Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 2130–2151
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Efficient algorithms for coding Hilbert curve of arbitrary-sized image and application to window query

Kuo-Liang Chung [a,*,1], Yi-Luen Huang [a], Yau-Wen Liu [b]

[a] *Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC*

[b] *Department of Computer Science and Information Engineering, National Taiwan University, No. 1, Section 4, Roosevelt Road, Taipei 10617, Taiwan, ROC*

Received 22 November 2005; received in revised form 12 December 2006; accepted 13 December 2006

Abstract

Previously, several efficient Hilbert scan-based operations were presented, but they all suffer from the constraint that the image size must be $2^r \times 2^r$. Considering an image with arbitrary size $I_1 \times I_2$, this paper first presents an efficient snake scan-based algorithm for coding the Hilbert curve of the given image. The proposed coding algorithm takes $O(k + \log U)$ time to code the Hilbert order of one pixel where k denotes the number of the quadrants and $U = \min(I_1, I_2)$. Next, a memory-saving Hilbert curve representation called HCGL is presented for representing the encoded Hilbert curve and it can be constructed in $O(L^2 \log L)$ time where $L = \max(I_1, I_2)$. Based on the HCGL representation of arbitrary-sized image, an application to window query is presented and the proposed window query algorithm takes $O(M \log L + P)$ time where M denotes the number of generated maximal quadtree blocks and P denotes the number of output codes. Under the same PSNR (Peak Signal to Noise Ratio), experimental results demonstrate that our proposed HCGL representation outperforms some existing related algorithms in terms of execution-time and BPP (Bit Per Pixel). In addition, our proposed window query algorithm has been justified in the GIS (Geographical Information System) application.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Compression; Hilbert curve; Image representation; Maximal quadtree block; Quadrant; Snake scan-based approach; Window query

1. Introduction

Cantor [7] was the first researcher to map the interval $[0, 1]$ into the square $[0, 1]^2$. Later the first space-filling curve, the Peano curve [33], was presented to construct a curve that passes through every entry of a two-dimensional region. Afterwards, several different space-filling curves [17,26,30,37] were presented and the

* Corresponding author.
E-mail address: klchung@csie.ntust.edu.tw (K.-L. Chung).

[1] Supported by National Council of Science of ROC under Contracts NSC94-2213-E-011-042, NSC95-2221-E-011-152, and NSC95-2218-E-001-009.

Hilbert curve is the most well-known. Previously, Liu and Schrack [27,28] presented efficient encoding and decoding algorithms to map 2-D/3-D images to Hilbert curves. Based on the Hilbert curve representation, many applications [3,4,6,8,10,18,23–25,35,36,39,43,11,9,15,5,29] have been developed. These Hilbert curve-based applications all suffer from the common constraint that the given image must be of size $2^r \times 2^r$. The main motivation of this research is to relax this constraint and to develop efficient algorithms for coding the Hilbert curve on arbitrary-sized image, for compressing the Hilbert curve, and for performing the window query on the compressed Hilbert curve representation.

Considering an image with arbitrary size $I_1 \times I_2$, first an efficient snake scan-based algorithm is presented in this paper for coding the Hilbert curve and the proposed algorithm takes $O(k + \log U)$ time to code the Hilbert order of one pixel where k denotes the number of the quadrants and $U = \min(I_1, I_2)$. Next, a memory-saving Hilbert curve representation called HCGL is presented to compress the encoded Hilbert curve and the proposed HCGL representation can be constructed in $O(L^2 \log L)$ time where $L = \max(I_1, I_2)$. The proposed HCGL representation outperforms the previous two representations – the split point approximation [19] and the zero order interpolation [25]. Based on HCGL representation for arbitrary-sized image, a window query application is presented and the proposed window query algorithm takes $O(ML + P)$ time where M denotes the number of maximal quadtree blocks and P denotes the number of output codes. The proposed window query algorithm is faster than the naive algorithm which needs $O(w_1 w_2 L + P)$ time where the query window is of size $w_1 \times w_2$. We definitely have $w_1 w_2 > M$. Under the same PSNR (Peak Signal to Noise Ratio), experimental results demonstrate that our proposed HCGL representation outperforms some existing related algorithms in terms of execution-time and BPP (Bit Per Pixel). In addition, our proposed window query algorithm has been justified in GIS (Geographical Information System) application.

The rest of the paper is organized as follows. Section 2 presents the previous efficient coding scheme proposed by Liu and Schrack [27] for coding the Hilbert curve of a $2^r \times 2^r$ image. Based on the proposed snake scan-based approach, Section 3 presents an efficient algorithm for coding the Hilbert curve of an arbitrary-sized image. Section 4 presents the proposed HCGL representation for compressing the constructed Hilbert curve. Section 5 presents the proposed window query algorithm based on HCGL representation. Section 6 demonstrates some related experimental results. Finally, Section 7 addresses some concluding remarks.

2. The past work by Liu and Schrack

This section introduces the previous efficient coding scheme proposed by Liu and Schrack [27] for generating the Hilbert curve of a $2^r \times 2^r$ image. The result of Liu and Schrack will be used as a subroutine in our proposed snake scan-based coding scheme for generating the Hilbert curve of an arbitrary-sized image.

Considering a sub-image of size $2^r \times 2^r$, according to the efficient encoding scheme by Liu and Schrack [27], the pixel at location $(x, y) = ((x_{r-1} \ldots x_1 x_0)_2, (y_{r-1} \ldots y_1 y_0)_2)$ can be encoded to Hilbert order which is represented by a quaternary digit string $h = (q_{r-1} \ldots q_1 q_0)_4 = \sum_0^{r-1} 4^i q_i$ where $q_i \in \{0, 1, 2, 3\}$. Let the two bits of a quaternary digit h_k in h be represented by h_{2k+1} and h_{2k}. The recursive encoding formulas for calculating h_{2k+1} and h_{2k} are given by

$$\begin{aligned} h_{2k+1} &= \bar{v}_{0,k}(v_{1,k} \oplus x_k) + v_{0,k}(v_{1,k} \oplus \bar{y}_k) \\ h_{2k} &= x_k \oplus y_k \end{aligned} \tag{1}$$

where $k = 0, 1, \ldots, r-1$ and the values of $v_{0,k}$ and $v_{1,k}$ can be calculated by

$$\begin{aligned} v_{0,r-1} &= 0 \\ v_{1,r-1} &= 0 \\ v_{0,j-1} &= v_{0,j}(v_{1,j} \oplus \bar{x}_j) + \bar{v}_{0,j}(v_{1,j} \oplus \bar{y}_j) \\ v_{1,j-1} &= v_{1,j}(x_j \oplus y_j) + (\bar{x}_j \oplus y_j)(v_{0,j} \oplus \bar{y}_j) \end{aligned} \tag{2}$$

where $j = r-1, \ldots, 2, 1$.

When $r = 1$, the pixel at position $(1, 1)$ can be encoded to Hilbert order $h = (h_1 h_0)_2 = 2\ (= (02)_4)$ according to the following derivation of Eq. (1) and (2):

$$\begin{aligned} v_{0,0} &= 0 \\ v_{1,0} &= 0 \\ h_1 &= \bar{0}(0 \oplus 1) + 0(0 \oplus \bar{1}) &&= 1 \\ h_0 &= 1 \oplus 1 &&= 0. \end{aligned}$$

According to the above encoding scheme, the Hilbert curves of resolution 1, resolution 2, and resolution 3 are shown in Fig. 1, where the lower-left corner of the image is defined as the origin and the encoded Hilbert orders are recorded along the Hilbert curve. In general, given the position (x, y) of one pixel in the image, the above encoding scheme by Liu and Schrack can encode the pixel into Hilbert order in $O(r)$ time [27].

The decoding formulas are used to map the Hilbert order h along the Hilbert curve into the position $(x, y) = ((x_{r-1} \ldots x_1 x_0)_2, (y_{r-1} \ldots y_1 y_0)_2)$ of the pixel in the image. For $k = 0, 1, \ldots, r-1$, x_k and y_k are calculated by

$$\begin{aligned} x_k &= (v_{0,k}\overline{h}_{2k}) \oplus v_{1,k} \oplus h_{2k+1} \\ y_k &= (v_{0,k} + h_{2k}) \oplus v_{1,k} \oplus h_{2k+1} \end{aligned} \tag{3}$$

where $v_{0,k}$ and $v_{1,k}$ are calculated by

$$\begin{aligned} v_{0,r-1} &= 0 \\ v_{1,r-1} &= 0 \\ v_{0,j-1} &= v_{0,j} \oplus h_{2j} \oplus \overline{h}_{2j+1} \\ v_{1,j-1} &= v_{1,j} \oplus (\overline{h}_{2j}\overline{h}_{2j+1}) \end{aligned} \tag{4}$$

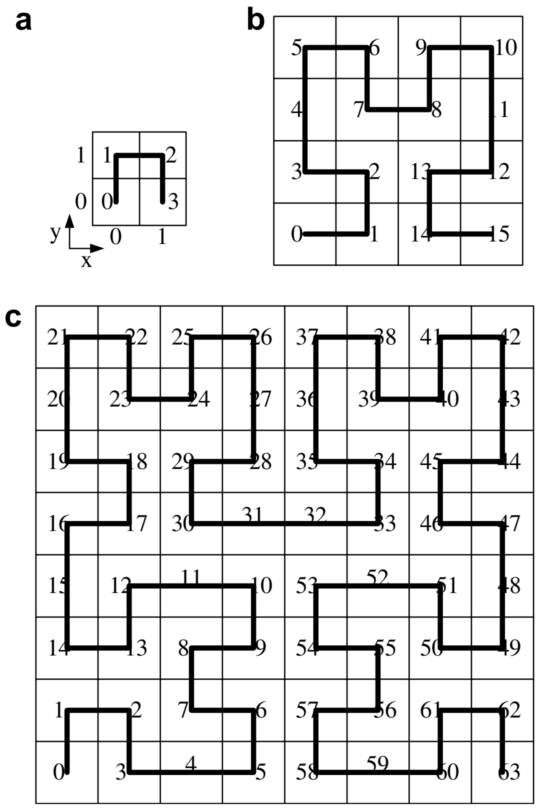

Fig. 1. Hilbert curves H_1, H_2, and H_3 for resolutions 1, 2, and 3, respectively. (a) Hilbert curve H_1. (b) Hilbert curve H_2. (c) Hilbert curve H_3.

where $j = r-1, \ldots, 2, 1$. For example, when $r = 1$, the Hilbert order $h = 3$ $(= (11)_2)$ along the Hilbert curve can be decoded to the position $(x, y) = ((x_0)_2, (y_0)_2)$ of the pixel in the image according to the following derivation using Eq. (3) and (4):

$$\begin{aligned} v_{0,0} &= 0 \\ v_{1,0} &= 0 \\ x_0 &= (0\overline{1}) \oplus 0 \oplus 1 &&= 1 \\ y_0 &= (0+1) \oplus 0 \oplus 1 &&= 0. \end{aligned}$$

Thus, the Hilbert order $h = 3$ yields the decoded position $(x, y) = (1, 0)$.

3. The proposed snake scan-based coding algorithm

Based on the proposed snake scan-based approach, this section presents an efficient scheme for coding the Hilbert curve of an arbitrary-sized image. The proposed coding scheme extends Liu and Schrack's work from the $2^r \times 2^r$ image domain to the image domain of arbitrary size.

3.1. Encoding arbitrary-sized image into a set of Hilbert curves

Definition 1. A square sub-image is called a quadrant if the square sub-image is of size $2^r \times 2^r$.

Lemma 1 [27]. *Each $2^r \times 2^r$ quadrant can be encoded into the Hilbert curve H_r.*

The main idea of the proposed snake scan-based scheme for coding the Hilbert curve of the given arbitrary-sized image is that the given arbitrary-sized image domain is first partitioned into a set of square sub-images. Then, each partitioned square sub-image is further decomposed into a set of quadrants. For one square sub-image, we first present one concatenation strategy to concatenate the corresponding Hilbert curves to form a longer Hilbert curve. Note that by Lemma 1, each concatenated Hilbert curve is encoded from the partitioned quadrant in the square sub-image. Then for the given arbitrary-sized image, we present another concatenation strategy to concatenate these Hilbert curves to form a complete Hilbert curve where each concatenated Hilbert curve is constructed from the partitioned square sub-image in the arbitrary-sized image.

Given a 15×15 square sub-image as shown in Fig. 2, the upper-left $2^3 \times 2^3$ quadrant is first partitioned because of $2^3 \leqslant 15 < 2^4$. By Lemma 1, the partitioned $2^3 \times 2^3$ quadrant can be represented by the Hilbert curve H_3. Because of $2^2 \leqslant 7(= (15 - 2^3)) < 2^3$, five $2^2 \times 2^2$ quadrants, which are adjacent to H_3, are partitioned. Each partitioned $2^2 \times 2^2$ quadrant can be represented by the Hilbert curve H_2. Following this partition rule, due to $2^1 \leqslant 3(= (7 - 2^2)) < 2^2$ and $2^0 \leqslant 1(= (3 - 2^1)) < 2^1$, 13 H_1's and 29 H_0's are obtained, respectively. According to well-ordering principle [21], the above decomposition process only needs finite partition steps

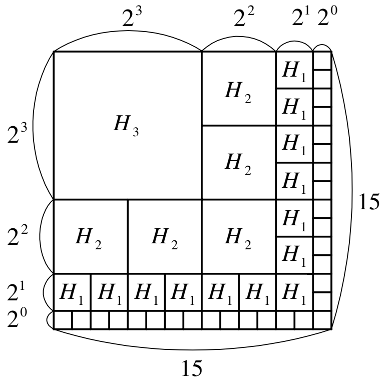

Fig. 2. An example for encoding a 15×15 square sub-image into a set of Hilbert curves.

to encode any square sub-image into a set of Hilbert curves. After discussing the special example in Fig. 2, the construction of Hilbert curves for the general square sub-image is discussed below.

Although the above case is for the image with size 15×15, in the following lemma, a general procedure is presented for such a partition. Let the general square sub-image be of size $I \times I$, then we have the following result.

Lemma 2. *For any $I \times I$ square sub-image, it can be encoded (partitioned) into a set of Hilbert curves (quadrants) and the number of Hilbert curves (quadrants) is bounded by $O(I)$.*

Proof. First, the case $I = 2^r - 1$ is taken into consideration (see Fig. 3). In this case, the given $I \times I$ $(=(2^r - 1) \times (2^r - 1))$ square sub-image can be decomposed into one $2^{r-1} \times 2^{r-1}$ quadrant, five $2^{r-2} \times 2^{r-2}$ quadrants, ..., and $(2^{r+1} - 3)$ $2^0 \times 2^0$ quadrants. These decomposed quadrants can be represented by one H_{r-1}, five H_{r-2}'s, ..., and $(2^{r+1} - 3)$ H_0's, respectively. For this case, the number of encoded Hilbert curves, i.e. the number of partitioned quadrants, is calculated by

$$\begin{aligned} Q &= \sum_{i=0}^{r-1}(2^{r-i+1} - 3) = \sum_{i=0}^{r-1} 2^{r-i+1} - \sum_{i=0}^{r-1} 3 = (2^2 + 2^3 + \cdots + 2^{r+1}) - 3r = 4(2^r - 1) - 3r \\ &= 4I - 3\log(I + 1) = \mathrm{O}(I). \end{aligned}$$

After discussing the case for $I = 2^r - 1$, we examine the general case for $I = 2^r - 1 - X$ where $X = (b_{r-1}2^{r-1} + b_{r-2}2^{r-2} + \cdots + b_1 2^1 + b_0 2^0)$, $b_i \in \{0, 1\}$ and $0 \leqslant i \leqslant r - 1$. $I = 2^r - 1 - X$ can be rewritten by

$$\begin{aligned} I &= 2^r - 1 - (b_{r-1}2^{r-1} + b_{r-2}2^{r-2} + \cdots + b_1 2^1 + b_0 2^0) \\ &= (2^{r-1} + 2^{r-2} + \cdots + 2^1 + 2^0) - (b_{r-1}2^{r-1} + b_{r-2}2^{r-2} + \cdots + b_1 2^1 + b_0 2^0) \\ &= [(1 - b_{r-1})2^{r-1} + (1 - b_{r-2})2^{r-2} + \cdots + (1 - b_1)2^1 + (1 - b_0)2^0]. \end{aligned}$$

Without loss of generality, assume $b_i = 1, b_j = 0, i \neq j$, i.e. $I = 2^{r-1} + \cdots + 2^{i+1} + 2^{i-1} + 2^{i-2} + \cdots + 2^0$. Then the quadrants shown in the shaded area of Fig. 4 are removed from the $(2^r - 1) \times (2^r - 1)$ square sub-image. In Fig. 4, each removed quadrant is of size $2^i \times 2^i$ and the number of the removed quadrants is

$$\begin{aligned} Q_i &= 2 \times (l_i/2^i) + 1 \\ &= 2 \times ((2^{r-1} + \cdots + 2^{i+1})/2^i) + 1 \\ &= 2^{r-i} + \cdots + 2^2 + 1 = 2^{r-i+1} - 2^2 + 1 \\ &= 2^{r-i+1} - 3. \end{aligned}$$

After removing these $2^{r-i+1} - 3$ quadrants, each with size $2^i \times 2^i$. Fig. 4 illustrates the removal of related quadrants. Before removing these $2^{r-i+1} - 3$ quadrants where each quadrant is of size $2^i \times 2^i$, the number of quadrants, each with size $2^{i-1} \times 2^{i-1}$, is given by

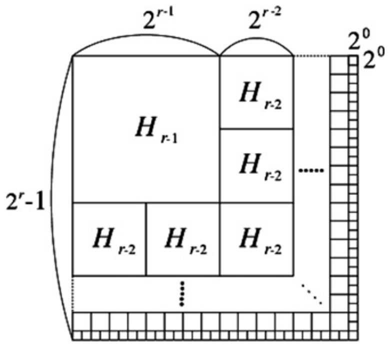

Fig. 3. The worst case for decomposing the $I \times I$ square image into a set of Hilbert curves.

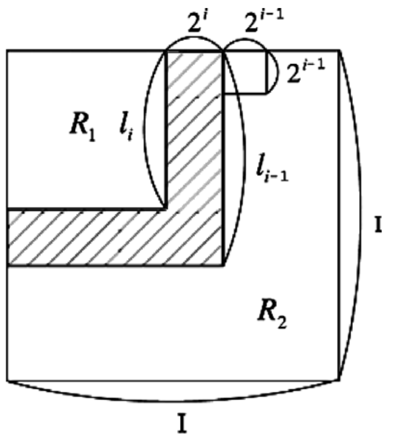

Fig. 4. Removing the quadrants, each with size $2^i \times 2^i$.

$$\begin{aligned}
Q_{i-1} &= 2 \times (l_{i-1}/2^{i-1}) + 1 \\
&= 2 \times ((2^{r-1} + \cdots + 2^i)/2^{i-1}) + 1 \\
&= 2^{r-i+1} + \cdots + 2^2 + 1 \\
&= 2^{r-i+2} - 2^2 + 1 = 2^{r-i+2} - 3.
\end{aligned}$$

After removing these quadrants in the shaded area of Fig. 4, the value of the above $Q_{\text{i-1}}$ is changed to

$$\begin{aligned}
Q'_{i-1} &= 2 \times (l_i/2^{i-1}) + 1 \\
&= 2 \times ((2^{r-1} + \cdots + 2^{i+1})/2^{i-1}) + 1 \\
&= 2^{r-i+1} + \cdots + 2^3 + 1 \\
&= 2^{r-i+2} - 2^3 + 1 \\
&= 2^{r-i+2} - 7.
\end{aligned}$$

Because of $4 = Q_{i-1} - Q'_{i-1}$, by induction, it is not hard to verify that $4 = Q_{i-2} - Q'_{i-2}$, $4 = Q_{i-3} - Q'_{i-3}, \ldots$, and $4 = Q_0 - Q'_0$ hold. Therefore, the number of the total removed quadrants is equal to $M_i = Q_i + 4i = 2^{r-i+1} - 3 + 4i$. Because of $0 \leqslant i \leqslant r - 1$, we have $M_i > 0$. Consequently, the number of the partitioned quadrants in the $(2^r - 1 - X) \times (2^r - 1 - X)$ square sub-image must be less than or equal to that of the partitioned $(2^r - 1) \times (2^r - 1)$ square sub-image. It comes to a conclusion that given an $I \times I$ square sub-image, the number of the constructed Hilbert curves (or partitioned quadrants) is always bounded by O(I). The proof is completed. □

From Lemma 2, it is known that an arbitrary $I \times I$ square sub-image can be encoded into O(I) Hilbert curves or partitioned into O(I) quadrants. We next describe how to partition an arbitrary $I_1 \times I_2$ image into a set of square sub-images and prove that any arbitrary-sized $I_1 \times I_2$ image can be encoded (partitioned) into a set of O(L) Hilbert curves (quadrants) where $L = \max(I_1, I_2)$.

Given a 7×17 image as shown in Fig. 5a, we partition the given image according to the left-to-right scanning way then top-to-down scanning way alternatively. Fig. 5a can be partitioned into two maximal square sub-images, each with size 7×7, two middle square sub-images, each with size 3×3, and three 1×1 square sub-images. By Lemma 2, as shown in Fig. 5b, each maximal square sub-image can be encoded by 1 H_2, 5 H_1's, and 13 H_0's. As shown in Fig. 5c, each middle square sub-image can be encoded by 1 H_1 and 5 H_0's.

Theorem 1. *Any arbitrary-sized $I_1 \times I_2$ image can be encoded (partitioned) into O(L) Hilbert curves (quadrants) where $L = \max(I_1, I_2)$.*

Proof. Without loss of generality, assume $I_1 < I_2$ and let $L_1 = \lfloor I_2/I_1 \rfloor \times I_1$. As shown in Fig. 6, the image is first partitioned into one $I_1 \times L_1$ sub-image and one $I_1 \times (I_2 - L_1)$ sub-image. Next, the $I_1 \times L_1$ sub-image is partitioned into (L_1/I_1) smaller square sub-images, each with size $I_1 \times I_1$. By Lemma 2, each $I_1 \times I_1$ square

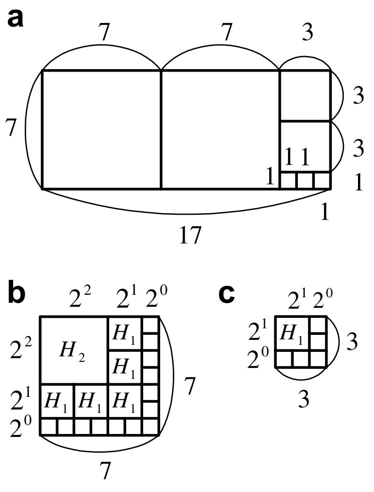

Fig. 5. Encoding a 7×17 image into a set of Hilbert curves. (a) The given 7×17 image. (b) The partitioned maximal 7×7 square sub-image. (c) The partitioned middle 3×3 square sub-image.

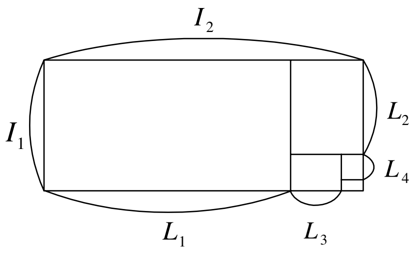

Fig. 6. Partitioning the arbitrary-sized image into a set of sub-images.

sub-image can be encoded into $O(I_1)$ Hilbert curves. Thus, the number of the encoded Hilbert curves in the $I_1 \times L_1$ sub-image is bounded by $O((L_1/I_1) \times I_1) = O(L_1)$. By the same arguments, the remaining $I_1 \times (I_2 - L_1)$ sub-image also can be partition into one $(I_2 - L_1) \times L_2$ sub-image and one $(I_2 - L_1) \times (I_1 - L_2)$ sub-image where $L_2 = \lfloor I_1/(I_2 - L_1) \rfloor \times (I_2 - L_1)$. The $(I_2 - L_1) \times L_2$ sub-image can be encoded into $O(L_2)$ Hilbert curves. Repeating the above rule to partition the remaining $(I_2 - L_1) \times (I_1 - L_2)$ sub-image until the remaining sub-image is of size $Z_1 \times Z_2$ such that when $Z_1 > Z_2$ ($Z_1 > Z_2$), Z_2 (Z_1) is a factor of Z_1 (Z_2); at that time, the final remaining sub-image can be partitioned into Z_1/Z_2 (Z_2/Z_1) square images. By Lemma 2, the final sub-image can be encoded into $O(L_r)$ Hilbert curves.

After encoding the $I_1 \times\ I_2$ image into a set of Hilbert curves, we now analyze the upper bound of the number of required Hilbert curves. Let us first consider the case when r is an even integer. The number of the encoded Hilbert curves in the partitioned $I_1 \times I_2$ image is calculated by

$$\begin{aligned} Q &= O(L_1) + O(L_2) + O(L_3) + O(L_4) + \cdots + O(L_r) \\ &= [O(L_1) + O(L_3) + \cdots + O(L_{r-1})] + [O(L_2) + O(L_4) + \cdots + O(L_r)]. \end{aligned}$$

Because of $I_2 > L_1 + L_3 + \cdots + L_{r-1}$ and $I_1 = L_2 + L_4 + \cdots + L_r$, the value of Q is bounded by

$$Q = O(I_2) + O(I_1) = O(I_2) = O(L)$$

where $L = \max(I_1, I_2)$. Considering the case when r is an odd integer, the value of Q is bounded by

$$\begin{aligned} Q &= O(L_1) + O(L_2) + O(L_3) + O(L_4) + \cdots + O(L_r) \\ &= [O(L_1) + O(L_3) + \cdots + O(L_r)] + [O(L_2) + O(L_4) + \cdots + O(L_{r-1})]. \end{aligned}$$

Because of $I_2 = L_1 + L_3 + \cdots + L_r$ and $I_1 > L_2 + L_4 + \cdots + L_{r-1}$, the value of Q is bounded by

$$\begin{aligned} Q &= \mathrm{O}(I_2) + \mathrm{O}(I_1) \\ &= \mathrm{O}(I_2) \\ &= \mathrm{O}(L). \end{aligned}$$

Consequently, it comes to a conclusion that any arbitrary-sized $I_1 \times I_2$ image can be encoded (partitioned) into O(L) Hilbert curves (quadrants) where $L = \max(I_1, I_2)$. The proof is completed. □

3.2. Concatenating encoded Hilbert curves to form a complete Hilbert curve

By Theorem 1, it is known that any arbitrary-sized $I_1 \times I_2$ image can be encoded into O(L) quadrants where $L = \max(I_1, I_2)$ and each quadrant can be encoded into a corresponding Hilbert curve. We present how to concatenate these encoded Hilbert curves to obtain a complete Hilbert curve.

Definition 2. After encoding one square sub-image into a set of Hilbert curves, a subset of Hilbert curves, in which each Hilbert curve is with the same resolution j, is called an H_j-set.

As shown in Fig. 7, assume the given square sub-image has been partitioned into a set of Hilbert curves. For convenience, all the Hilbert curves in the H_j-set are arranged in clockwise orientation. For Fig. 7, by Definition 2, we have H_r-set $= < H_r^1 >$, H_{r-1}-set $= < H_{r-1}^1, H_{r-1}^2, \ldots, H_{r-1}^5 >$, and H_{r-2}-set $= < H_{r-2}^1, H_{r-2}^2, \ldots, H_{r-2}^{13} >$. Therefore, Fig. 7 can be represented by the set $\{H_r\text{-set}, H_{r-1}\text{-set}, H_{r-2}\text{-set}\}$.

Definition 3. After reversing the scanning order of the Hilbert curves in the H_j-set, the reversed H_j-set is called the $\overline{H}_j$-set.

It is known that in Fig. 7, H_{r-1}-set is equal to $< H_{r-1}^1, H_{r-1}^2, \ldots, H_{r-1}^5 >$. By Definition 3, the reversed H_{r-1}-set is equal to $\overline{H}_{r-1}$-set $=< H_{r-1}^5, H_{r-1}^4, \ldots, H_{r-1}^1 >$ and the reversed H_{r-2}-set is equal to $\overline{H}_{r-2}$-set $= \{H_{r-2}^{13}, H_{r-2}^{12}, \ldots, H_{r-2}^1\}$. Following Definition 2 and 3, we present how to concatenate all the encoded Hilbert curves in the square sub-image. Suppose one square sub-image has been represented by $\{H_r\text{-set}, H_{r-1}\text{-set}, \ldots, H_{r-k}\text{-set}\}$.

Definition 4 (*NW–NE Concatenation Strategy*). When the entrance point is at the north–west corner of H_r-set and the exit point is at the north–east corner of H_r-set, starting from the H_r-set, we first concatenate its exit point with the entrance point of H_{r-1}-set. Then we concatenate the exit point of H_{r-1}-set with the entrance point of $\overline{H}_{r-2}$-set, and so on.

Definition 5 (*NW–SW Concatenation Strategy*). When the entrance point is at the north–west corner of H_r-set and the exit point is at the south–west corner of H_r-set, starting from the H_r-set, we first concatenate its exit point with the entrance point of $\overline{H}_{r-1}$-set. Then we concatenate the exit point of $\overline{H}_{r-1}$-set with the entrance point of H_{r-2}-set, and so on.

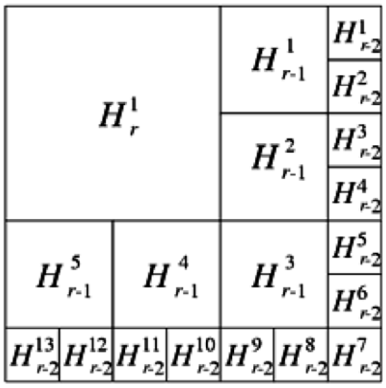

Fig. 7. One example for packing a subset of Hilbert curves into an H_j-set, each Hilbert curve with resolution j.

From Lemma 2, it is known that we partition the input image into a set of Hilbert curves from NW direction to SE direction. On the contrary, if we partition the input image into a set of Hilbert curves from the SE direction to the NW direction, instead of using the NW–NE (NW–SW) concatenation strategy, we utilize the SE–SW (SE–SN) concatenation strategy.

Following Definition 4 (Definition 5), applying the NW–NE (NW–SW) concatenation strategy to Fig. 7, the resulting concatenated configuration is shown in Fig. 8a (Fig. 8b). Based on the NW–NE concatenation strategy, Fig. 8a indicates that the entrance (exit) point of the resulting concatenated configuration of Fig. 7 is at the upper-left (upper-right) corner of H_r^1 (H_{r-2}^1). From Fig. 8b, the entrance (exit) point of the resulting concatenated configuration of Fig. 7 is at the upper-left (lower-left) corner of H_r^1 (H_{r-2}^{13}).

By Lemma 2, either applying the NW–NE concatenation strategy or the NW–SW concatenation strategy to the partitioned square sub-image, we have the following result.

Lemma 3. *For any partitioned square sub-image, the encoded Hilbert curves can be concatenated to obtain a complete Hilbert curve.*

Given a 7×7 square sub-image, by Lemma 3, the constructed complete Hilbert curve by applying the NW–NE (NW–SW) concatenation strategy to the given square sub-image is shown in Fig. 9a (Fig. 9b).

After describing the NW–NE concatenation strategy and the NW–SW concatenation strategy to construct a complete Hilbert curve for any partitioned square sub-image, we further generalize it to construct a complete Hilbert curve for any arbitrary-sized $I_1 \times I_2$ image. For convenience, we still assume $I_1 < I_2$. As mentioned in Theorem 1, let $N_1 = \lfloor I_2/I_1 \rfloor$ denote the number of $I_1 \times I_1$ square sub-images which are partitioned from the given $I_1 \times I_2$ image. Let $L_1 = N_1 \times I_1 = \lfloor I_2/I_1 \rfloor \times I_1$, then the $I_1 \times (I_2 - L_1)$ sub-image denotes the remaining image after removing these N_1 square sub-images which each square sub-image is of size $I_1 \times I_1$. Next, the remaining $I_1 \times (I_2 - L_1)$ sub-image is partitioned into one $L_2 \times (I_2 - L_1)$ sub-image and one $(I_1 - L_2) \times (I_2 - L_1)$ sub-image where $L_2 = \lfloor I_1/(I_2 - L_1) \rfloor \times (I_2 - L_1)$. The $(I_2 - L_1) \times L_2$ sub-image is further partitioned into $N_2 = L_2/(I_2 - L_1)$ square sub-images in which each square sub-image is of size $(I_2 - L_1) \times (I_2 - L_1)$. For example, as shown in Fig. 10, suppose one given image is partitioned into two maximal square sub-images, namely S_1 and S_2, and two middle square sub-images, namely S_3 and S_4, and so on. For exposition, the two square sub-images, S_1 and S_2, are called horizontally oriented square sub-images and the two square sub-images, S_3 and S_4, are called vertically oriented square sub-images.

The following definition is used to characterize four types of one square sub-image in the partitioned arbitrary-sized image.

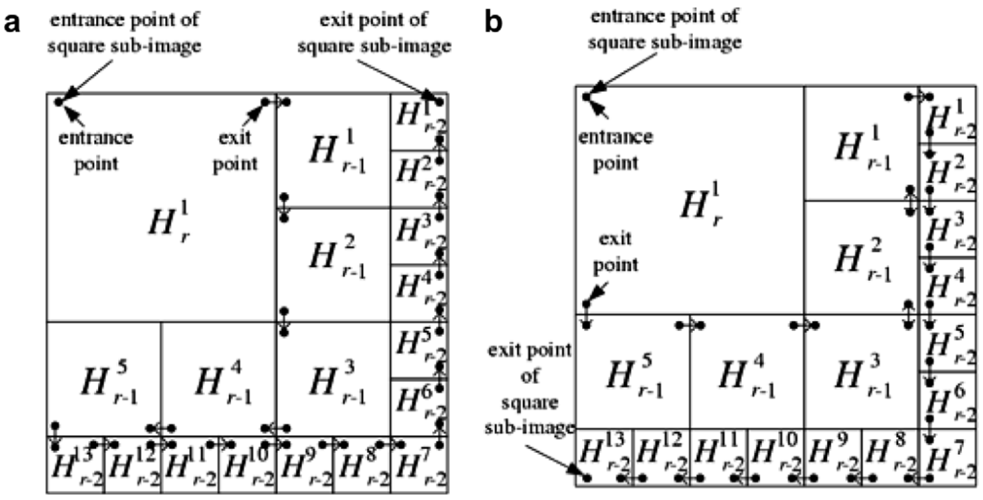
a
entrance point of square sub-image
exit point of square sub-image
entrance point
exit point
H_r^1
H_{r-1}^1 H_{r-1}^2 H_{r-1}^3 H_{r-1}^4 H_{r-1}^5
H_{r-2}^1 … H_{r-2}^{13}
b
entrance point of square sub-image
entrance point
exit point
exit point of square sub-image

Fig. 8. Two concatenation strategies for Fig. 7. (a) Applying the NW–NE concatenation strategy to Fig. 7. (b) Applying the NW–SW concatenation strategy to Fig. 7.

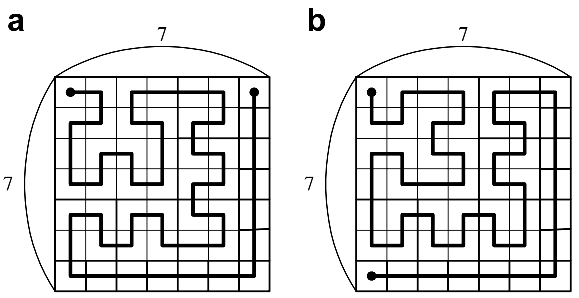

Fig. 9. Two complete Hilbert curves constructed from the 7×7 partitioned square sub-image. (a) The complete Hilbert curve applying the NW–NE concatenation strategy. (b) The complete Hilbert curve applying the NW–SW concatenation strategy.

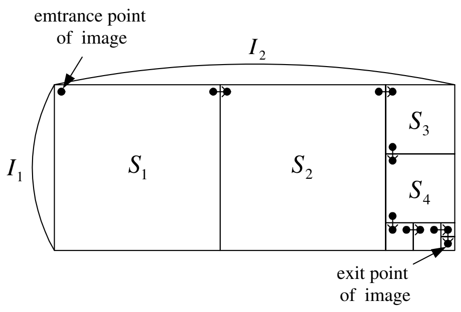

Fig. 10. One example used to construct the complete Hilbert curve from the given arbitrary-sized image.

Definition 6. If the number of H_J-set's in the horizontally oriented square sub-image is even (odd), the horizontally oriented square sub-image is called the H-EVEN (H-ODD) square sub-image. If the number of H_J-set's in the vertically oriented square sub-image is even (odd), the vertically oriented square sub-image is called the V-EVEN (V-ODD) square sub-image.

Returning to the 7×17 image as shown in Fig. 5a and the corresponding one in Fig. 10, by Definition 6, the first partitioned horizontally oriented 7×7 square sub-image is S_1; the number of H_J-set's is 3 and the three H_J-set's are denoted by H_2, H_1, and H_0. Therefore, for the square sub-image S_1, it is called the H-ODD square sub-image. Similarly, the second partitioned square sub-image S_2 is also called the H-ODD square sub-image. Further, the two partitioned 3×3 square sub-images, S_3 and S_4, are the V-EVEN square sub-images since the number of H_J-set's in each vertically oriented 3×3 square sub-image is even.

After characterizing the type of each partitioned square sub-image, by Lemma 3, we have the following result.

Theorem 2. *Given a partitioned arbitrary-sized image, for the H-ODD square sub-image or V-EVEN square sub-image, by Definition 4, the NW–NE Concatenation Strategy is used to concatenate the set of the related Hilbert curves. For the H-EVEN square sub-image or V-ODD square sub-image, by Definition 5, the NW–SW Concatenation Strategy is used to concatenate the set of the related Hilbert curves. After concatenating the exit point of the former square sub-image with the entrance point of the current square sub-image, the complete Hilbert curve can be constructed from the arbitrary-sized image.*

By Theorem 2, the complete Hilbert curve constructed from Fig. 5a is illustrated in Fig. 11.

3.3. The proposed coding scheme

In this section, the encoding and decoding scheme for one pixel is presented. Given the (x, y)-coordinate of one pixel in the input arbitrary-sized image, the encoding scheme is to find the corresponding Hilbert order.

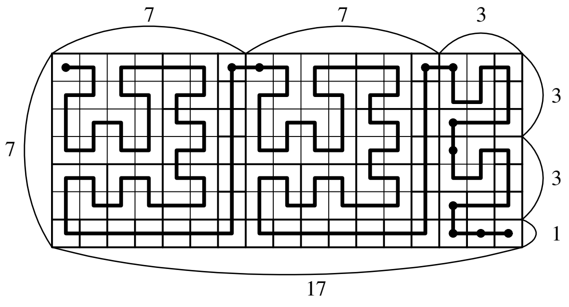

Fig. 11. Constructing the complete Hilbert curve from Fig. 5a.

On the contrary, given a Hilbert order of one pixel, the decoding scheme is to find the (x,y)-coordinate of that pixel.

Assume one arbitrary-sized image has been partitioned into k quadrants. According to the scanned quadrants in the input partitioned arbitrary-sized image, let the sequence of the scanned quadrants be represented by $< ((x,y)_{e_1}, o_1, r_1), ((x,y)_{e_2}, o_2, r_2), \ldots, ((x,y)_{e_k}, o_k, r_k) >$ where $(x,y)_{e_i}$ denotes the (x,y)-coordinate of the entrance point e_i in the quadrant q_i; o_i denotes the orientation of the quadrant q_i; r_i denotes the resolution of the encoded Hilbert curve of the quadrant q_i for $1 \leqslant i \leqslant k$. As shown in Fig. 12a–d, one quadrant may have four orientations, namely orientation-0, 1, 2, and 3, respectively.

Given the (x,y)-coordinate of one pixel in one partitioned $I_1 \times I_2$ image, the sequence of the scanned quadrants is first searched one by one to find the quadrant to which the pixel belongs. The search is bounded by O(k) time. Assume the scanned quadrant is $((x,y)_{e_i}, o_i, r_i)$ and (x', y') denotes the (x,y)-coordinate of one pixel in the quadrant q_i. According to Table 1 associated with the orientation information of the quadrant q_i, the (x,y)-coordinate of that pixel, (x', y'), can be determined in O(1) time.

Based on the determined (x', y')-coordinate of the pixel, the Hilbert order of the pixel in the quadrant q_i, h_i can be calculated by Liu and Schrack's coding scheme [27] (see Eqs. (1) and (2)). Upon calculating the Hilbert order of the pixel in the quadrant q_i, by Theorem 2, the Hilbert order of the pixel along the complete Hilbert curve is given by

$$\begin{aligned} h &= \sum_{j=1}^{i-1} (2^{r_j} \times 2^{r_j}) + h_i \\ &= \sum_{j=1}^{i-1} 4^{r_j} + h_i. \end{aligned} \tag{5}$$

Since the calculation of h_i takes $O(r_i) = O(\log U)$ time based on Liu and Schrack's encoding scheme where the size of each quadrant is bounded by $U \times U$ where $U = \min(I_1, I_2)$. Consequently we have the following result because at most O(k) quadrants must be examined.

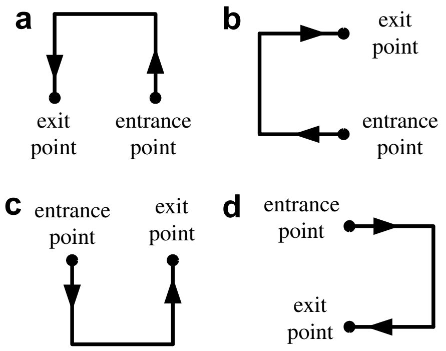

Fig. 12. Four orientations of one quadrant: (a) orientation-0, (b) orientation-1, (c) orientation-2, and (d) orientation-3.

Table 1
The (x, y)-coordinate of one pixel in the ith quadrant

Quadrant's orientation	(x', y')
0	$(x_{e_i} - x, y_{e_i} - y)$
1	$(y_{e_i} - y, x_{e_i} - x)$
2	$(x - x_{e_i}, y - y_{e_i})$
3	$(y - y_{e_i}, x - x_{e_i})$

Theorem 3. *Given an arbitrary-sized $I_1 \times I_2$ image, if the (x, y)-coordinate of one pixel is known, the Hilbert order of that pixel can be encoded in $O(k + \log U)$ time.*

For example, assume the (x, y)-coordinate of one pixel is $(10, 3)$ in the 6×16 image as shown in Fig. 13. For Fig. 13, by Theorem 2, the sequence of the scanned quadrants is $< ((0,0)_{e_1}, 3, 2), ((0,4)_{e_2}, 2, 1), \ldots, ((15,4)_{e_{15}}, 2, 1) >$. According to the above discussion, we can find that the quadrant $q_{11} = ((11,3)_{e_{11}}, 1, 1)$ to which the pixel with coordinate $(10, 3)$ belongs. According to the four orientations defined in Fig. 12, the orientation type of the quadrant q_{11} is orientation-1. By Table 1, the $(x\prime, y\prime)$-coordinate of the pixel in the quadrant q_{11} is $(0, 1)(= (y_{e_{11}} - y, x_{e_{11}} - x) = (3 - 3, 11 - 10))$. By Eqs. (1) and (2), the Hilbert order of the pixel in the quadrant q_{11} is 1. Further, by Eq. (5), the Hilbert order of the pixel along the complete Hilbert curve in Fig. 13 is calculated by

$$\begin{aligned} h &= \sum_{j=1}^{10} (2^{r_j} \times 2^{r_j}) + 1 \\ &= (16 + 5 \times 4 + 16 + 3 \times 4) + 1 \\ &= 65. \end{aligned}$$

In the proposed decoding scheme, the sequence of the scanned quadrants represented by $< ((x,y)_{e_1}, o_1, r_1), ((x,y)_{e_2}, o_2, r_2), \ldots, ((x,y)_{e_k}, o_k, r_k) >$ is used again. Based on the obtained Hilbert order of the pixel by Eq. (5), the Hilbert order h_i of the pixel in the quadrant q_i can be calculated and it can be done in O(k) time. Using the Hilbert order h_i as the input, the (x', y')-coordinate of the pixel in the quadrant q_i can be calculated by Eqs. (3) and (4) and it can be done in O(U) time. Finally, the (x, y)-coordinate of the pixel in the arbitrary-sized image can be determined by Table 1. The total time complexity of our proposed decoding scheme is O($k + \log U$) time.

4. Memory-saving Hilbert curve representation: HCGL

By Theorem 2, it is known that any arbitrary-sized image can be encoded into a complete Hilbert curve which concatenates a set of Hilbert curves. Two pixels are said to be two neighboring pixels if their Hilbert orders are consecutive. From this locality property, the gray levels of two neighboring pixels may be rather similar. Based on the locality property, Gouraud shading technique [16], recursive binary partition scheme, and S-tree data structure [22], this section presents a modified Hilbert curve representation called Hilbert curve-based gray levels (HCGL) representation. Because of employing the recursive binary partition scheme

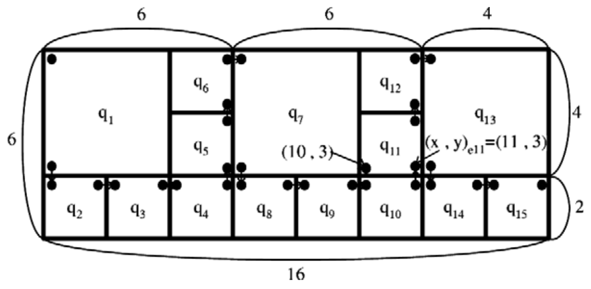

Fig. 13. One example for calculating the Hilbert order of one pixel along the complete Hilbert curve.

and S-tree data structure, experimental results indicate that under the same image quality, the proposed HCGL representation has better compression effect when compared to the previous two representations – the split point approximation [19] and the zero order interpolation [25].

Based on the constructed complete Hilbert curve mentioned in Section 3, we scan all the pixels along the complete Hilbert curve and generate a 3-D curve consisting of gray levels of all the scanned pixels. The scanned pixels are called the HCGL. According to the binary recursive subdivision principle on the HCGL, the HCGL is partitioned into two segments first. If any segment is not homogeneous, it is subdivided into two smaller segments until all the homogeneous segments are obtained. During subdividing the HCGL, a segment is called a homogeneous segment if and only if the estimated grey level of each of the pixels in the segment is in some vicinity of its real grey level. For any segment of HCGL, suppose the Hilbert orders of the end-points of the segment are i_1 and i_2 and their corresponding grey levels are g_1 and g_2, respectively. Using the Gouraud shading technique, i.e. the linear interpolation, the estimated grey level of the pixel with Hilbert order i is calculated as

$$g_{\text{est}}(i) = g_1 + \frac{g_2 - g_1}{i_2 - i_1}(i - i_1). \tag{6}$$

Given a specified error tolerance ε, if the following image quality condition holds:

$$|g(i) - g_{\text{est}}(i)| \leqslant \varepsilon$$

for every Hilbert order i, $i_1 \leqslant i \leqslant i_2$, then the segment from the Hilbert order i_1 to the Hilbert order i_2 is called the homogeneous segment. In fact, the above segmentation process can be emulated by a binary tree structure called the bintree.

In our proposed HCGL representation, we traverse the bintree in breadth-first-search manner, at each time, we emit a ‘0’ when an internal node is encountered; emit a ‘1’ when a leaf node is encountered. After traversing the bintree, the sequence of these ordered binary values is saved in the linear-tree table. Meanwhile, at each time, we do nothing when an internal node is encountered. When a leaf node is traversed, we emit the grey level of the starting point in the concerning homogeneous segment. The sequence of these emitted grey levels is stored in the color table. The constructed linear-tree and color tables form the proposed HCGL representation. The two tables used in the proposed HCGL representation are some similar to the S-tree representation [22].

For example, suppose the given complete Hilbert curve has been partitioned into the set of homogeneous segments as shown in Fig. 14a according to the above bintree decomposition scheme. The corresponding bintree is illustrated in Fig. 14b. The final HCGL representation for Fig. 14b is listed below:

linear-tree table: 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1
color table: g_e, g_h, g_k, g_l, g_a, g_b, g_c, g_d, g_f, g_g, g_i, g_{j_1}, g_{j_2}

where the symbol g_e denotes the grey level of the leftmost endpoint in the segment e; the symbol g_h denotes the grey level of the leftmost endpoint in the segment h, and so on. Specifically, g_{j_1} and g_{j_2} denote the grey levels of two endpoints of the segment j.

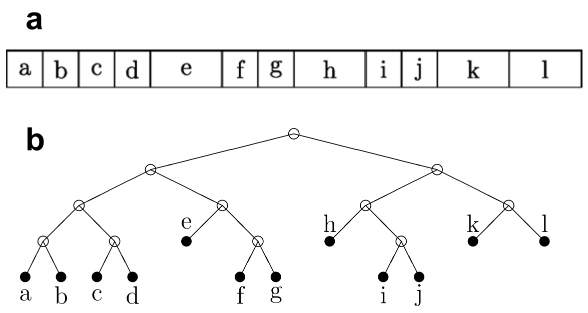

Fig. 14. One example of building up the proposed HCGL representation. (a) The homogeneous segments for one complete Hilbert curve. (b) The corresponding bintree.

Given a complete Hilbert curve, assume there are n_1 homogeneous segments based on the proposed HCGL representation. Since each grey level in the color table needs 8 bits, it needs $8n_1$ bits for saving the color table. It is easy to verify that the linear-tree table needs $2n_1 - 1$ bits. Consequently, the proposed HCGL representation needs $10n_1 - 1$ bits.

In the previous Hilbert curve representations [19,25], there is one common disadvantage that the lengths of some homogeneous segments may be rather long. According to our experiments, seven bits are needed to encode the length of one segment. In the zero order interpolation [25], excepting the last segment, (g, l) is used to represent each segment where g denotes the mean value of the segment with length l. Specifically, (g_1, g_2, l) is used to represent the last segment where g_1 (g_2) denotes the grey level of the leftmost (rightmost) endpoint of the segment with length l. If there are n_2 homogeneous segments in the Hilbert curve representation, it needs $15n_2+8$ $(= 7n_2 + 8(n_2 + 1))$ bits. In the split point approximation [19], suppose there are n_3 segments, then each segment is represented by (g_1, g_2, l) and the grey level of each pixel in the segment is approximated by linear interpolation. Consequently, the split point approximation method needs $23n_3(= (7 + 8 + 8)n_3)$ bits.

When comparing the proposed HCGL representation with the previous two Hilbert curve representations, the proposed HCGL representation needs $10n_1 - 1$ bits while the previous zero order interpolation (split point approximation) needs $15n_2 + 8$ $(23n_3)$ bits. We thus have the following result.

Theorem 4. *When $n_1 < 1.5n_2 + 1(n_1 < 2.3n_3)$, the memory requirement of the proposed HCGL representation is less than the previous zero order interpolation method (split point approximation method).*

Based on four testing images, experimental results (see Section 6) demonstrate that under the same image quality, the proposed HCGL representation has better compression performance when compared to the zero order interpolation [25] and the split point approximation [19].

5. Window query application

Previously, based on a $2^r \times 2^r$ image, Chung et al. [10] presented a window query algorithm for image database application [1,31,32]. In this section, based on the proposed results mentioned in Sections 3 and 4, we extend the previous window query algorithm from $2^r \times 2^r$ image domain to arbitrary-sized image domain.

Definition 7 (*Maximal Quadtree Blocks* [2]). Given any query window, the set of maximal quadtree blocks can be obtained by decomposing the window into a set of quadrants according to the quadtree decomposition over the $2^r \times 2^r$ image domain.

For example, we are given a 16×16 image domain and a query window $W = w(x, y, n_1, n_2) = w(11, 10, 6, 5)$ as shown in Fig. 15 where x and y represent the x- and y-coordinates of its upper-left corner; n_1 is the height and n_2 is the width of the query window. By Definition 7, the set of maximal quadtree blocks is $\{B_1, B_2, \ldots, B_9\}$ based on the quadtree decomposition of the query window over the 16×16 image domain. For decomposing a query window into maximal quadtree blocks over the $2^4 \times 2^4$ image domain, Aref and Samet [2] presented the first algorithm, then two different optimal algorithms [34,40] for determining maximal quadtree blocks were presented.

Definition 8. Given an $n_1 \times n_2$ window as a query, the window query problem is to output the codes of the concerning maximal quadtree blocks.

For any arbitrary-sized image, our proposed window query algorithm is listed below.

ALGORITHM: WINDOW QUERY

Input: The query window W and the proposed HCGL representation for representing the given $I_1 \times I_2$ image.

Output: The corresponding codes of the concerning maximal quadtree blocks.

Step 1. (Constructing the sequence of the scanned quadrants) From the height I_1 and width I_2 of the image domain, by Theorem 1, the sequence of the k scanned quadrants is constructed. This step takes $O(k)$ time where k is the number of the quadrants.

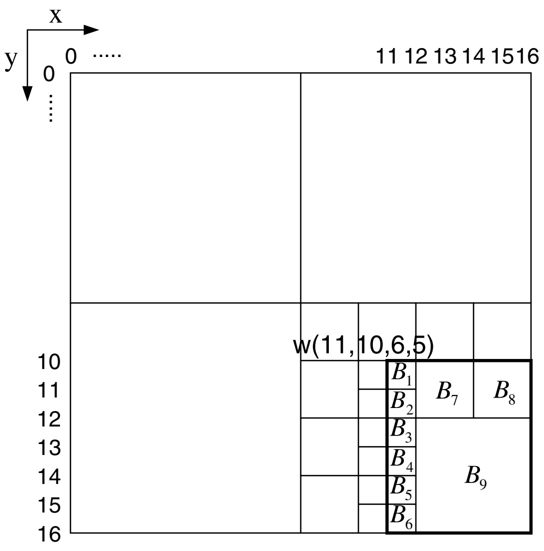

Fig. 15. An example for decomposing a query window into a set of maximal quadtree blocks.

Step 2. *(Finding the quadrants intersected with the query window W)* The sequence of the k scanned quadrants is searched one by one to find the quadrants intersected with the query window W. Then we collect the intersected sub-windows and the corresponding intersected quadrants. For exposition, assume these intersected (sub-window, quadrant)-pairs are $\{(W_1, q'_1), (W_2, q'_2), \ldots, (W_m, q'_m)\}$. This step takes O($k$) time.

Step 3. *(Generating the maximal quadtree blocks)* For these intersected (sub-window, quadrant)-pairs, applying any maximal quadtree blocks partition strategy [34,40] to (W_i, q'_i), $1 \leqslant i \leqslant m$, it takes O($M$) time to generate all the M maximal quadtree blocks.

Step 4. *(Outputting the codes)* For $1 \leqslant i \leqslant m$, apply the code-generation method presented in [10] to generate the code for each (W_i, q'_i)-pair. It takes $O(M \log L + P)$ time to generate all the codes where $L = \max(I_1, I_2)$ and P denotes the number of output codes.

According to the time analysis of each step in the above algorithm, our proposed window query algorithm takes $O(M \log L + P)(= O(k) + O(M) + O(M \log U + P))$ time, commonly $k \leqslant M$.

Given an 18×22 image domain as shown in Fig. 16, after performing Step 1, the image domain can be partitioned into a set of quadrants which is represented by the sequence $< ((0,0)_{e_1}, 3, 4), ((0,16)_{e_2}, 2, 1), \ldots, ((20,16)_{e_{24}}, 2, 1) >$. Next, a query window $W = w(11,10,7,8)$ shown in Fig. 16 is used to perform the window query operation in the 18×22 image domain. After performing Step 2, the intersected quadrants are $((0,0)_{e_0}, 3, 4), ((10,16)_{e_6}, 2, 1), \ldots,$ and $((18,16)_{e_{22}}, 2, 1)$ and their corresponding sub-windows are $w(11,10,6,5), w(11,16,1,1), \ldots,$ and $w(18,16,1,1)$, respectively.

After performing Step 3, for each (W_i, q'_i)-pair, the maximal quadtree blocks partition strategy is used to obtain the set of the maximal quadrant blocks. For instance, the (W_1, q'_1)-pair denotes the intersected $2^4 \times 2^4$ quadrant and the intersected 6×5 query window where $W_1 = w(11,10,6,5)$ and $q'_1 = ((0,0)_{e_1}, 3, 4)$. Over the $2^4 \times 2^4$ image domain, the maximal quadtree blocks partition strategy can decompose the query window W_1 into a set of maximal quadtree blocks $\{B_1, B_2, \ldots, B_9\}$ as shown in Fig. 15. Fig. 17 illustrates the resulting maximal quadtree blocks. After performing Step 4, the code-generation method [10] is used to generate the maximal Hilbert order and the minimal Hilbert order of the Hilbert curve in each generated maximal quadtree block. Next, the set of all the generated maximal Hilbert orders and the minimal Hilbert orders are sorted [14]. Among these sorted maximal Hilbert orders and the minimal Hilbert orders, if the difference of the Hilbert orders of the entrance point of the current maximal quadtree block and the exit point of the previous maximal quadtree block is exactly one, the two consecutive Hilbert curves in the

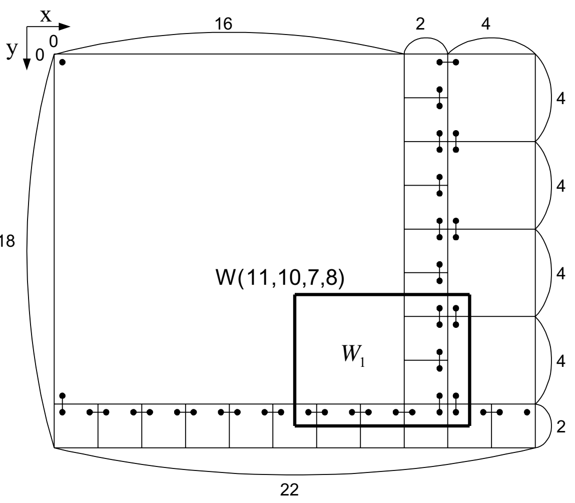

Fig. 16. A query window $W = w(11, 10, 7, 8)$ and the given 18×22 image domain.

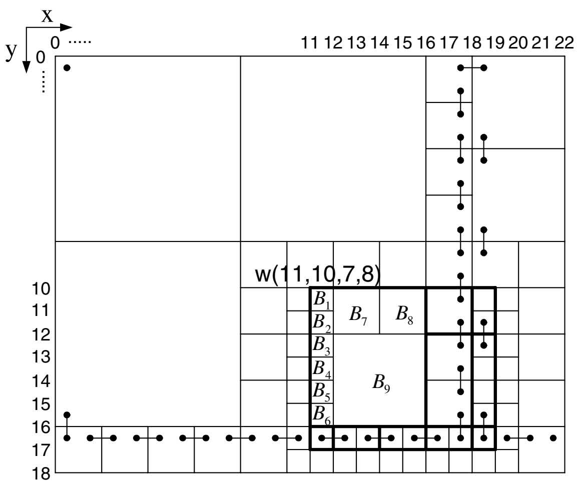

Fig. 17. The resulting maximal quadtree blocks.

two maximal quadtree blocks can be merged to one longer Hilbert curve in order to reduce the number of the Hilbert orders used. Only the Hilbert orders of the entrance point of the first maximal quadtree block and the exit point of the last maximal quadtree block are needed to represent the new merged maximal quadtree block if some maximal quadtree blocks can be merged. The maximal Hilbert order and the minimal Hilbert order of each longer Hilbert curve are considered as two search keys. Then, the binary search [14] is used to find the corresponding output codes [10]. Consequently, the corresponding output codes of all maximal quadtree blocks in the query window are united as the final output codes.

6. Experimental results

As shown in Fig. 18, two CIF format images, namely the Akiyo image and the mother image, each with size 288×352, and two 256×352 images, namely the Susie image and the dancer image, are used to evaluate the performance among the related algorithms. First we compare the compression and image quality performance

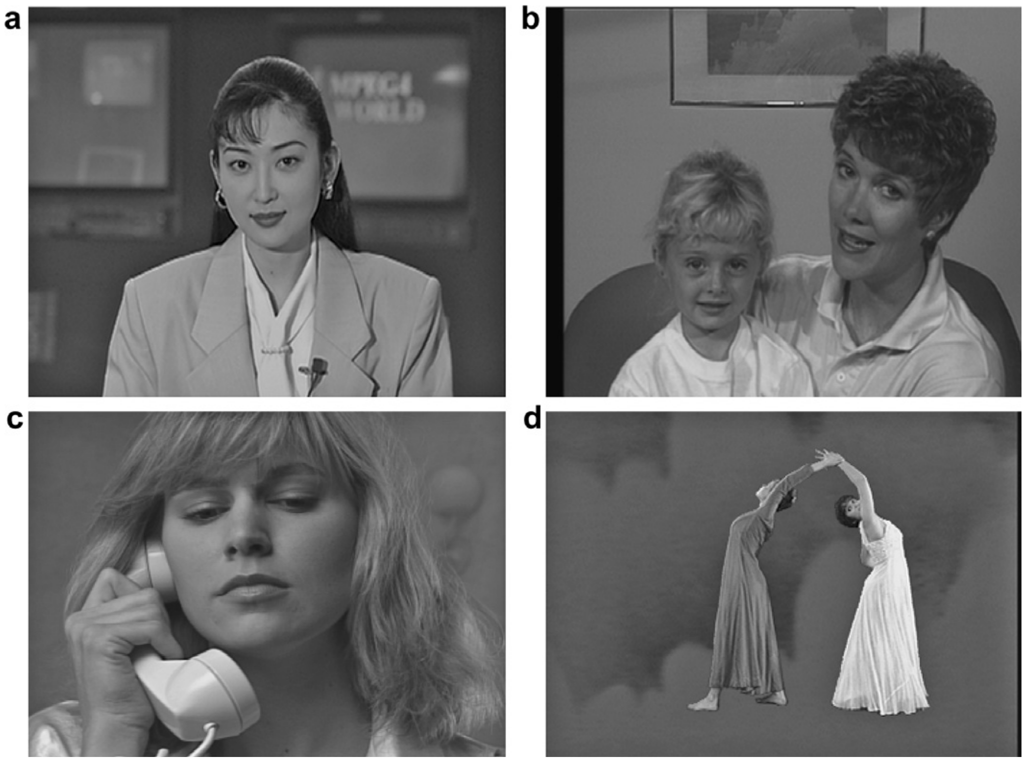

Fig. 18. Four testing images. (a) Akiyo image. (b) Mother image. (c) Susie image. (d) Dancer image.

between the proposed snake scan-based coding scheme and the raster scan-based coding scheme on the proposed HCGL representation. Under the same image quality, we next compare the compression performance among the proposed snake scan-based HCGL representation, the zero order interpolation [25], and the split point approximation [19]. Finally, one aerial photo, namely the San Francisco image, is used as the testing image in our proposed window query application. Moreover, the execution time performance between the proposed window query algorithm and the naive window query algorithm is evaluated. All experiments are performed on the Mobile Intel Pentium 4 microprocessor with 1.4 G MHz and 256 MB RAM. The operating system is MS-Windows XP and all programs are developed by Borland C++ Builder.

For four testing images associated with $\varepsilon = 20$, Table 2 illustrates the compression and image quality performance comparison between our proposed snake scan-based HCGL scheme and the raster scan-based HCGL scheme. The image quality performance is measured in terms of PSNR and the PSNR measure is defined by

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right)$$

Table 2
Compression and PSNR performance comparison between the proposed snake scan-based HCGL scheme and the raster scan-based HCGL scheme

$\varepsilon = 20$	Number of segments		BPP		PSNR	
Image	Snake scan	Raster scan	Snake scan	Raster scan	Snake scan	Raster scan
Akiyo	7247	12,509	0.71	1.23	32.91	32.83
Mother	6698	11,348	0.67	1.12	32.90	32.47
Susie	4903	8644	0.54	0.96	31.45	31.03
Dancer	5091	10,056	0.57	1.12	32.33	34.17

where MSE is the mean square error between the decompressed image and the original image. It is observed that for three of the four testing images, the image quality of our proposed snake scan-based coding scheme outperforms that of the raster scan-based coding scheme. For four testing images, the compression effect of our proposed snake scan-based coding scheme, in terms of BPP, outperforms that of the raster scan-based coding scheme since the number of segments in the proposed snake scan-based coding scheme is much less than that of the raster scan-based one. With respect to five ε's, $\varepsilon = 5, 10, 15, 20, 25$, under the same PSNR, Fig. 19 shows that the proposed snake scan-based HCGL scheme has the higher compression effect.

Given the same error tolerances, $\varepsilon = 5, 10, 15, 20$, and 25, Table 3 illustrates the performance comparison among the proposed snake scan-based HCGL representation, the zero order interpolation [25], and the split point approximation [19] for Akiyo image. In order to reduce the total bits needed, we restricted the longest length in the zero order interpolation and the split point approximation to be 127, i.e. the length is represented by 7 bits. Although the number of segments required in the HCGL representation is more than that of the zero order interpolation and the split point approximation, the memory requirement of the HCGL representation is still less than that of the zero order interpolation and that of the split point approximation due to the memory-saving advantage of S-tree structure. For Akiyo image and mother image, Fig. 20 shows the compression performance comparison among the proposed snake scan-based HCGL representation, the zero order interpolation, and the split point approximation. It is observed that under the same PSNR, the compression effect of the HCGL representation is better than that of zero order interpolation and the split point approximation. Moreover, as shown in Table 3, the execution time of the HCGL representation is faster than that of zero order interpolation and the split point approximation.

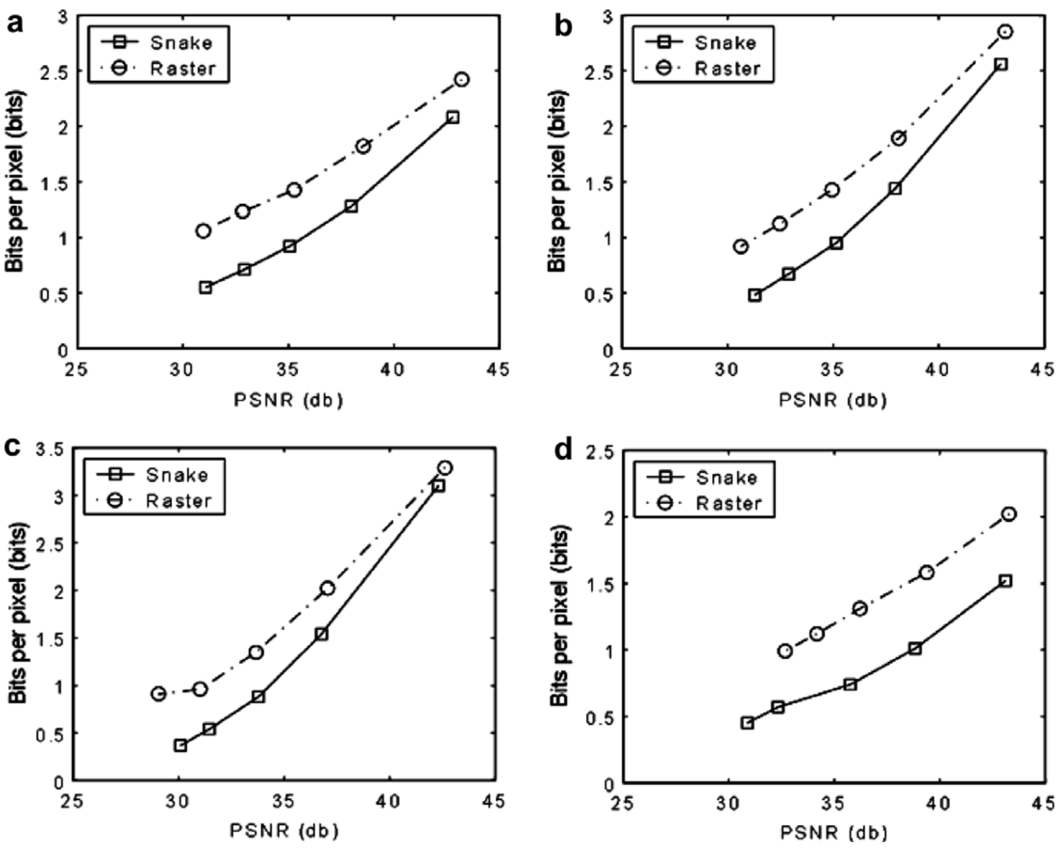

Fig. 19. The compression performance comparison between the proposed snake scan-based HCGL scheme and the raster scan-based HCGL scheme. (a) For Akiyo image; (b) for mother image; (c) for Susie image; (d) for dancer image.

Table 3
Number of segments and execution time performance comparison among the HCGL representation, zero order interpolation, and split point approximation for Akiyo image

ε	Number of segments			Executive time (s)		
	HCGL	Zero	Split	HCGL	Zero	Split
5	21,103	15,031	11,978	0.17	4.36	8.58
10	12,926	9714	7572	0.14	4.25	8.31
15	9357	7554	5692	0.12	4.20	8.21
20	7247	6301	4613	0.11	4.17	8.11
25	5553	5565	3879	0.10	4.15	8.06
Average	11,237	8833	6747	0.13	4.23	8.25

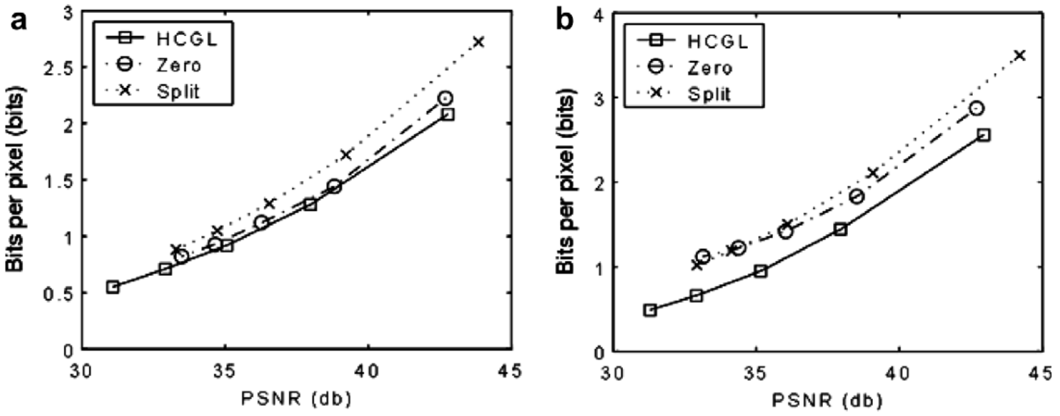

Fig. 20. The compression performance comparison among the proposed snake scan-based HCGL scheme, the zero interpolation, and the split point approximation. (a) For Akiyo image; (b) for mother image.

Based on the our proposed window query algorithm, an aerial photo of one GIS application is used as an example for window query application. In GIS application, the image size is usually very huge such that performing operations on the related images is very time-consuming. Besides, from the window query viewpoint, the high-frequency content of the image in GIS application is more important than the low-frequency content. Since our proposed HCGL representation can preserve the high-frequency content of the given image quite well, the proposed HCGL-based window query algorithm is suitable for the aerial photo in GIS application. As shown in the upper-middle image of Fig. 21, the testing image is the San Francisco aerial photo whose size is 896×1024. Under the error tolerance $\varepsilon = 10$, the testing San Francisco aerial photo has been compressed in terms of the proposed HCGL representation. Four window queries, each represented by the position of left-top corner, the width and the length, are given and are represented by $((300, 22), 200, 200)$, $((2, 373), 300, 200)$, $((549, 622), 200, 200)$, and $((700, 571), 200, 300)$. According to the proposed window query algorithm, the four resulting sub-images are depicted by the four sub-images which are around the testing image.

Finally, based on the arbitrary-sized image, we compare the execution time performance between the proposed window query algorithm and the naive algorithm. The input image is represented by using the proposed snake scan-based HCGL representation with error tolerance $\varepsilon = 20$. Six types of query window sizes are used and they are 10,000, 15,000, 20,000, 25,000, 30,000, and 35,000. For each specific window size, 100 query windows are generated by choosing the width and the height of the window randomly. Here the locations of the generated windows are also given randomly and it will affect the execution time required in our proposed window query algorithm. Based on four testing images, Table 4 illustrates the execution time performance comparison between the proposed window query algorithm and the naive algorithm. In Table 4, the time unit is microsecond. It is observed that the execution time of the proposed window query algorithm is less than that of naive algorithm significantly and the execution time improvement ratio ranges from 89% to 99%.

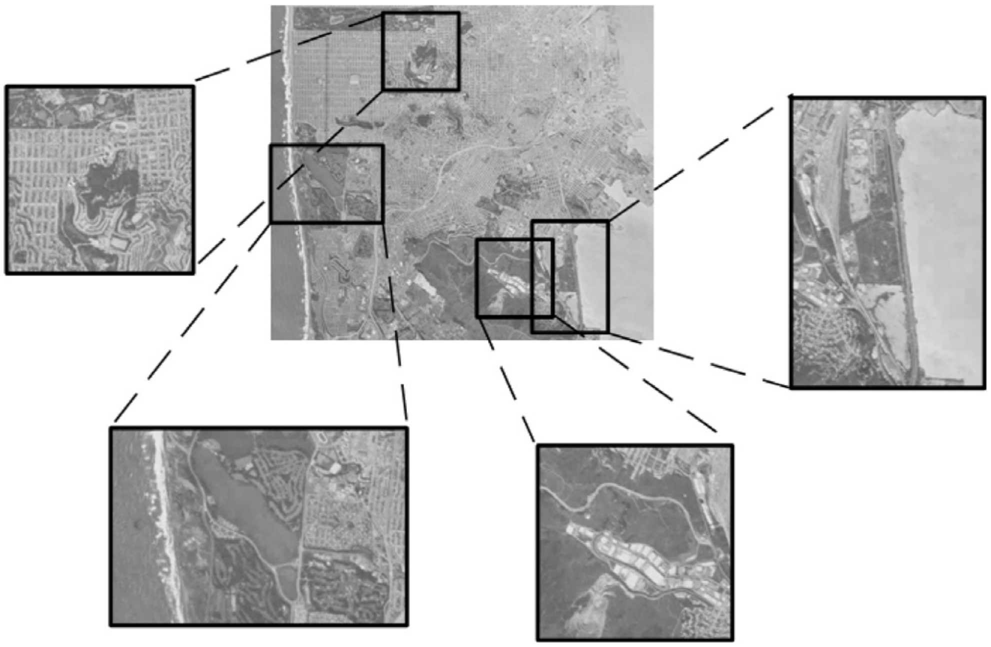

Fig. 21. One window query example for San Francisco aerial photo.

Table 4
Execution time performance comparison between the proposed window query algorithm and the naive algorithm

(ms)	Akiyo		Mother		Susie		Dancer	
Window size	T_{our}	T_{nav}	T_{our}	T_{nav}	T_{our}	T_{nav}	T_{our}	T_{nav}
10,000	8.61	86.92	9.22	85.93	4.41	87.03	3.11	85.82
15,000	11.92	130.18	7.92	130.69	9.11	134.30	5.81	129.99
20,000	6.22	174.15	9.10	175.75	8.51	175.26	12.52	174.05
25,000	6.41	217.91	13.22	231.74	7.40	221.22	13.51	219.00
30,000	13.92	266.98	10.11	267.69	12.52	280.21	8.81	261.57
35,000	6.01	311.05	3.60	319.06	19.43	314.65	8.91	307.93

7. Conclusions

This paper has presented the proposed snake scan-based algorithm for coding the Hilbert curve of an image with arbitrary size $I_1 \times I_2$. The proposed algorithm takes $O(k + \log U)$ time, where k denotes the number of the quadrants and $U = \min(I_1, I_2)$, for coding the Hilbert order of one pixel and it extends the coding scheme of Liu and Schrack from the quadrant domain to the arbitrary-sized image domain. Next the proposed memory-saving HCGL representation has been presented for representing the constructed complete Hilbert curve. The proposed HCGL representation can be constructed in $O(L^2 \log L)$ time. Experimental results show that the proposed HCGL representation outperforms the previous two representations – the split point approximation [19] and the zero order interpolation [25]. Finally, based on the arbitrary-sized image domain, a window query algorithm is presented and the proposed window query algorithm takes $O(kL + P)$ time where k denotes the number of maximal quadtree blocks and P denotes the number of output codes. The proposed window query algorithm is faster than the naive algorithm which needs $O(w_1 w_2 L + P)$ time where the query window is of size $w_1 \times w_2$. Under the same PSNR, experimental results demonstrate that our proposed HCGL representation outperforms some existing related algorithms in terms of execution-time requirement and BPP. In addition, our proposed window query algorithm has been justified in the window query application in the GIS application.

Recently, Wang et al. [42] presented an efficient globally adaptive pixel-decimation (GAPD) algorithm for block motion estimation. In their algorithm, for the current frame, the scanning order is based on the block-based raster scan. Upon scanning the current block, it is transformed into Hilbert curve-based sequence, and then the sequence is divided into some segments. Instead of using the block-based scanning order, it seems that our proposed snake scan-based algorithm for coding the Hilbert curve of an arbitrary-sized image can be applied to Wang et al.'s motion estimation algorithm in order to have better estimation accuracy and to generalize their result to the arbitrary-sized image domain. In addition, it is also an interesting research issue to extend the results of this paper to the other applications such as retrieval [38], region segmentation [12], moment computation [13], neighbor-finding, watermarking, alternative patterns, cost modeling of spatial operators [20], count queries [41], image coding, and so on.

References

[1] W.G. Aref, H. Samet, Efficient processing of window queries in the pyramid data, in: Proceedings of the 9th ACM-SIGMOD Symposium Principles Database Systems, 1990, pp. 265–272.

[2] W.G. Aref, H. Samet, Decomposing a window into maximal quadtree blocks, Acta Informatica 30 (5) (1993) 425–439.

[3] T. Asano, D. Ranjan, T. Roos, E. Welzl, Space-filling curves and their use in the design of geometric data structures, Theoretical Computer Science 181 (1) (1997) 3–15.

[4] T. Bially, Space-filling curves: their generation and their application to bandwidth reduction, IEEE Transactions Information Theory IT-15 (6) (1969) 658–664.

[5] S. Biswas, One-dimensional B–B polynomial and Hilbert scan for graylevel image coding, Pattern Recognition 37 (4) (2004) 789–800.

[6] N. Bourbakis, C. Alexopoulos, Picture data encryption using scan patterns, Pattern Recognition 25 (6) (1992) 567–581.

[7] G. Cantor, Ein Beitrag zur Mannigfaltigkeitslehre, Crelle Journal 84 (1878) 242–258.

[8] C.C. Chang, J.H. Jiang, Multiple disk allocation based on round-robin Hilbert curve for spatial data, International Journal of Information and Management Sciences 9 (1) (1998) 11–23.

[9] H.L. Chen, Y.I. Chang, Neighbor-finding based on space-filling curves, Information Systems 30 (3) (2005) 205–226.

[10] K.L. Chung, Y.H. Tsai, F.C. Hu, Space-filling approach for fast window query on compressed images, IEEE Transactions Image Processing 9 (12) (2000) 2109–2116.

[11] K.L. Chung, L.C. Chang, A novel two-phase Hilbert-scan-based search algorithm for block motion estimation using CTF data structure, Pattern Recognition 37 (7) (2004) 1451–1458.

[12] K.L. Chung, H.L. Huang, H.I. Lu, Efficient region segmentation on compressed grey images using quadtree and shading representation, Pattern Recognition 37 (8) (2004) 1591–1605.

[13] K.L. Chung, Y.W. Liu, W.M. Yan, A hybrid gray image representation using spatial- and DCT-based approach with application to moment computation, Journal of Visual Communication and Image Representation 17 (6) (2006) 1209–1226.

[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[15] G.R. Feng, L.G. Jiang, C. He, Y. Xue, Chaotic spread spectrum watermark of optimal space-filling curves, Chaos, Solitons and Fractals 27 (3) (2006) 580–587.

[16] J.D. Foley, A. Dam, S.K. Feiner, J.F. Hughes, Computer Graphics: Princiles and Practice, second ed., Addison-Wesley, New York, 1996.

[17] D. Hilbert, Über die stetige Abbildung einer Linie auf ein Flächenstuck, Mathematische Annalen 38 (1891) 459–460.

[18] H.V. Jafadish, Analysis of the Hilbert curve for representing two-dimensional space, Information Processing Letters 62 (1) (1997) 17–22.

[19] F.C. Jian, Hilbert curves and its applications on image processing, M.S. thesis, National Taiwan University, 1996.

[20] S. Jiang, B.S. Lee, Z. He, Cost modeling of spatial operators using non-parametric regression, Information Sciences 177 (2) (2007) 607–631.

[21] R. Johnsonbaugh, Discrete Mathematics, fifth ed., Prentice Hall, NJ, 2001.

[22] W. de Jonge, P. Scheuerman, A. Schijf, S^+-Tree: An efficient structure for the representation of large picture, CVGIP: Image Understanding 59 (3) (1994) 265–280.

[23] S. Kamata, M. Niimi, E. Kawaguchi, A method of an interactive analysis for multi-dimensional images using a Hilbert curve, IEICE Transactions J77-D-II (7) (1993) 1255–1264.

[24] S. Kamata, R.O. Eaxon, E. Kawaguchi, An implementation of the Hilbert scanning algorithm and its application to data compression, IEICE Transactions Information and Systems E76-D (4) (1993) 420–428.

[25] S. Kamata, M. Niimi, E. Kawaguchi, A gray image compression using a Hilbert scan, in: Proceedings of the 13th International Conference on Pattern Recognition, 1996, pp. 905–909.

[26] H. Lebesgue, Leçons sur l'Intégration et la Recherche des Fonctions Primitives, Gauthier-Villars, Paris, France, 1904.

[27] X. Liu, G.F. Schrack, Encoding and decoding the Hilbert order, Software Practice and Experience 26 (12) (1996) 1335–1346.

[28] X. Liu, G.F. Schrack, An algorithm for encoding and decoding the 3-D Hilbert order, IEEE Transactions Image Processing 6 (9) (1997) 1333–1337.

[29] X. Liu, Four alternative patterns of the Hilbert curve, Applied Mathematics and Computation 147 (3) (2004) 741–752.

[30] E.H. Moore, On certain crinkly curves, Transactions of American Mathematical Society 1 (1900) 72–90.
[31] E. Nardelli, Efficient secondary memory processing of window queries on spatial data, Information Sciences 84 (1–2) (1995) 67–83.
[32] E. Nardelli, G. Proietti, Time and space efficient secondary memory representation of quadtrees, Information Systems 22 (1) (1997) 25–37.
[33] G. Peano, Sur une courbe qui remplit toute une aire plane, Mathematische Annalen 36 (1890) 157–160.
[34] G. Proietti, An optimal algorithm for decomposing a window into maximal quadtree blocks, Acta Informatica 36 (4) (1999) 257–266.
[35] M.K. Quweider, E. Salari, Peano scanning partial distance search for vector quantization, IEEE Signal Processing Letters 2 (9) (1995) 169–171.
[36] C.S. Refazzoni, A. Teschioni, A new approach to vector median filtering based on space filling curves, IEEE Transactions Image Processing 6 (7) (1997) 1025–1037.
[37] H. Sagan, Space-Filling Curves, Springer-Verlag, New York, 1994.
[38] J.L. Shih, L.H. Chen, A context-based approach for color image retrieval, International Journal of Pattern Recognition and Artificial Intelligence 16 (2) (2002) 239–255.
[39] R.J. Stevens, A.F. Lehar, F.H. Preston, Manipulation and presentation of multidimensional image data using the peano scan, IEEE Transactions Pattern Analysis and Machine Intelligence PAMI-5 (5) (1983) 520–526.
[40] Y.H. Tsai, K.L. Chung, W.Y. Chen, A strip-splitting-based optimal algorithm for decomposing a query window into maximal quadtree blocks, IEEE Transactions Knowledge and Data Engineering 16 (5) (2004) 519–523.
[41] A. Unal, Y. Saygin, O. Ulusoy, Processing count queries over event streams at multiple time granularities, Information Sciences 176 (14) (2006) 2066–2096.
[42] Y. Wang, Y. Wang, H. Kuroda, A globally adaptive pixel-decimation algorithm for block-motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 10 (6) (2000) 1006–1011.
[43] Y.F. Zhang, Space-filling curve ordered dither, Computers and Graphics 22 (4) (1998) 559–563.