea, the details of the proposed coding algorithm are given in this section. The key components—the *bitplane coding* and the *context-based entropy coding*—are elaborated. The bitplane coding outputs the embedded bitstream, while the context-based entropy coding makes the output bitstream more compact. At the end of this

section, the whole dilation-run coding algorithm is summarized in pseudocode.

5.1 Bitplane coding

5.1.1 Embedded coding

Embedded coding refers to generating a *scalable* bitstream for the encoded source image, such that all lower bit rate codes are embedded at the beginning of the bitstream [1]. This feature is required for progressive transmission and highly desirable for diversified multimedia applications (such as image browsing). A widely-used approach for achieving embedded coding for wavelet transformed image is the *bitplane coding*, in which the coefficients are sequentially described by bitplanes—from the most significant bit to the least significant one [1], [2]. Table 4 describes the general procedure of the bitplane coding, where the wavelet coefficient at the location (i, j) is denoted as $w_{i,j}$.

The bitplane coding is a successive process for gradually approximating or building up the original wavelet-transformed image, by using a set of successive quantization stepsizes $\{2^{n_{\max}}, 2^{n_{\max}-1}, \dots, 2^n, 2^{n-1}, \dots\}$. Given the n th bitplane, a coefficient $w_{i,j}$ is considered *significant* at this bitplane if $|w_{i,j}| \geq 2^n$; otherwise, *insignificant*. The coding process of the n th bitplane includes (1) the *dominant pass*—updating the significance map of the coefficients with respect to the new threshold 2^n . That is, for the insignificant coefficients of the $(n+1)$ th bitplane, their significance states need to be checked to identify new significant coefficients; (2) the *refinement pass*—refining the magnitudes of the significant coefficients identified on all the previous bitplanes with the indexes $\{n_{\max}, n_{\max}-1, \dots, n+1\}$, with respect to the new stepsize 2^n , by outputting their n th most significant bits. As the bitplanes are successively processed, a symbol stream is generated with the property that a sequence of image reconstructions approaching to the original image can be obtained from the stream, and any initial substream corresponds to a lower-resolution reconstruction. Hence, the bitplane coding procedure can be terminated at any desired target bit rate.

During the process of the bitplane coding, the symbol streams generated in the dominant pass are highly dominated by “0”s, because most wavelet coefficients are small in their magnitudes. For the small quantity of “1”s generated in this pass, which correspond to the newly identified significant coefficients, their positions tend to be highly correlated, due to that the significant wavelet coefficients tend to cluster in each subband. Previous analysis in the study has shown that the dilation-run coding process is advantageous in describing this kind of information more compactly and simply. The next subsection will show in detail how the dilation-run coding process is implemented within the framework of the bitplane coding.

5.1.2 Cluster updating in the dominant pass

For exploiting the dilation-run coding idea in the dominant pass, the significant coefficients newly appearing on each bitplane need to be considered from the viewpoint of *clusters* (rather than one coefficient at a time). This is because the main objective of the dilation process in the dilation-run coding is to separate the clustered significant coefficients from the run-length coding.

Among the significant coefficients newly appearing on each bitplane, many of them locate nearby the clusters formed on the previous bitplane; hence, the clusters formed on the previous bitplane tend to be further expanded on the current bitplane. Besides expanding the existing clusters, *new* clusters are possibly formed on the current bitplane, based on the new seed positions. According to the inter-band dependencies among wavelet coefficients, the positions of some new seeds can be traced from their significant parents. For the seeds without significant parents, their positions need to be identified by using the run-length coding process.

On the basis of the previous considerations, the dominant pass in the dilation-run coding algorithm is further split into three passes—the first one is for expanding the existing clusters, the second one is for identifying new clusters via the relationship of parent and children, and the final one is for seeking isolated new clusters by using the dilation-run coding procedure.

5.1.3 Coding passes over bitplanes

With three passes evolving from the dominant pass and the magnitude refinement pass, each bitplane is sequentially processed through four passes as follows.

1. Intra-band cluster growing pass: In this pass, the recursive dilation process is applied to each significant coefficient identified on all the previous bitplanes, for expanding the already-existing clusters of significant coefficients. For each coefficient involved in the recursive dilation process, its state is checked first. If it is insignificant on the previ-

ous bitplane and is not coded yet at the current bitplane, a *significance symbol* “1” or “0” is sent for it to signify whether it is significant on the current bitplane; if significant, its *sign symbol* “+” or “−” is also sent.

2. Inter-band cluster growing pass: During this pass, the parent–child dependency among the wavelet coefficients is exploited to predict the presence of *new* clusters. For that, the seeds of new clusters are sought as follows. For each significant coefficient identified on all the previous bitplanes, its four children are checked. If there is any child which is insignificant on the previous bitplane and is not yet coded on the current bitplane, a *significance symbol* “1” or “0” is sent for it to signify whether it is significant on the current bitplane; if significant, its *sign symbol* “+” or “−” is also sent and a new seed is found. Then the recursive dilation process is applied to the seed to form the whole cluster.

3. Magnitude refinement pass: For each significant coefficient identified on all the previous bitplanes, a *refinement symbol* “1” or “0” is sent according to its bit on the current bitplane.

4. Isolated cluster seeking pass: Here, the dilation-run coding procedure is utilized to extract the *isolated new* clusters which have not been coded in the second pass. For each subband, the un-coded coefficients are one-by-one scanned for seeking the seeds of new clusters. When a seed is found, the number of un-coded zeros before it is sent in binary notation by using the *run-length symbols* “0” and “1”, followed by the sign of the seed represented in the *run-length symbol* “+” or “−”. Then the recursive dilation process is applied to the seed. Once the whole cluster is formed, the scanning is resumed on those un-coded coefficients. After the scanning of each subband is finished, a *run-length symbol* “E” is sent to signify that no additional significant coefficients are found and thus all the coefficients within the current subband have been processed. An example of this dilation-run coding procedure is shown in Fig. 5, where the structuring element SE8 is used for the dilation operation.

Each step of Fig. 5 is explained as follows. The subband is scanned in the raster order. In step (b), two (with the binary notation 10) zeros are counted before the first significant coefficient, so a run-length symbol “0” is sent for recording the zero-run length (with the MSB omitted), followed by the run-length symbol “+” for representing the sign of the significant coefficient. In step (c), the dilation operation is applied to the seed identified in (b), and a significance symbol is output for each coefficient involved in the 3×3 window of the structuring element SE8 (except the coefficients which have been coded in the previous step). As no significant coefficient is found in (c), scan is resumed for the remaining coefficients. In step (d), another significant coefficient is found after 13 (with the binary notation 1101) un-coded zeros are counted. Then the zero-run length and the sign of the significant coefficient are coded in the same

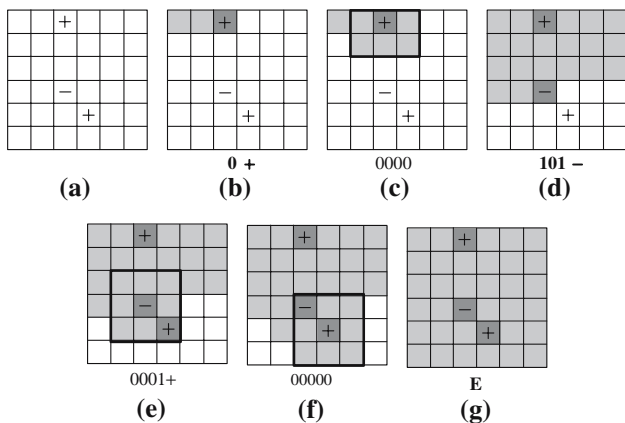


Fig. 5 Illustration of the dilation-run coding on a 6×6 subband. **a** is the significance map, in which the significant coefficients are labelled by their signs. **b–g** are the sequential steps of the dilation-run operation, where the *white pixels* denote the coefficients that are not coded, and the *dark-gray* and *light-gray* pixels denote the coded significant and insignificant coefficients, respectively. The stream of symbols below each figure corresponds to those transmitted at that step. The symbols generated in the run-length coding, i.e., the symbols transmitted in **b**, **d** and **g**, are represented in *bold face*, whereas the symbols produced in the dilation operations, i.e., the symbols transmitted in **c**, **e** and **f**, are represented in *non-bold face*

way as that in step (b). In step (e), the dilation operation is applied to the seed identified in (d), from which a new significant coefficient is found. So, its sign symbol is also output and the dilation operation is applied to it in step (f). In step (g), scan is resumed again. As there is no further significant coefficient found, a run-length symbol “E” is finally sent.

To further improve the efficiency of the isolated cluster seeking pass, a scan path designed according to the coefficient correlations is employed, rather than the simple raster scan used in the example. First, the scan path is aligned with the subband orientation. For the *LL* (lowpass) subband and the *LH* (vertically highpass) subbands in which the correlation along the horizontally adjacent coefficients is more pronounced, a row-by-row scan is used. For the *HL* (horizontally highpass) subbands in which the correlation along the vertically adjacent coefficients is more distinct, a column-by-column scan is used. For the *HH* (diagonally highpass) subbands in which the coefficients have strong correlations in the diagonal directions, a column-by-column scan is still used, due to that the computation incurred by exploiting a zigzag scan path along the diagonal direction is much more than that incurred by using the raster scan. Additionally, with the consideration that the clusters of significant coefficients tend to occur in the same region, each subband is partitioned into 16×16 blocks. All the blocks within each subband are scanned in the row-by-row or column-by-column manner as mentioned earlier. Within each block, the coefficients are again scanned in the same manner.

Besides the descriptions of the passes given above, the detailed coding operations in each pass will be outlined in Table 8.

5.2 Context-based entropy coding

The symbols generated by the proposed coding algorithm are further entropy-coded for yielding a more compact bitstream. Four types of symbols are produced by the proposed coding algorithm: the *run-length* symbols, the *significance* symbols, the *sign* symbols and the *refinement* symbols. Except for the run-length symbols which come from a five-valued set {“0”, “1”, “+”, “-”, “E”}, the other three types of symbols are all binary-valued. These symbols are coded by using the context-based adaptive arithmetic coder [18].

For a sequence of symbols, x_1, x_2, \dots , and x_M , its minimum code length

$$L = -\log_2 \prod_{i=1}^M P(x_i|x^{i-1})$$

is obtainable with an arithmetic coder driven by the probability model $P(x_i|x^{i-1})$. Here, x^{i-1} denotes $\{x_{i-1}, x_{i-2}, \dots, x_1\}$, which is called the *context* of x_i . In practice, it is difficult to directly obtain a good estimation of $P(x_i|x^{i-1})$, because x^{i-1} changes with the output symbol sequence and contains a large set of symbols. Thus, the key issue of context-based entropy coding is to map x^{i-1} to a context set \underline{x}^{i-1} , which consists of only a few cases, and the estimation of $P(x_i|x^{i-1})$ then changes to the estimation of $P(x_i|\underline{x}^{i-1})$. Of course, $P(x_i|\underline{x}^{i-1})$ should be a good approximation of $P(x_i|x^{i-1})$, so the context \underline{x}^{i-1} usually consists of a small number of most relevant symbols regarding x_i .

The context sets for the four types of symbols generated in the proposed algorithm are separately introduced in the following paragraphs. For each type of symbols, the various cases of contexts are labeled with 0, 1, 2, ... for a simple notation. Then for each symbol being output, the arithmetic coder will utilize the probability model corresponding to its type and context label to perform entropy coding.

1. Contexts for the run-length symbols: Here, six contexts are used. Table 5 defines the assignment rules of these contexts. In the alphabet set of run-length symbols, i.e., {“0”, “1”, “+”, “-”, “E”}, let symbol “R” denote “1” and “0” (for recording the zero-run lengths), and symbol “S” denote “+” and “-” (for representing the signs of the seeds). Then for the run-length symbol which is being coded, its context is decided by the number of the “R” symbols sent between it and the last “S” symbol sent before it, as shown in Table 5. For the first run-length symbol sent in each subband, its context label is set to 0.

These contexts are assigned according to the fact that a sequence of “R” symbols (which is the binary representation

Table 5 Assignment of the contexts for the run-length symbols

Number of the “R” symbols	Context label
0	0
b_{\max}	1
$b_{\max} - 1$	2
1 ~ 2	3
3 ~ 4	4
≥ 5	5

The context is decided by the number of “R” symbols between the current run-length symbol and the last “S” symbol before it. The contexts are not mutually exclusive; thus, the contexts are evaluated sequentially, and the first one which satisfies the situation of the symbol being coded will be selected

of a zero-run length) is always ended by an “S” symbol (which signifies the significant coefficient interrupting the zero-run, i.e., a seed). Therefore, with more “R” symbols being transmitted, the next “S” symbol is to be incurred more possibly. The length of “R”-symbol sequence has an upper limit, i.e., the factor b_{\max} in Table 5. Each time before a zero-run length is sent, the number N of all the un-coded coefficients left in the subband is counted. Then b_{\max} is computed as the number of the bits needed to represent N , and the length of “R” symbol sequence will not exceed b_{\max} .

2. Contexts for other types of symbols: For the significance symbols and sign symbols, the contexts used are the same as those defined in EBCOT [6] and JPEG2000 [7], which are established to capture the dependencies between the currently encoded coefficient and its surrounding neighbors; this approach has been proved to be very efficient.

When coding the refinement symbol of each coefficient, the context is dependent on whether this symbol is the first refinement bit of the coefficient, i.e., whether the coefficient has just become significant in the previous bitplane. If so, the context is further determined by the magnitude of its eight immediate neighboring coefficients. Such context setting is due to the fact that the magnitude of a coefficient is weakly dependent on that of its neighbors, so only the context of the first refinement bit is related to the magnitudes of the neighbors. The assignment rules are given in Table 6, in which “x” indicates a “don’t care” state. A state variable mag_n is introduced in Table 6 to flag whether there is at least one neighbor whose current magnitude (i.e., the magnitude restored from the bits which have been coded for the coefficient) is greater than that of the coefficient which is currently being coded. If there is, $mag_n = 1$; otherwise, $mag_n = 0$.

5.3 Summary of the dilation-run coding algorithm

With the notations defined in Table 7, the proposed dilation-run encoding algorithm is summarized with the pseudocode

Table 6 Assignment of the contexts for the refinement symbols

First refinement for a coefficient	mag_n	Context label
No	x	0
Yes	0	1
Yes	1	2

in Table 8. The decoding algorithm is straightforward and can be obtained by simply reversing the encoding process.

In the algorithm, a list LSC is used to record the coordinates of the significant coefficients, so that in the first three passes of each bitplane, the image needs not to be repeatedly scanned for seeking the coefficients which are already flagged as being significant at the previous bitplanes. In the isolated cluster seeking pass, the subbands are scanned in the zigzag order from the lowest resolution to the highest resolution (the same scanning order as that practiced in [1]) for processing the whole image. In order to avoid the unnecessary scanning of the subbands containing no significant coefficient, a preprocessing step is first executed when coding each bitplane. That is, a bit is first output for identifying whether the subband contains no significant coefficient or not. If a subband contains no significant coefficient, a bit 0 is sent; otherwise, 1 is sent. For a subband which already has significant coefficients on the previous bitplane, this bit is redundant and hence skipped.

When implementing the above algorithm for image coding, several state variables, such as the significance state of the coefficient $\sigma_{i,j}$ and the encoding state of the coefficient $e_{i,j}$, need be stored for each pixel in the image. To save the memory overhead, these states are grouped in one byte and saved as *char* type for each coefficient. The list LSC is another item that consumes memory. However, at medium- or high-compression ratios (which are the cases of the most interest for the proposed coder), only a fraction of coefficients are significant, and thus only limited amount of overhead will be consumed by the list.

6 Experimental results

The proposed dilation-run coder has been tested on various images. In the paper, the coding results of eight typical images are shown for performance evaluation. The images include six 512×512 images—Lena, Barbara, Goldhill, Baboon, Boat and Couple, a 2560×2048 high-definition image Café and a 768×768 fingerprint image. All the images are monochrome. Figure 6 shows the contents of seven natural images, while the fingerprint image is given in Fig. 8a. When implementing the proposed coder, all the images are wavelet-transformed by using the Daubechies 9/7 bi-orthogonal filters [17] with a five-level decomposition.

Table 7 Notations used in the algorithm of dilation-run coding

LSC: list of significant coefficients. Each entry in the list is denoted by its coordinate (i, j) ;

S_k : the k th subband. The subbands are numbered with $0, 1, 2, \dots$ in accordance with the zigzag scan order from the lowest resolution to the highest resolution;

n_k : the index of the most significant bitplane of subband S_k ;

$\sigma_{i,j}$: the significance state of coefficient $w_{i,j}$. It is first set to 0, but changed to 1 after it is identified as being significant on some bitplane;

$e_{i,j}$: the encoding state of coefficient $w_{i,j}$, which is reset to 0 when starting the coding of a new bitplane for indicating that $w_{i,j}$ has not been encoded, but changed to 1 after it is encoded;

N_k : the number of the coefficients which have not been encoded in the subband S_k ;

rl : the zero-run length;

$\{(i, j) \oplus B$: the set obtained by applying the dilation operation to pixel (i, j) , with the structuring element B .

Table 8 Algorithm of the dilation-run coding**Dilation-run-encode()**

1. Output $n = \lfloor \log_2(\max_{(i,j)} |w_{i,j}|) \rfloor$. Compute $n_k = \lfloor \log_2(\max_{(i,j) \in S_k} |w_{i,j}|) \rfloor$ for each subband S_k . Set the LSC as an empty list.
 2. Encode the n th bitplane.
 - (2.1) **Preprocessing:** For each subband S_k , if $n_k < n$, output a bit 0; if $n_k = n$, output a bit 1.
 - (2.2) **Intra-band cluster growing pass:** For each entry (i, j) in the LSC (except for those included in the current bitplane), call **Recursive-Dilation-encode** (i, j) .
 - (2.3) **Inter-band cluster growing pass:** For each entry (i, j) in the LSC (except for those included in the current bitplane), do
 - If the pixel has children (i.e., it is neither in the low-pass subband nor in the three finest high-pass subbands) and its children are located within a subband with $n_k \geq n$, then for each child $(i', j') \in \{(2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)\}$, do
 - * If $e_{i',j'} = 0$ and $\sigma_{i',j'} = 0$, then
 - . Set $e_{i',j'} = 1$ and if $|w_{i',j'}| \geq 2^n$, set $\sigma_{i',j'} = 1$;
 - . Output $\sigma_{i',j'}$ with a significance symbol;
 - . If $\sigma_{i',j'} = 1$, output the sign symbol of $w_{i',j'}$, add (i', j') to the LSC and call **Recursive-dilation-encode** (i', j') .
 - (2.4) **Magnitude refinement pass:** For each entry (i, j) in the LSC (except for those included in the current bitplane), output its n th most significant bit with a refinement symbol.
 - (2.5) **Isolated cluster seeking pass:** For each subband S_k with $n_k \geq n$, calculate N_k and set $rl = 0$. If $N_k > 0$, then
 - Scan the subband S_k as described in Section 5.1.3. Along the scanning path, for each coefficient $w_{i,j}$ with $e_{i,j} = 0$ and $\sigma_{i,j} = 0$, do
 - * If $|w_{i,j}| < 2^n$, then
 - . Set $e_{i,j} = 1$ and $rl = rl + 1$;
 - * Else
 - . Output rl in its binary representation with the run-length symbols “0” and “1”, with the MSB dropped for $rl \neq 2^m - 1$, where m is a positive integer;
 - . Output the sign of $w_{i,j}$ with a run-length symbol “+” or “-”;
 - . Set $e_{i,j} = 1$ and $\sigma_{i,j} = 1$. Add (i, j) to the LSC and call **Recursive-dilation-encode** (i, j) ;
 - . Set $rl = 0$;
 - Output a run-length symbol “E”.
 3. Set $n = n - 1$, and go to Step 2.
- Recursive-dilation-encode** (i, j)
1. Scan the pixels in $\{(i, j) \oplus B$ in the raster order. For each pixel (i', j') , do
 - If $e_{i',j'} = 0$ and $\sigma_{i',j'} = 0$, then
 - * Set $e_{i',j'} = 1$ and if $|w_{i',j'}| \geq 2^n$, set $\sigma_{i',j'} = 1$;
 - * Output $\sigma_{i',j'}$ with a significance symbol;
 - * If $\sigma_{i',j'} = 1$, output the sign symbol of $w_{i',j'}$ and add (i', j') to the LSC.
 2. For each new significant coefficient (i', j') identified in the above procedure, call **Recursive-dilation-encode** (i', j') .
-

Fig. 6 Natural test images used for evaluating the coding performance of the proposed dilation-run coder: **a** 512×512 images—Lena, Barbara, Goldhill, Baboon, Boat and Couple (from *top to bottom*, from *left to right*); **b** 2560×2048 image Café



Table 9 Performance comparison (in PSNR [dB]) of the dilation-run algorithm against other state-of-the-art coders on three 512×512 images: (a) Lena, (b) Barbara and (c) Goldhill

Algorithm/rate [bpp]	0.0625	0.125	0.25	0.50	1.00
(a) Lena					
SPIHT [2]	28.38	31.10	34.11	37.21	40.41
SLCCA [5]	–	31.25	34.28	37.35	40.47
ECZQR [13]	–	–	34.10	37.13	40.18
EMDC [14]	–	–	34.50	37.57	40.50
EBCOT (SL) [6]	28.30	31.22	34.28	37.43	40.61
Dilation-run (proposed)	28.58	31.41	34.50	37.59	40.64
(b) Barbara					
SPIHT [2]	23.35	24.86	27.58	31.39	36.41
SLCCA [5]	–	25.36	28.18	31.89	36.69
ECZQR [13]	–	–	28.45	32.44	37.38
EMDC [14]	–	–	28.42	32.16	37.35
EBCOT (SL) [6]	23.45	25.55	28.55	32.48	37.37
Dilation-run (proposed)	23.50	25.49	28.58	32.44	37.40
(c) Goldhill					
SPIHT [2]	26.73	28.48	30.56	33.13	36.55
SLCCA [5]	–	–	30.60	33.26	36.66
ECZQR [13]	–	–	30.58	32.96	36.25
EMDC [14]	–	–	30.75	33.45	36.97
EBCOT (SL) [6]	26.74	28.58	30.71	33.35	36.72
Dilation-run (proposed)	26.92	28.66	30.68	33.44	36.88

The dilation-run algorithm is first evaluated by using classical 512×512 images. The coding results of Lena, Barbara and Goldhill are shown in Table 9, where the PSNR

Table 10 Performance comparison (in PSNR [dB]) of the dilation-run algorithm against the SPIHT coder on three 512×512 images: (a) Baboon, (b) Boat and (c) Couple

Algorithm/rate [bpp]	0.0625	0.125	0.25	0.50	1.00	
(a) Baboon						
SPIHT [2]		20.74	21.72	23.27	25.65	29.17
Dilation-run (proposed)	20.74	21.85	23.31	25.82	29.39	
(b) Boat						
SPIHT [2]	25.38	27.48	30.09	33.22	36.57	
Dilation-run (proposed)	25.49	27.61	30.44	33.64	36.99	
(c) Couple						
SPIHT [2]	25.43	27.74	30.46	33.93	38.31	
Dilation-run (proposed)	25.74	28.14	30.93	34.63	38.93	

(in dB) performance of the dilation-run algorithm is compared against those of five state-of-the-art wavelet image coders including the SPIHT [2] with arithmetic coding, EBCOT [6] in its single layer (SL) mode and three morphology-based coders—SLCCA [5], ECZQR [13] and EMDC [14], which are, to our best knowledge, the top-ranked morphology-based coders described in the literature. Among them, SPIHT, ECZQR and EMDC are embedded coders, while SLCCA and EBCOT are not. In Table 9, the coding results of the SPIHT are obtained by using the program provided by the authors of the original paper, while the coding results of other coders under comparison are directly cited from their respective literature. The coding results of Baboon, Boat and Couple are shown in Table 10, where the performance of the dilation-run coder is only compared against that of the SPIHT, because for these images the results of other

Table 11 Average performance comparison (in PSNR [dB]) of the dilation-run algorithms using different structuring elements

Image/structuring element (SE)	SE4	SE8	SE12	SE24
Lena	34.53	34.54	34.50	34.39
Barbara	29.34	29.48	29.39	29.35
Goldhill	31.31	31.32	31.31	31.29
Baboon	24.20	24.22	24.23	24.16
Boat	30.83	30.83	30.82	30.81
Couple	31.68	31.67	31.66	31.64

The listed are the average PSNRs over five test bit rates (0.0625, 0.125, 0.25, 0.50 and 1.00 bpp)

coders are not available. In both tables, the run-length coders introduced in Sect. 2 are not included, because their PSNR performances are inferior to that of SPIHT.

The results of the dilation-run coding in Tables 9 and 10 are obtained by using the structuring element SE8. On average, SE8 lends better performance to the dilation-run algorithm than other structuring elements shown in Fig. 2, which can be verified by Table 11. The results in Table 11 differ from the phenomena observed in Tables 2 and 3 that the dilation-run coder performs best when the smallest structuring element SE4 is used. This is due to the utilization of the context-based entropy-coding introduced in Sect. 5.2, which increases the coding efficiency of the symbols sent in the dilation process to a certain extent.

When compared with the run-length coders, the dilation-run algorithm has yielded apparent superiority in terms of PSNR performance because it outperforms the SPIHT. Table 9 also shows that for the images Lena and Barbara, the dilation-run algorithm outperforms the morphology-based coders SLCCA, ECZQR and EMDC. The exception is the image Goldhill, for which the EMDC achieves the best performance. When compared with the EBCOT run in its single-layer mode, which is optimized for one target rate but is not scalable, the dilation-run algorithm shows a higher average PSNR for all the images. For a subjective comparison between the performances of the SPIHT and the proposed coder, the details of the compressed Barbara images are displayed in Fig. 7. When observing the high-frequency content of the images (e.g., Barbara's checked trousers), it is apparent that the dilation-run coder reserves much more texture information than the SPIHT does at both bit rates of 0.125 and 0.25 bpp. These comparisons are consistent with the PSNR results documented in Table 9b.

The dilation-run coder is further tested on the high-definition image with large size, because the compression of such images is of more and more interest with the fast development of image acquisition and display equipments. For that, the coding results of the 2560×2048 image Café are given in Table 12, where the PSNR performances of the

Table 12 A performance comparison (in PSNR [dB]) of the dilation-run algorithm against the SPIHT and EBCOT coders on a 2560×2048 high-definition image Café

Algorithm/bit rate [bpp]	0.0625	0.125	0.25	0.50	1.00
SPIHT [2]	18.95	20.67	23.03	26.49	31.74
EBCOT [6]	19.10	20.88	23.29	27.00	32.27
Dilation-run (proposed)	19.23	20.85	23.48	27.19	32.47

Table 13 A performance comparison (in PSNR [dB]) of the dilation-run algorithm against the WSQ, SPIHT and SLCCA coders on a 768×768 fingerprint image

Algorithm/bit rate [bpp]	0.25	0.40	0.444	0.50	1.00
WSQ [19]	–	–	34.43	–	–
SPIHT [2]	32.82	34.98	35.45	36.03	40.00
SLCCA [5]	33.09	35.16	35.65	36.25	40.32
Dilation-run (proposed)	33.24	35.32	35.79	36.46	40.63

SPIHT, EBCOT and dilation-run (using SE8) coders are compared at various bit rates. The proposed coder works well for the high-definition image category and performs best in most cases (only slightly inferior to the PSNR performance of EBCOT at 0.125 bpp by 0.03 dB).

Finally the coding performance of the dilation-run algorithm on fingerprint images, which is a very important type of image data that demands the best coding techniques, is evaluated. As known, the digitized fingerprints of a person could occupy 10MB of storage without compression. Such a huge data amount makes it very difficult to transmit the uncompressed fingerprints in real time. The FBI has developed a fingerprint image compression standard, called the wavelet scalar quantization (WSQ) [19]. The coding results of a 768×768 fingerprint image from the WSQ, SPIHT, SLCCA coders and the dilation-run algorithm (using SE8) are shown in Table 13. Again, the dilation-run algorithm outperforms other three image coders in the PSNR performance. For an appreciation, the original fingerprint image and the reconstructed image from the dilation-run algorithm coded at 0.25 bpp are shown in Fig. 8. In the compressed image, it can be easily observed that the features which are important for fingerprint recognition, such as the ends and the bifurcations of the ridges, are all well reserved.

7 Conclusion

In this study, a new wavelet image coder, called *dilation-run*, is presented that takes the advantages of both the run-length coding and the morphological representation. In the proposed algorithm, the morphological dilation is used to extract the clusters of significant wavelet coefficients, and the run-length

Fig. 7 Details of the 512×512 Barbara images for a subjective evaluation: **a** the original image; **b** images compressed by the SPIHT coder at 0.125 bpp (*top*) and 0.25 bpp (*bottom*); **c** images compressed by the dilation-run coder at 0.125 bpp (*top*) and 0.25 bpp (*bottom*)

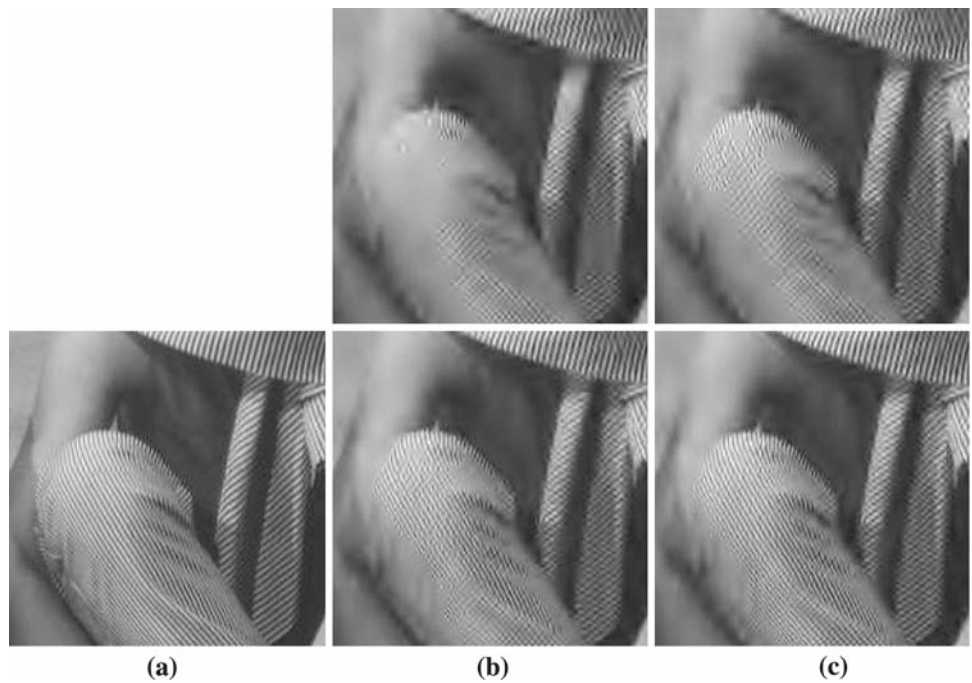
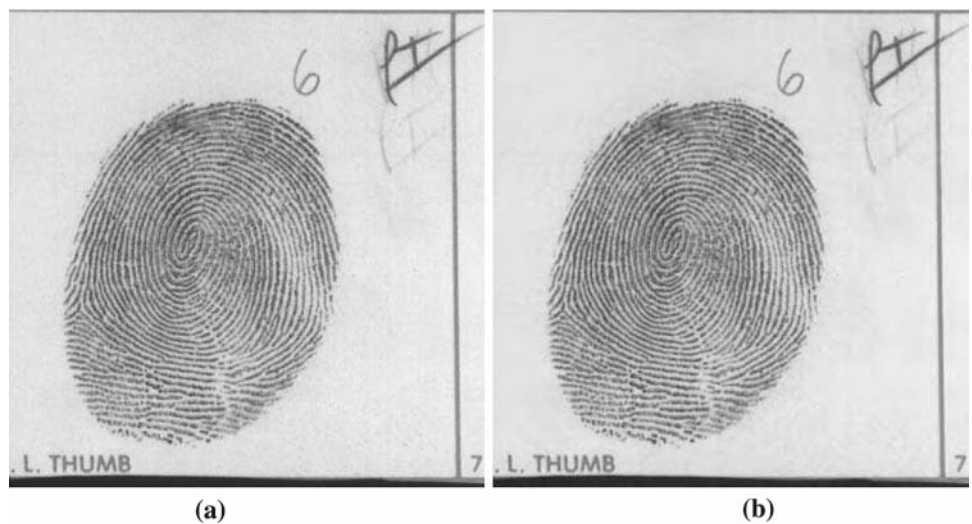


Fig. 8 A dilation-run coding result of a 768×768 fingerprint image: **a** the original image; **b** reconstructed image coded at 0.25 bpp



coding is utilized to record the lengths of zero-runs between clusters. As a result, the run-length coding provides an efficient and low-complexity method to record the seed positions—the key information which needs to be identified in the morphological representation, and the performance of the run-length coding are also much improved by exploiting the morphological representation.

The dilation-run algorithm is implemented in the framework of bitplane coding. With elaborately designed coding passes in each bitplane, the coder is embedded and possesses fine SNR scalability. The coding results justify that the proposed algorithm is among the state-of-the-art wavelet image coders reported in the literature. For the images inherited with distinct clustering nature in the wavelet domain (such as the

fingerprint images), the superiority of our coder in PSNR performance is especially prominent. As shown in Table 13, an additional 1.36 dB PSNR improvement can be achieved beyond that of the WSQ for a fingerprint image coded at 0.444 bpp.

References

1. Shapiro, J.M.: Embedded image coding using zero trees of wavelet coefficients. *IEEE Trans. Signal Process.* **41**(12), 3445–3462 (1993)
2. Said, A., Pearlman, W.A.: A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circuits Syst. Video Technol.* **6**(3), 243–250 (1996)

3. Tsai, M.J., Villasenor, J.D., Chen, F.: Stack-run image coding. *IEEE Trans. Circuits Syst. Video Technol.* **6**(5), 519–521 (1996)
4. Servetto, S.D., Ramchandran, K., Orchard, M.T.: Image coding based on a morphological representation of wavelet data. *IEEE Trans. Image Process.* **8**(9), 1161–1174 (1999)
5. Chai, B., Vass, J., Zhuang, X.: Significance-linked connected component analysis for wavelet image coding. *IEEE Trans. Image Process.* **8**(6), 774–784 (1999)
6. Taubman, D.: High performance scalable image compression with EBCOT. *IEEE Trans. Image Process.* **9**(7), 1158–1170 (2000)
7. JPEG 2000 Part I, Final Committee Draft Version 1.0, ISO/IEC JTC-1/SC 29/WG-1, N1646R, Mar. 2000
8. Tian, J., Wells, R.O. Jr.: A lossy image codec based on index coding. In: *Proc. IEEE Conf. on Data Compression*, pp. 456. 31 March–3 April, 1996
9. Tian, J., Wells, R.O. Jr.: Embedded image coding using wavelet difference reduction. In: Topiwala, P.N., Norwell, M.A. (eds.) *Wavelet Image and Video Compression.*, pp. 289–301. Kluwer, Dordrecht (1998)
10. Walker, J.S., Nguyen, T.Q.: Adaptive scanning methods for wavelet difference reduction in lossy image compression. In: *Proc. IEEE Int. Conf. Image Processing*, vol. 3, pp. 182–185, 10–13 Sept. 2000
11. Yuan, Y., Mandal, M.K.: Context-modeled wavelet difference reduction coding based on fractional bit-plane partitioning. In: *Proc. IEEE Int. Conf. Image Processing*, vol. 2, pp. 251–254. 14–17 Sept. 2003
12. Berghorn, W., Boskamp, T., Lang, M., Peitgen, H.O.: Fast variable run-length coding for embedded progressive wavelet-based image compression. *IEEE Trans. Image Processing* **10**(12), 1781–1790 (2001)
13. Zhong, J.M., Leung, C.H., Tang, Y.Y.: Image compression based on energy clustering and zero-quadtree representation. *Proc. IEEE Vision, Image and Signal Processing* **147**(6), 564–570 (2000)
14. Lazzaroni, F., Leonardi, R., Signoroni, A.: High-performance embedded morphological wavelet coding. *IEEE Signal Process. Lett.* **10**(10), 293–295 (2003)
15. Maragos, P., Schafer, R.: Morphological systems for multidimensional signal processing. *Proc. IEEE* **78**(4), 690–710 (1990)
16. Vincent, L.: Morphological grayscale reconstruction in image analysis: Applications and effective algorithms. *IEEE Trans. Image Process.* **2**(2), 176–201 (1993)
17. Antonini, M., Barlaud, M., Mathieu, P., Daubechies, I.: Image coding using wavelet transform. *IEEE Trans. Image Process.* **1**(2), 205–220 (1992)
18. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. *Commun. ACM* **30**(6), 520–540 (1987)
19. Bradley, J.N., Brislawn, C.M.: The wavelet/scalar quantization compression standard for digital fingerprint images. *Proc. IEEE Int. Symp. Circuits Syst.* **3**, 205–208 (1994)