PAPER

# Bit-Quad-Based Euler Number Computing

**Bin YAO**[†], ***Nonmember***, **Lifeng HE**[†,††a)], ***Member***, **Shiying KANG**[†††], **Xiao ZHAO**[†], ***and*** **Yuyan CHAO**[††††], ***Nonmembers***

**SUMMARY** The Euler number of a binary image is an important topological property for pattern recognition, image analysis, and computer vision. A famous method for computing the Euler number of a binary image is by counting certain patterns of bit-quads in the image, which has been improved by scanning three rows once to process two bit-quads simultaneously. This paper studies the bit-quad-based Euler number computing problem. We show that for a bit-quad-based Euler number computing algorithm, with the increase of the number of bit-quads being processed simultaneously, on the one hand, the average number of pixels to be checked for processing a bit-quad will decrease in theory, and on the other hand, the length of the codes for implementing the algorithm will increase, which will make the algorithm less efficient in practice. Experimental results on various types of images demonstrated that scanning five rows once and processing four bit-quads simultaneously is the optimal tradeoff, and that the optimal bit-quad-based Euler number computing algorithm is more efficient than other Euler number computing algorithms.

***key words:*** *Euler number, topological property, computer vision, pattern recognition, image analysis*

## 1. Introduction

The topological properties of binary images are very important features in the fields of pattern recognition and computer vision. Among others, the Euler number of a binary image, which is defined as the difference between the number of connected components and that of holes in the image, is one of the most important topological properties [1]. The Euler number of a binary image will not change when the image is stretched, flexed or rotated. Therefore, it has been used in many applications: processing cell images in medical diagnosis [2], document image processing [3], shadow detection [4], reflectance-based object recognition [5], and robot vision [6]. Moreover, the Euler number is the most clinically useful feature for discriminating many cervical disorders [7].

Manuscript received January 9, 2017.
Manuscript revised May 4, 2017.
Manuscript publicized June 20, 2017.

[†]The authors are with Artificial Intelligence Institute, College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an, Shaanxi 710021, China.

[††]The author is with Faculty of Information Science and Technology, Aichi Prefectural University, Nagakute-shi, 480–1198, Japan.

[†††]The author is with School of Computer Science, Xianyang Normal University, Xianyang, Shaanxi 712000, China.

[††††]The author is with Faculty of Environment, Information and Business, Nagoya Sangyo University, Owariasahi-shi, 488–8711 Japan.

a) E-mail: helifeng@ist.aichi-pu.ac.jp (Corresponding author)

DOI: 10.1587/transinf.2017EDP7012

Various types of algorithms have been proposed for computing the Euler number of a binary image. There are: (1) skeleton-based algorithm [8], which computes the Euler number by use of the number of terminal points and the number of three edge points in a skeletonized version of the image; (2) bit-quad-based algorithm (*BQ* algorithm) proposed by Gray [9], which computes the Euler number by counting certain bit-quad ($2 \times 2$ pixels) patterns in the image; (3) run-based algorithm (*RUN* algorithm) [10], which computes the Euler number by use of the numbers of runs and the neighboring runs in the image; (4) labeling-based algorithm (*HCS* algorithm) [11], which computes the Euler number by labeling connected components and holes in the image; (5) graph-based algorithms (*GT* algorithm) [12], [13], which compute the Euler number by the number of vertices, edges and basic faces in the shape corresponding to the given image.

Because the *BQ* algorithm is simple in principle, and quite efficient in practice, it is adopted by the famous commercial image processing tools MATLAB [14]. This algorithm has been improved in Refs. [15] and [16]. For convenience, we denote the algorithms proposed in Refs. [15], [16] as the 1-*BQ* algorithm and 2-*BQ* algorithm, respectively.

This paper studies the bit-quad-based Euler number computing problem. We show that by scanning more rows simultaneously to process more bit-quads once and using the information obtained when processing previous pixels, the average number of pixels needed to be checked for processing a bit-quad can continuously decrease in theory, which will lead to a more efficient processing. However, with the increase of the number of bit-quads being processed simultaneously, the length of the codes for implementing a corresponding algorithm will increase, which will make the algorithm less efficient in practice. Experimental results on various types of images showed that scanning five rows once and processing four bit-quads simultaneously is the best choice for the implementation of bit-quad based Euler number computation.

The rest of this paper is organized as follows: in the next section, we review conventional Euler number computing algorithms. We study bit-quad-based Euler number computation in Sect. 3. In Sect. 4, we use experimental results on various types of images to show the best implementation for bit-quad-based Euler number computation and compare it with conventional Euler number comput-

ing algorithms. Lastly, we give our concluding remarks in Sect. 5.

## 2. Reviews of Conventional Euler Number Computing Algorithms

For an $N \times M$-size binary image, we assume that the object (foreground) pixels and non-object (background) pixels in a given binary image are represented by 1 and 0, respectively. As in most image processing algorithms, we assume that all pixels on the border of an image are background pixels. Moreover, we only consider 8-connectivity for object pixels in this paper.

### 2.1 Bit-Quad-Based Algorithms

#### 2.1.1 The *BQ* Algorithm

The *BQ* algorithm [9] for computing the Euler number of a binary image is based on counting certain $2 \times 2$ pixel patterns called bit-quads in the image. For each pixel $b_0$ in the raster scan, the *BQ* algorithm checks the four pixels in the corresponding bit-quad $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \end{bmatrix}$ to confirm whether the bit-quad is one of patterns $Q_1$, $Q_2$, and $Q_3$ shown in Fig. 1. Let $W_1$, $W_2$, and $W_3$ be the numbers of patterns $Q_1$, $Q_2$, and $Q_3$ in a binary image, respectively. Then, the Euler number of the image, namely $E$, can be computed by the following formula.

$$E = (W_1 - W_2 - 2W_3)/4 \tag{1}$$

#### 2.1.2 The 1-*BQ* Algorithm

The 1-*BQ* algorithm proposed in Ref. [15] is an improvement on the *BQ* algorithm. Notice that the left two pixels in the current bit-quad being processed occurred in the previous bit-quad, the 1-*BQ* algorithm utilizes the information about the two pixels obtained during processing the previous bit-quad for processing the current bit-quad, i.e., the 1-*BQ* algorithm only needs to check the right two pixels in the current bit-quad. Thus, for processing a bit-quad in the 1-*BQ* algorithm, the number of pixels to be checked is 2.

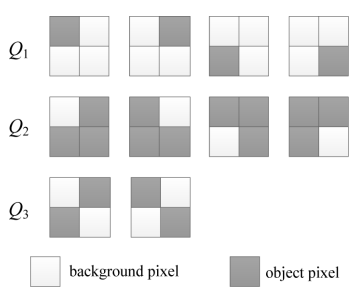



**Fig. 1** Bit-quads for computing the Euler number of a binary image in the *BQ* algorithm.

#### 2.1.3 The 2-*BQ* Algorithm

The 2-*BQ* algorithm [16] is a further improvement on the 1-*BQ* algorithm. For computing the Euler number of a binary image, the 2-*BQ* algorithm scans every other row of the image. For the current pixel $b_0$ in the scan, the 2-*BQ* algorithm processes the bit-quad block $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$, to decide whether the two bit-quads $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \end{bmatrix}$ and $\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$ are the patterns to be counted or not. When processing a bit-quad block $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$, similar as in the 1-*BQ* algorithm, by use of the information about the pixels $a_0$, $a_1$ and $a_2$ obtained during processing the previous bit-quad block, the 2-*BQ* algorithm only needs to check three pixels $b_0$, $b_1$ and $b_2$. Thus, for processing a bit-quad, the number of pixels to be checked is $3/2 = 1.5$.

### 2.2 The *RUN* Algorithm

The *RUN* algorithm [10] computes the Euler number of a binary image by use of the number of runs and the number of neighboring runs in the image.

A *run* is defined to be a maximal sequence of consecutive object pixels in a row. A run $R_1$ is said to be a *neighboring run* of another run $R_2$ if there is at least a pixel in $R_1$ such that it is 8-connected with a pixel in $R_2$. For example, in Fig. 2, there are three runs in the first row, four runs in the second row and four neighboring runs marked by black oval shape between two rows. Let $R$ and $NR$ be the number of runs and that of neighboring runs in a binary image, respectively, the Euler number of the image can be computed by the following formula.

$$E = R - NR \tag{2}$$

Then, the Euler number $E$ of the image shown in Fig. 2 will be $3 + 4 - 4 = 3$.

### 2.3 The *HCS* Algorithm

The *HCS* algorithm [11] computes the Euler number of a binary image according to the definition of the Euler number:

$$E = C - H \tag{3}$$

where $C$ is the number of the connected components, and $H$

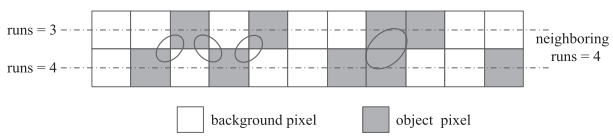



**Fig. 2** An example for explaining runs and neighboring runs in a binary image.

is that of the holes in the image, respectively.

For computing $C$ and $H$ in a binary image, this algorithm extends the first scan of the labeling algorithm proposed in Ref. [17] to calculate the numbers of connected components and holes in the image. For each pixel in the scan, the *HCS* algorithm assigns it a provisional label, and all provisional labels assigned to an 8-connected component or a 4-connected hole in the processed area of the image are combined in an equivalent label set. Thus, after the raster scan, all provisional labels assigned to a connected component or a hole in the image will be combined in an equivalent label set, respectively. Then, by counting the number of the equivalent label sets corresponding to connected components and that for holes, we can obtain the number of connected components, i.e., $C$, and that of holes, i.e., $H$, respectively. Last, we can calculate the Euler number of the image by the formula (3).

### 2.4 The *GT* Algorithm

Both of the algorithms proposed in Ref. [12] and Ref. [13] are graph-based methods. The algorithm proposed in Ref. [12] computes the Euler number of a binary image for 4-connectivity by use of graph theory. A binary image is taken as a square graph, which is constructed by taking all object pixels as vertices and adding all edges $e_{ij}$ such that object pixels $p_i$ and $p_j$ is 4-neighbored. Let $v$, $s$, $e$ be the number of vertices, basic square faces[†] and 4-connected edges, respectively, then the Euler number $E$ can be computed by the following formula.

$$E = v - e + s \tag{4}$$

This algorithm can be extended to compute the Euler number in a binary image for 8-connectivity as follows [13]: Let $v$, $s$, $e$ be the number of vertices, basic right-angled triangle faces[††] and 8-neighbored edges except those hypotenuse edges inside basic square faces, respectively, then the Euler number $E$ of the image can also be computed by the formula (4).

## 3. Study on Bit-Quad-Based Euler Number Computing

As mentioned above, the *BQ* algorithm [9], i.e., the basic bit-quad-based Euler number computing algorithm, is simple in principle and efficient in practices. The *BQ* algorithm has been improved by use of the following two strategies: one is proposed in Ref. [15], which uses the information obtained when processing previous bit-quads to avoid checking pixels repeatedly; the other is proposed in Ref. [16], which reduces the number of pixels to be checked by scanning every other row and processing bit-quads two by two simultaneously.

[†]A basic square face is a pattern $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ in an image.

[††]A basic right-angled triangle face is one of the patterns $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ in an image.

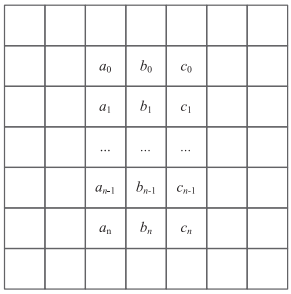



**Fig. 3** Example for explaining our method in the Euler number computation of a binary image.

For any bit-quad-based algorithm, when processing the current bit-quads, by use of the information obtained during processing the previous bit-quads, we can avoid checking those pixels occurring in the previous bit-quads. In the other words, for processing the current bit-quads, we only need to check the pixels that did not occur in the previous bit-quads. Thus, the number of pixels to be check for processing a block of bit-quads is equal to the number of the pixels that did not occurred in the previous bit-quad block. Suppose that we need to check $u$ pixels for processing a pixel block consisting of $v$ bit-quads, the average number of pixels to be checked for processing a bit-quad will be $u/v$.

For convenience, hereafter, we denote a bit-quad-based algorithm which processes $n$ bit-quads simultaneously as *n-BQ* algorithm.

As shown in Fig. 3, if we scan image rows every $n$ rows, and process the bit-quad block $\begin{bmatrix} b_0 & c_0 \\ b_1 & c_1 \\ \vdots & \vdots \\ b_n & c_n \end{bmatrix}$ corresponding to the current pixel $c_0$, we can process $n$ bit-quads $\begin{bmatrix} b_0 & c_0 \\ b_1 & c_1 \end{bmatrix}$, ..., $\begin{bmatrix} b_{n-1} & c_{n-1} \\ b_n & c_n \end{bmatrix}$ simultaneously. Because the pixels $b_0$, $b_1$, ..., $b_n$ have been checked when processing the bit-quad block $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix}$ corresponding to each current pixel $b_0$, we only need to check the $(n+1)$ pixels $c_0, c_1, \ldots, c_n$. Thus, the average number of pixels checked for processing a bit-quad will be $(n+1)/n$.

Obviously, the larger $n$, the smaller the average numbers of pixels to be checked for processing a bit-quad, i.e., the more efficient the corresponding bit-quad-based algorithm will be.

On the other hand, with the increase of $n$, for implementing the *n-BQ* algorithm, we need to consider more and more cases. When $n = 1$, the bit-quad block is $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \end{bmatrix}$, where there are four pixels, we should consider $2^4$ cases.

When $n = 2$, the bit-quad block is $\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$, where there are six pixels, we should consider $2^6$ cases. In the same way, there are $2^8$ cases for $n = 3$, $2^{10}$ cases for $n = 4$, ..., and $2^{2\times(n+1)}$ cases for $n$, to be considered.

Obviously, when the number of bit-quads to be processed increases by 1, the number of cases to be considered will increase by 4 times, i.e., the length of the code for implementing an algorithm will also increase by about 4 times; thus, the efficiency of the algorithm may decrease in practice.

According to the above discussion, with the increase of $n$, for the *n-BQ* algorithm, the average number of pixels to be checked for processing a bit-quad, i.e., $(n+1)/n$, will decrease, which is advantageous for enhancing the efficiency. On the other hand, the cases to be considered, i.e., $2^{2\times(n+1)}$ will increase, which is disadvantageous for enhancing the efficiency. Thus, there will be an optimal $n$ such that the *n-BQ* algorithm is the most efficient. In the next section, we will show, by experimental results on various types of images, that the 4-*BQ* algorithm will be the most efficient.

Now we show that for any $n$, the *n-BQ* algorithm can compute the Euler number in a given binary image correctly. Because the *BQ* algorithm proposed in Ref. [9] is correct, we do this work by showing that for any $n$, the *n-BQ* algorithm does the exactly same work as does in the *BQ* algorithm.

As introduced above, for any $n$, the *n-BQ* algorithm computes the Euler number of a binary image in the same way as in the *BQ* algorithm, i.e., it first counts the certain bit-quad patterns $Q_1$, $Q_2$, and $Q_3$ shown in Fig. 1 in the image and then uses the formula (1) to calculate the Euler number of the image. Therefore, to show the correctness of the *n-BQ* algorithm, we only need to show that each bit-quad is checked once and only once by the *n-BQ* algorithm.

According to the above introduction, the pseudo codes of the *n-BQ* algorithm for calculating the Euler number $E$ of an $N \times M$ binary image can be given as follows[†].

$i \leftarrow 0, j \leftarrow 1, W_1 \leftarrow 0, W_2 \leftarrow 0, W_3 \leftarrow 0$
while $i \le N-1$
  while $j \le M-1$
    check the bit-quads $\begin{bmatrix} a_{i(j-1)} & b_{ij} \\ a_{(i+1)(j-1)} & b_{(i+1)j} \end{bmatrix}$,
    ..., $\begin{bmatrix} a_{(i+n-1)(j-1)} & b_{(i+n-1)j} \\ a_{(i+n)(j-1)} & b_{(i+n)j} \end{bmatrix}$, for each
    $Q_k$, do $W_k \leftarrow W_k + 1$
    $j \leftarrow j + 1$
  end of while
  $i \leftarrow i + n$
end of while
$E = (W_1 - W_2 - 2W_3) / 4$

[†]For convenience, we assume that $N$ is dividable by $n$. If $N = nq + r$, where $q$ and $r$ are integrals and $1 \le r < n$, we can use the *n-BQ* algorithm to process the first $nq$ rows, and then use the *r-BQ* algorithm to process the last $r$ rows.

Obviously, each bit-quad in the image is checked once and only once. Therefore, the *n-BQ* algorithm can calculate the Euler number correctly.

Because any algorithm for computing the Euler number of an $N \times M$-size binary image will access all pixels of the image at least once. Therefore, the lower bound of the time complexity for Euler number computation is $O(N \times M)$.

It is obvious that for processing a bit-quad, the *BQ* algorithm needs to check four pixels in the bit-quad, i.e., it requires four pixel accesses. Therefore, for an $N \times M$-size binary image, the total number of pixel accesses for the *BQ* algorithm is $4N \times M$. On the other hand, as discussed above, for processing a bit-quad, the average number of pixel accesses for the *n-BQ* algorithm is $(n+1)/n$, thus, the total number of pixel accesses for the *n-BQ* algorithm is $(n+1)N \times M/n$.

As for space complexity, the *BQ* algorithm needs no memory to calculate the Euler number. For the *n-BQ* algorithm, as discussed above, in order to reduce the number of pixels to be checked for processing a bit-quad, when processing a bit-quad block $\begin{bmatrix} a_0 & b_0 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix}$, it needs to record $n + 1$ pixels $b_0, \ldots, b_n$ for processing the next bit-quad block. Thus, the necessary memory space for the *n-BQ* algorithm is $(n + 1)$.

## 4. Experimental Results

All algorithms used for our comparison were implemented in the C language on a PC-based workstation (Intel Core i5-3470 CPU@3.20GHz, 4GB Memory, Ubuntu Linux OS), and compiled by the GNU C compiler (version 4.2.3) with the option –O.

Because noise images have complicated geometric shapes and complex connectivity, severe evaluations of algorithms can be performed with these images. Forty-one noise images of each of five sizes (128×128, 256×256, 512×512, 1024×1024, and 2048×2048 pixels) were used for testing (a total of 205 images). For each size, the 41 noise images were generated by thresholding of the images containing uniform random noise with 41 different threshold values from 0 to 1000 in steps of 25. Moreover, fifty natural images obtained from the Standard Image Database (SIDBA) developed by the University of Tokyo[††] and the image database of the University of Southern California[†††], seven texture images downloaded from the Columbia-Utrecht Reflectance and Texture Database[††††], twenty-five medical images obtained from a medical image database of the University of Chicago, and four specialized patterns images (stair-like, spiral-like, saw-tooth-like, and checker-board-like images) were used for testing these algorithms.

[††]http://sampl.ece.ohiostate.edu/data/stills/sidba/index.htm (June 2010)
[†††]http://sipi.usc.edu/database/ (2017)
[††††]http://www1.cs.columbia.edu/CAVE/software/curet/ (2017)

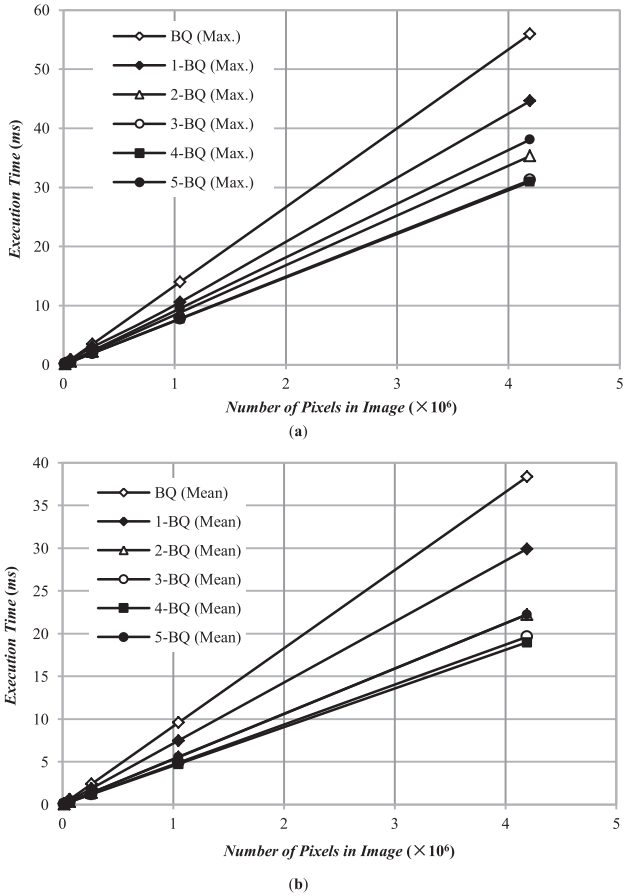



**Fig. 4** Execution time versus image sizes of the *n-BQ* algorithms: (a) the maximum execution time; (b) the average execution time.

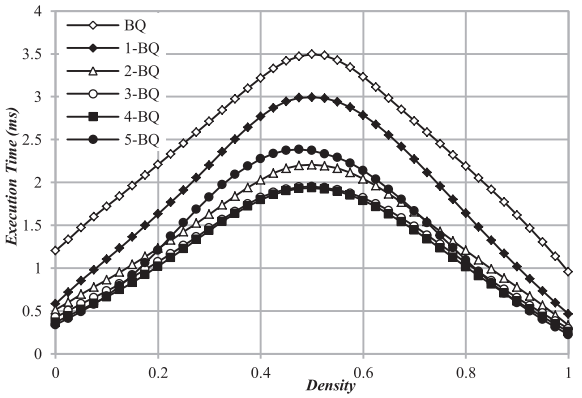



**Fig. 5** Execution time versus image densities of the *n-BQ* algorithms.

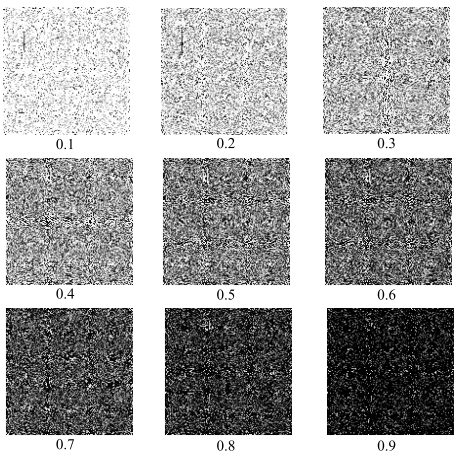



**Fig. 6** Noise images with various densities.

All experimental results presented in this section were obtained by averaging of the execution time for 5000 runs.

### 4.1 Finding an Optimal Bit-Quad-Based Euler Number Computing Algorithm

We implemented the 3-*BQ* algorithm, 4-*BQ* algorithm and 5-*BQ* algorithm, respectively. Then, we compared the *BQ* algorithm, 1-*BQ* algorithm, 2-*BQ* algorithm, 3-*BQ* algorithm, 4-*BQ* algorithm, and 5-*BQ* algorithm on various types of images.

#### 4.1.1 Execution Time versus Image Sizes

All noise images were used for this test. The results are shown in Fig. 4. From Fig. 4, we can find that for both the maximum execution time and the average execution time, all bit-quad-based algorithms have the ideal linear characteristics versus image sizes. Moreover, the efficiency of the *n-BQ* algorithm increases with the increase of $n$ from 0 to 4, but stops at $n = 4$.

#### 4.1.2 Execution Time versus Image Densities

Forty-one noise images with a size of $512 \times 512$ pixels were used for testing the execution time versus the density of the object pixels in an image. The results are shown in Fig. 5. From Fig. 5, we can find that for all images, the efficiency of the *n-BQ* algorithm increases with the increase of $n$ from 0 to 4, but the efficiency of the 5-*BQ* algorithm is worse than that of the 4-*BQ* algorithm. Nine noise images with densities 0.1, 0.2, ..., and 0.9, respectively, are shown in Fig. 6.

#### 4.1.3 Comparisons in Terms of the Maximum, Mean, and Minimum Execution Times on Various Types of Real Images

In this test, all the 50 natural images, 25 medical images, 7 texture images, and 4 artificial images with specialized shape patterns were used for evaluating the algorithms. The resolution of all of these images is $512 \times 512$ pixels. The results are shown in Table 1. From Table 1, we find that for all types of images, the 4-*BQ* algorithm is much more efficient than the *BQ* algorithm, the 1-*BQ* algorithm, the 2-*BQ* algorithm and the 3-*BQ* algorithm for all of the minimum time, average time and the maximum time. Moreover, the 4-*BQ* algorithm is more efficient than the 5-*BQ* algorithm for almost all images for the average time and the maximum time.

**Table 1** Maximum, mean, and minimum execution time on various types of images.

| Image Type | | *BQ* | 1-*BQ* | 2-*BQ* | 3-*BQ* | 4-*BQ* | 5-*BQ* |
|---|---|---|---|---|---|---|---|
| Natural | Max. | 1.86 | 1.34 | 0.98 | 0.85 | 0.82 | 0.97 |
| | Mean | 1.42 | 0.86 | 0.66 | 0.56 | 0.50 | 0.51 |
| | Min. | 1.10 | 0.54 | 0.43 | 0.36 | 0.31 | 0.29 |
| Medical | Max. | 1.47 | 0.90 | 0.69 | 0.58 | 0.52 | 0.52 |
| | Mean | 1.29 | 0.72 | 0.57 | 0.47 | 0.42 | 0.43 |
| | Min. | 1.17 | 0.63 | 0.48 | 0.41 | 0.36 | 0.33 |
| Textural | Max. | 1.73 | 1.17 | 0.87 | 0.75 | 0.71 | 0.79 |
| | Mean | 1.38 | 0.83 | 0.62 | 0.53 | 0.48 | 0.49 |
| | Min. | 1.00 | 0.49 | 0.36 | 0.31 | 0.28 | 0.24 |
| Artificial | Max. | 1.11 | 0.56 | 0.46 | 0.36 | 0.32 | 0.33 |
| | Mean | 0.70 | 0.35 | 0.29 | 0.23 | 0.21 | 0.19 |
| | Min. | 0.28 | 0.16 | 0.12 | 0.10 | 0.08 | 0.08 |

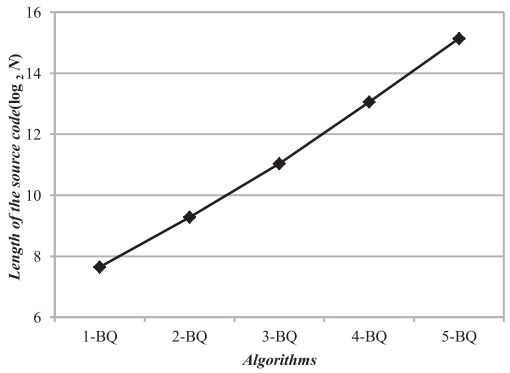



**Fig. 7** Length of the source code of the *n-BQ* algorithms.

#### 4.1.4 Comparisons on the Length of the Code of the *n-BQ* Algorithm

Obviously, if the number of bit-quads to be processed increases by 1, the number of cases to be considered will increase by 4 times. Thus, if the length of the source code of the 1-*BQ* algorithm is $L$, then the length of the source code of the *n-BQ* algorithm will be about $4^{n-1} \times L$. The relation of the real length of the source code of the *n-BQ* algorithm versus $n$ is shown in Fig. 7, which is consistent with our analysis.

### 4.2 Comparison with Other Conventional Euler Number Computing Algorithms

In this subsection, we compare the most efficient bit-quad-based algorithm, i.e., the 4-*BQ* algorithm with the *RUN* algorithm, the *HCS* algorithm, and the *GT* algorithm.

#### 4.2.1 Execution Time versus Image Sizes

All noise images were used for testing the execution time versus image sizes of each algorithm. The results are shown in Fig. 8. From Fig. 8, we can find that both the maximum execution time and the average execution time of all algorithms have ideal linear characteristics versus image sizes. Moreover, for either the maximum execution time or the average execution time, the 4-*BQ* algorithm takes much less time than any of the other algorithms.

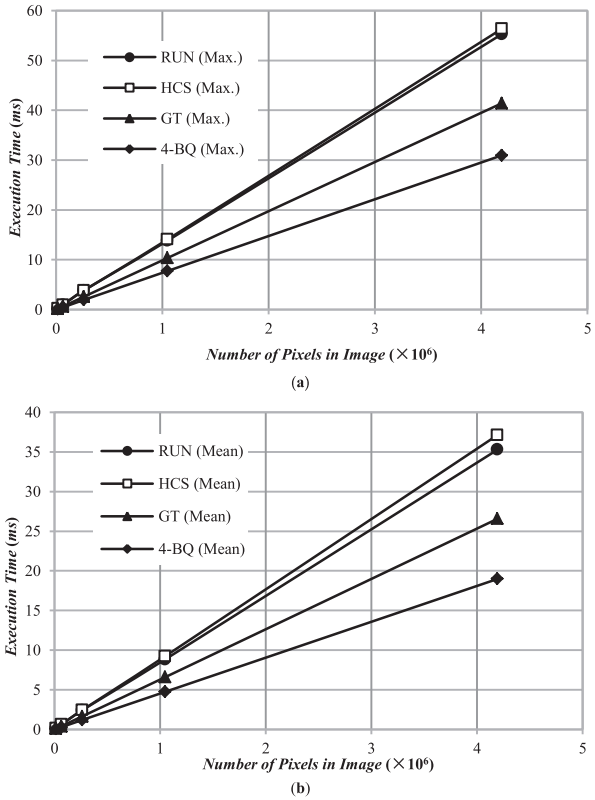



**Fig. 8** Execution time versus image sizes of conventional algorithms: (a) the maximum execution time; (b) the average execution time.

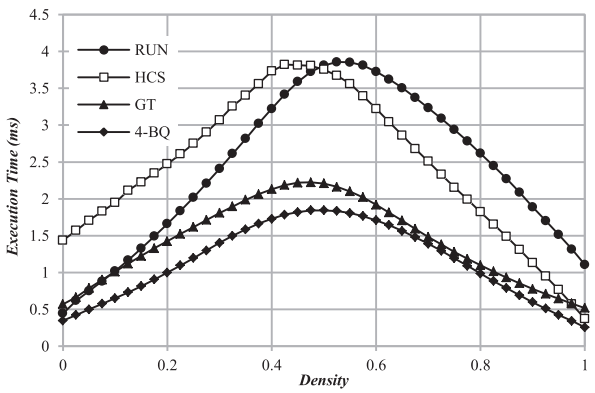



**Fig. 9** Execution time versus image densities of conventional algorithms.

#### 4.2.2 Execution Time versus Image Densities

Similar as in 4.1.2, forty-one noise images with a size of $512 \times 512$ pixels were used for testing the execution time

**Table 2** Maximum, mean, and minimum execution times on various types of images.

| Image Type | | *HCS* | *RUN* | *GT* | 4-*BQ* |
|---|---|---|---|---|---|
| Natural | Max. | 1.97 | 1.69 | 1.02 | 0.82 |
| | Mean | 1.40 | 1.07 | 0.71 | 0.50 |
| | Min. | 0.87 | 0.61 | 0.49 | 0.31 |
| Medical | Max. | 1.50 | 1.07 | 0.73 | 0.52 |
| | Mean | 1.25 | 0.92 | 0.62 | 0.42 |
| | Min. | 0.91 | 0.75 | 0.54 | 0.36 |
| Textural | Max. | 1.60 | 1.66 | 0.92 | 0.71 |
| | Mean | 1.10 | 1.35 | 0.68 | 0.48 |
| | Min. | 0.51 | 1.04 | 0.51 | 0.28 |
| Artificial | Max. | 1.35 | 1.03 | 0.49 | 0.32 |
| | Mean | 0.70 | 0.67 | 0.28 | 0.21 |
| | Min. | 0.32 | 0.24 | 0.11 | 0.08 |

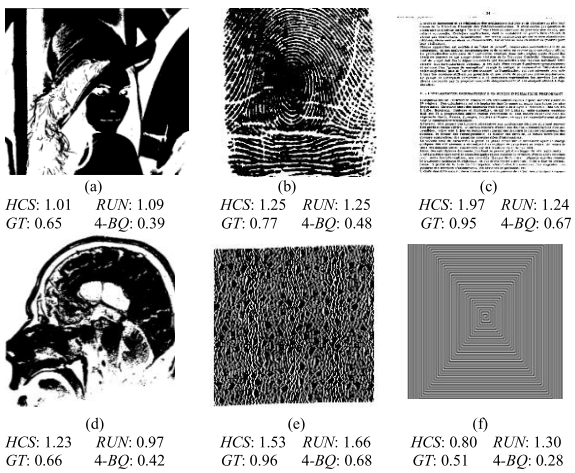



**Fig. 10** Execution time for the selected six images: (a) a portrait image; (b) a fingerprint image; (c) a text image; (d) a medical image; (e) a texture image; (f) an artificial image.

of an algorithm versus the density of the object pixels in an image. The results are shown in Fig. 9. From Fig. 9, we can find that for all images, the 4-*BQ* algorithm is better than all other algorithms, especially the *HCS* algorithm and *RUN* algorithm.

#### 4.2.3 Comparisons in Terms of the Maximum, Mean, and Minimum Execution Time on Various Types of Real Images

All the 50 natural images, 25 medical images, 7 texture images, and 4 artificial images with specialized shape patterns were used for evaluating the algorithms. The results are shown in Table 2.

From Table 2, we find that for all types of images, our algorithm is much more efficient than the *HCS* algorithm, the *RUN* algorithm and the *GT* algorithm for all of the minimum time, the average time and the maximum time. The execution time (*ms*) for the selected six images is illustrated in Fig. 10, where the object pixels are displayed in black.

## 5. Conclusion

In this paper, we studied bit-quad-based Euler number computing problem. On the one hand, we showed that the average number of pixels to be checked for processing a bit-quad would decrease in theory with the increase of the number of bit-quads being processed simultaneously. On the other hand, the more bit-quads being processed simultaneously, the more complicated an algorithm will be. Thus, the algorithm will be less efficient in practice.

Experimental results on various types of images demonstrated that scanning five rows once and processing four bit-quads simultaneously is the optimal tradeoff, and that the optimal bit-quad-based Euler number computing algorithm is more efficient than other Euler number computing algorithms.

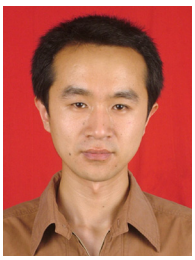
## Acknowledgments

We thank the anonymous referees for their valuable comments that improved this paper greatly. We are grateful to our associate editor Dr. Shimada for his kind cooperation. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61603234, No. 61471227, No. 61601271, and the Scientific Research Program Funded by Shaanxi Provincial Education Department under Grant No. 16JK1099.

### References

[1] R.C. Gonzalez and R.E. Woods, Digital Image Processing, third ed., Pearson Prentice Hall, Upper Saddle River, NJ 07458, 2008.
[2] A. Hashizume, R. Suzuki, H. Yokouchi, et al., "An algorithm of automated RBC classification and its evaluation," Bio Medical Engineering, vol.28, no.1, pp.25–32, 1990.
[3] S.N. Srihari, "Document image understanding," Proc. ACM/IEEE Joint Fall Computer Conference, Dallas, TX, pp.87–95, Nov. 1986.
[4] P.L. Rosin and T. Ellis, "Image difference Threshold strategies and shadow detection," Proc. British Machine Vision Conference, pp.347–356, Sept. 1995.
[5] S.K. Nayar and R.M. Bolle, "Reflectance-based object recognition," International Journal of Computer Vision, vol.17, no.3, pp.219–240, 1996.
[6] B.P.K. Horn, Robot Vision, pp.73–77, McGraw-Hill, New York, 1986.
[7] B.W. Pogue, M.-A. Mycek, and D. Harper, "Image analysis for discrimination of cervical neoplasia," J. Biomedical Optics, vol.5, no.1, pp.72–82, 2000.
[8] J.L. Díaz de León S. and H. Sossa, "On the computation of the Euler number of a binary object," Pattern Recognition, vol.29, no.3, pp.471–476, 1996.
[9] S.B. Gray, "Local properties of binary images in two dimensions," IEEE Trans. Comput., vol.C-20, no.5, pp.551–561, 1971.
[10] A. Bishnu, B.B. Bhattacharya, M.K. Kundu, C.A. Murthy, and T. Acharya, "A pipeline architecture for computing the Euler number of a binary image," Journal of Systems Architecture, vol.51, no.8, pp.470–487, 2005.
[11] L. He, Y. Chao, and K. Suzuki, "An algorithm for connected-component labeling, hole labeling and Euler number computing," Journal of Computer Science and Technology, vol.28, no.3, pp.468–

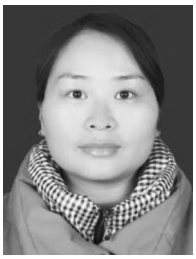
478, 2013.
[12] M.H. Chen and P.F. Yan, "A fast algorithm to calculate the Euler number for binary images," Pattern Recognition Letters, vol.8, no.5, pp.295–297, 1988.
[13] B. Yao, L. He, S. Kang, et al., "A novel bit-quad-based Euler number computing algorithm," Springerplus, 4(735), pp.1–16, 2015.
[14] C.M. Thompson and L. Shure, Image Processing Toolbox, The Math Works, 2012.
[15] B. Yao, H. Wu, Y. Yang, Y. Chao, A. Ohta, H. Kawanaka, and L. He, "An efficient strategy for bit-quad-based Euler number computing algorithm," IEICE Trans. Inf. & Syst., vol.E97-D, no.5, pp.1374–1378, 2014.
[16] B. Yao, L. He, S. Kang, X. Zhao, and Y. Chao, "A further improvement on bit-quad-based Euler number computing algorithm," IEICE Trans. Inf. & Syst., vol.E99-D, no.2, pp.545–549, 2016.
[17] L. He, Y. Chao, and K. Suzuki, "An efficient first-scan method for label-equivalence-based labeling algorithms," Pattern Recognition Letters, vol.31, no.1, pp.28–35, 2010.



**Bin Yao** received the B.E. and M.S. degrees from the Shaanxi University of Science and Technology, China, in 2003 and 2006, respectively. From 2006 to 2010, he was an Assistant Professor with the College of Electrical and Information Engineering, Shaanxi University of Science and Technology. Since 2011, he has been a Lecturer. His research interests include image processing, artificial intelligence, pattern recognition. He is a member of CCF.

**Lifeng He** received the B.E. degree from the Northwest Institute of Light Industry, China, in 1982, the second B.E. degree from Xian Jiaotong University, China, in 1986, and the M.S. and Ph.D. degrees in AI and computer science from the Nagoya Institute of Technology, Japan, in 1994 and 1997, respectively. He is a Professor with Aichi Prefectural University, Japan, and a Guest Professor with the Shaanxi University of Science and Technology, China. From 2006 to 2007, he was with the University of Chicago, USA, as a Research Associate. His research interests include intelligent image processing, computer vision, automated reasoning, pattern recognition, string searching, and artificial intelligence. He has been serving as a referee for more than ten journals in computer science fields, including the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Image Processing, the IEEE Transactions on Computers, Pattern Recognition, Computer Vision and Image Understanding, and Pattern Recognition Letter. He is a member of IPSJ, IEICE, and AAR.

**Shiying Kang** received the B.E. and M.S. degrees from the Shaanxi University of Science and Technology, China, in 2004 and 2007, respectively. From 2007 to 2010, she was an Assistant Professor with the School of Computer Science, Xianyang Normal University. Since 2011, she has been a Lecturer. Her research interests include image processing and pattern recognition.



**Xiao Zhao** received the B.E. and M.S. degrees from the Shaanxi University of Science and Technology, China, in 2001 and 2006, respectively. From 2001 to 2006, she was an Assistant Professor with the College of Electrical and Information Engineering, Shaanxi University of Science and Technology. Since 2007, she has been a Lecturer. Her research interests include image processing, artificial intelligence, pattern recognition, and string searching.

**Yuyan Chao** received the B.E. degree from the Northwest Institute of Light Industry, China, in 1984, and the M.S. and Ph.D. degrees from Nagoya University, Japan, in 1997 and 2000, respectively. From 2000 to 2002, she was a Special Foreign Researcher of the Japan Society for the Promotion of Science, Nagoya Institute of Technology. She is a Professor with Nagoya Sangyo University, Japan, and a Guest Professor with the Shaanxi University of Science and Technology, China. Her research interests include image processing, graphic understanding, CAD, pattern recognition, and automated reasoning.