Int J Theor Phys (2014) 53:2463–2484
DOI 10.1007/s10773-014-2046-4

Quantum Hilbert Image Scrambling

Nan Jiang · Luo Wang · Wen-Ya Wu

Received: 14 October 2013 / Accepted: 5 February 2014 / Published online: 30 April 2014

Abstract Analogies between quantum image processing (QIP) and classical one indicate that quantum image scrambling (QIS), as important as quantum Fourier transform (QFT), quantum wavelet transform (QWT) and etc., should be proposed to promote QIP. Image scrambling technology is commonly used to transform a meaningful image into a disordered image by permutating the pixels into new positions. Although image scrambling on classical computers has been widely studied, we know much less about QIS. In this paper, the Hilbert image scrambling algorithm, which is commonly used in classical image processing, is carried out in quantum computer by giving the scrambling quantum circuits. First, a modified recursive generation algorithm of Hilbert scanning matrix is given. Then based on the flexible representation of quantum images, the Hilbert scrambling quantum circuits, which are recursive and progressively layered, is proposed. Theoretical analysis indicates that the network complexity scales squarely with the size of the circuit's input n.

Keywords Hilbert image scrambling · Quantum circuit · Quantum computation · Quantum watermarking

1 Introduction

Quantum computers theoretically have the ability to perform computations much faster than classical computers [1–4], which causes people's interest to study quantum image processing (QIP) [3, 5, 6].

At present, QIP has three main directions: (1) the representation of quantum images; (2) the development of efficient quantum image transformations; (3) the quantum watermarking algorithms.

N. Jiang (✉) · L. Wang · W.-Y. Wu
College of Computer, Beijing University of Technology, Beijing 100124, China
e-mail: jiangnan@bjut.edu.cn

The first task of QIP is to represent quantum images. Recently, a few representing methods have been proposed such as Qubit Lattice [7, 8], Real Ket [9], FRQI [10] and NEQR [11]. The Hilbert quantum circuits proposed in this paper are based on FRQI.

It also has been proven that there are quantum processing transformations more efficient than their classical versions: quantum Fourier transform [3], quantum wavelet transform [12] and the quantum discrete cosine transform [13, 14]. In addition, there are some classical image processing operations that can not be applied on quantum images, for example convolution and correlation [15], because all operations in quantum computation must be invertible.

Watermarking is different from cryptography. It aims at guard against image abuse by embedding invisible signal (watermark) carrying information about the copyright owner into multimedia data (carrier, such as audio, video and image). The watermarked carriers are still readable. Several quantum images watermarking methods have been proposed [16–18].

In this paper we address the second problem. The basic transformation we focused on is image scrambling. We provide quantum circuits for computing the Hilbert image scrambling.

Image scrambling, which transforms a meaningful image into a meaningless or disordered image, has been widely used in the area of digital image transmission, the confidentiality storage, digital image watermarking etc... [19–23]. The popular image scrambling algorithms include Hilbert transform, Arnold transform, Fibonacci transform, Magic Square transform etc... [19, 20, 24–27]. Among them is the Hilbert scrambling algorithm, which uses the Hilbert scanning matrix [28] formed by Hilbert Curve. This is the most commonly used transformer and so we used quantum circuits to achieve the Hilbert image scrambling.

Because it is at the very start of the research, only a few papers focus the realization of image scrambling on quantum computers.

The first quantum image scrambling circuit is proposed in [29]. The circuits that achieve geometric transformations including two-point swapping, flip, coordinate swapping, orthogonal rotations and their variants on N-sized quantum images are proposed based on the basic quantum gates: NOT, CNOT and Toffoli gates. Nevertheless, it is only a “semi-scrambling” method because it focuses on geometric transformations, which just provide a realization method for scrambling.

In [16, 17], Zhang and et al. presented an image scrambling method and its quantum circuit as one step in quantum image watermarking algorithms. The key to the scrambling method is made up of two parts: two random permutations M and N, sized m and n, where m and n are the sizes of the watermark image. Assume that $M(i)$, $N(j)$ are the ith and jth numbers of M, N respectively, the pixel $(M(i), N(j))$ of the original watermark image replaces the pixel at position (i, j) of the scrambled watermark image. The quantum circuit can easily recognize this method. However, because M and N are given artificially, the scrambling uniformity and the scrambling degree cannot be guaranteed.

Jiang and et al. proposed the Arnold and Fibonacci scrambling quantum circuits based on FRQI [30]. The circuits take advantage of the plain adder and adder modulo N by modifying its input and output in order to scramble the images. The Hilbert scrambling quantum circuit is much more complex and cannot be simplified for use in the addition operation as well as other simple operations.

In this paper we study the quantum realization of Hilbert image scrambling using Hilbert Scanning Matrix.

The rest of the paper is organized as follows. A brief background on the FRQI representation, Hilbert image-scrambling and quantum gates used in this paper is presented in Section 2. A new recursive generation algorithm for Hilbert Scanning Matrix is presented in Section 3. A quantum circuit to realize the algorithm is presented in Section 4. This is followed in Section 5 by the theoretical analysis of circuits complexity. Finally, a short conclusion is given in Section 6.

2 Preliminaries

2.1 The Flexible Representation for Quantum Images (FRQI)

In order to represent images on quantum computers, the flexible representation for quantum images (FRQI) was proposed in [10, 29]. According to the FRQI, a quantum image can be written as the form shown below.

$$|I(\theta)\rangle = \frac{1}{2^n}\sum_{i=0}^{2^{2n}-1} |c_i\rangle \otimes |i\rangle$$

$$|c_i\rangle = \cos\theta_i|0\rangle + \sin\theta_i|1\rangle, \theta_i \in \left[0, \frac{2}{\pi}\right], i = 0, 1, \cdots, 2^{2n} - 1$$

where $|0\rangle, |1\rangle$ are 2 dimension computational basis quantum states, $(\theta_0, \theta_1, \cdots, \theta_{2^{2n}-1})$ is the vector of angles encoding colors, $|i\rangle$, for $i = 0, 1, \cdots, 2^{2n} - 1$ dimension computational basis quantum states, and the Kronecker product denoted by $\otimes$. There are two parts in the FRQI of an image: $|c_i\rangle$ and $|i\rangle$, which encode information about the colors and their corresponding positions in the image, respectively. The size of the quantum image is $2^n \times 2^n$.

The location information encoded in the position qubit $|i\rangle$ includes two parts: the vertical and horizontal coordinates.

$$|i\rangle = |y\rangle|x\rangle = |y_{n-1}y_{n-2}\cdots y_0\rangle|x_{n-1}x_{n-2}\cdots x_0\rangle$$

where $|y_i\rangle|x_i\rangle \in 0, 1, i = 0, 1, \cdots, n-1$. For every $i = 0, 1, \cdots, n-1$, encodes the first n-qubit $y_{n-1}y_{n-2}\cdots y_0$ along the vertical location and the second n-qubit $x_{n-1}x_{n-2}\cdots x_0$ along the horizontal axis. An example of a 2×2 FRQI image is shown in Fig. 1. Its FRQI representation is shown below. In this example, $n = 1$.

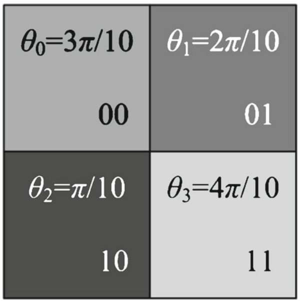
$$|I\rangle = \frac{1}{2}[(\cos\theta_0|0\rangle + \sin\theta_0|1\rangle) \otimes |00\rangle + (\cos\theta_1|0\rangle + \sin\theta_1|1\rangle) \otimes |01\rangle$$

$$+(\cos\theta_2|0\rangle + \sin\theta_2|1\rangle) \otimes |10\rangle + (\cos\theta_3|0\rangle + \sin\theta_3|1\rangle) \otimes |11\rangle]$$

2.2 Hilbert Image Scrambling

In 1890, Italian mathematician G. Peano presented a family of curves which pass through all points in a space [31]. Since this publication, many researchers have worked on this problem. Among them, Hilbert [32] found one of the simplest curves in two-dimensional (2-*D*) space, which is now called the Hilbert curve [33]. Along the Hilbert curve, an image can be scrambled very effectively.

Fig. 1 A simple image and its FRQI state



2.2.1 Hilbert Scanning Matrix and Hilbert Curve

An $2^n \times 2^n$ original image can be considered as a matrix. We call this matrix the Start matrix (or the Original matrix) S_n and use 1 to 2^{2n} to code all the pixels.

$$S_n = \begin{pmatrix} 1 & 2 & 3 & \cdots & 2^n \\ 2^n+1 & 2^n+2 & 2^n+3 & \cdots & 2^{n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 2^{2n-1}+1 & 2^{2n-1}+2 & 2^{2n-1}+3 & \cdots & 2^{2n} \end{pmatrix}$$

For example, $S_0 = (1)$, $S_1 = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $S_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$.

The Hilbert scanning matrix H_n is a permutation of S_n. For example, $H_0 = \begin{pmatrix} 1 \end{pmatrix}$, $H_1 = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$, $H_2 = \begin{pmatrix} 1 & 2 & 15 & 16 \\ 4 & 3 & 14 & 13 \\ 5 & 8 & 9 & 12 \\ 6 & 7 & 10 & 11 \end{pmatrix}$, and so on. The position (i, j) of the Hilbert scanning matrix H_n accommodates the pixel $H_n(i, j)$ of the Start matrix S_n.

Along H_n, the Hilbert curve (see Fig. 2) and scramble images can be obtained (see Fig. 3).

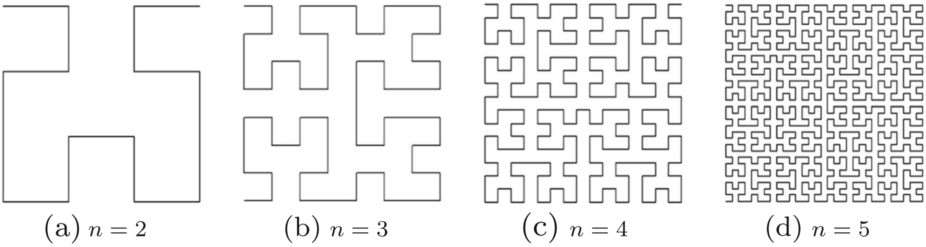

(a) $n = 2$ (b) $n = 3$ (c) $n = 4$ (d) $n = 5$

Fig. 2 Hilbert curve

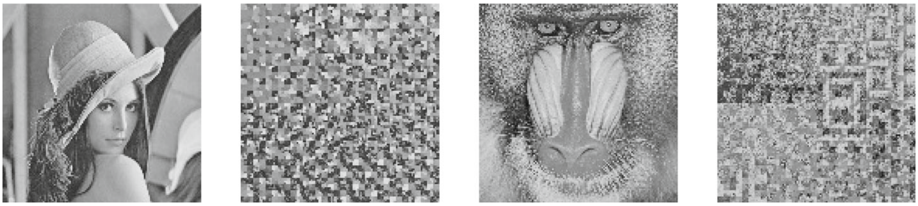

(a) Original image Lena (b) Scrambled image Lena (c) Original image Baboon (d) Scrambled image Baboon

Fig. 3 The results of Hilbert image scrambling. The image size is 128×128, i.e., $n = 7$

2.2.2 Generation of Hilbert Scanning Matrix

According to 2.2.1, the Hilbert scanning matrix is important to the Hilbert image scrambling. Reference [28] gives a recursive generation algorithm for generating the Hilbert scanning matrix. The algorithm uses several matrix operations. We first introduce them.

If A is a matrix, A^{T} is the transpose of A, A^{ud} is the up-down reversal of A, A^{lr} is the left-right reversal of A, A^{pp} is the central rotation of A. That is to say, if

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,m} \end{pmatrix},$$

then,

$$A^{\mathrm{T}} = \begin{pmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{m,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1,m} & a_{2,m} & \cdots & a_{m,m} \end{pmatrix}, A^{\mathrm{lr}} = \begin{pmatrix} a_{1,m} & \cdots & a_{1,2} & a_{1,1} \\ a_{2,m} & \cdots & a_{2,2} & a_{2,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,m} & \cdots & a_{m,2} & a_{m,1} \end{pmatrix},$$

$$A^{\mathrm{ud}} = \begin{pmatrix} a_{m,1} & a_{m,2} & \cdots & a_{m,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ a_{1,1} & a_{1,2} & \cdots & a_{1,m} \end{pmatrix}, A^{\mathrm{pp}} = \begin{pmatrix} a_{m,m} & \cdots & a_{m,2} & a_{m,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{2,m} & \cdots & a_{2,2} & a_{2,1} \\ a_{1,m} & \cdots & a_{1,2} & a_{1,1} \end{pmatrix}.$$

The recursive generation algorithm is cited with corrections from [28].

$$H_{n+1} = \begin{cases} \begin{pmatrix} H_n & 4^n E_n + H_n^{\mathrm{T}} \\ (4^{n+1}+1)E_n - H_n^{\mathrm{ud}} & (3 \times 4^n + 1)E_n - (H_n^{\mathrm{lr}})^{\mathrm{T}} \end{pmatrix}, & n \ is \ even. \\ \begin{pmatrix} H_n & (4^{n+1}+1)E_n - H_n^{\mathrm{lr}} \\ 4^n E_n + H_n^{\mathrm{T}} & (3 \times 4^n + 1)E_n - (H_n^{\mathrm{T}})^{\mathrm{lr}} \end{pmatrix}, & n \ is \ odd. \end{cases} \tag{1}$$

where, n is a positive integer, the initial matrix is $H_1 = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ and $E_n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$.

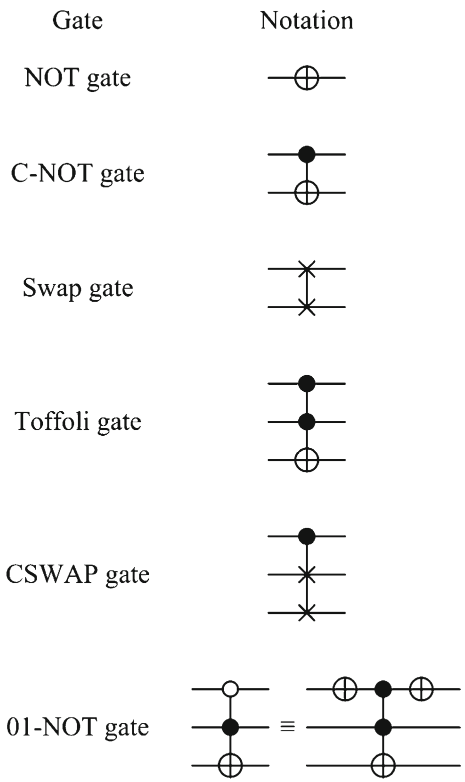


Fig. 4 Notations for quantum gates

2.3 Quantum Gates used

In this paper, we need a number of basic quantum gates [3, 4] which are all unitary.

- NOT gate. This is a single qubit gate, which inverts the content of the qubit that it operates upon.
- Controlled-NOT or C-NOT gate. It is a two-qubit gate and the content of the target qubit is inverted if and only if the control qubit is 1.
- Swap gate. It is a two-qubit gate swapping two qubits.
- Toffoli gate. This is a controlled-CNOT gate, thus, making it a three-qubit gate comprising two controls and a single target qubit. The target qubit is inverted only if both control qubits are 1.
- CSWAP gate (Fredkin gate). CSWAP gate is a 3-bit gate that performs a controlled swap, comprising a control and two target qubits. The two target qubits are swapped only if control qubit is 1.
- 01-NOT gate. This is a 01-controlled-CNOT gate, thus, making it a three-qubit gate comprising two controls and a single target qubit. The target qubit is inverted only if a control qubit is 1 and the other control qubit is 0.

The notation for these quantum gates are shown in Fig. 4.

3 The Modified Recursive Generation Algorithm for Hilbert Scanning Matrix

Equation (1), which gives the generation algorithm of Hilbert scanning Matrix, needs to be modified because we find out that it is difficult to give the quantum circuits according to it directly.

3.1 Properties of Matrix Transformations

The modifications are based on the theorem and the lemmas proposed in this section.

Theorem 1 *Assume that A, B, C, D are four $m \times m$ matrixes, then*

1. $(A+B)^{\mathrm{pp}} = A^{\mathrm{pp}} + B^{\mathrm{pp}}$;
2. $(A+B)^{\mathrm{lr}} = A^{\mathrm{lr}} + B^{\mathrm{lr}}$;
3. $(A+B)^{\mathrm{ud}} = A^{\mathrm{ud}} + B^{\mathrm{ud}}$;
4. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{\mathrm{lr}} = \begin{pmatrix} B^{\mathrm{lr}} & A^{\mathrm{lr}} \\ D^{\mathrm{lr}} & C^{\mathrm{lr}} \end{pmatrix}$;
5. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{\mathrm{ud}} = \begin{pmatrix} C^{\mathrm{ud}} & D^{\mathrm{ud}} \\ A^{\mathrm{ud}} & B^{\mathrm{ud}} \end{pmatrix}$;
6. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{\mathrm{pp}} = \begin{pmatrix} D^{\mathrm{pp}} & C^{\mathrm{pp}} \\ B^{\mathrm{pp}} & A^{\mathrm{pp}} \end{pmatrix}$;
7. $\left(A^{\mathrm{T}}\right)^{\mathrm{ud}} = \left(A^{\mathrm{lr}}\right)^{\mathrm{T}}$;
8. $A^{\mathrm{ud}} = [(A^{\mathrm{T}})^{\mathrm{lr}}]^{\mathrm{T}}$;
9. $(A^{\mathrm{ud}})^{\mathrm{pp}} = (A^{\mathrm{pp}})^{\mathrm{ud}} = A^{\mathrm{lr}},\ (A^{\mathrm{lr}})^{\mathrm{pp}} = (A^{\mathrm{pp}})^{\mathrm{lr}} = A^{\mathrm{ud}}$

where, m is a positive integer.

Proof Omitted. □

Lemma 1 *If n is an even number and $n > 0$,*

$$H_n + H_n^{\mathrm{lr}} = (4^n + 1)E_n.$$

Proof Use mathematical induction.

(1) When $n = 2$, $H_2 = \begin{pmatrix} 1 & 2 & 15 & 16 \\ 4 & 3 & 14 & 13 \\ 5 & 8 & 9 & 12 \\ 6 & 7 & 10 & 11 \end{pmatrix}$, obviously $H_2 + H_2^{\mathrm{lr}} = (4^2 + 1)E_2$.

(2) Assume that when $n = k$, $H_k + H_k^{\mathrm{lr}} = (4^k + 1)E_k$, where k is an even number. According to (1), we have

$$H_{k+1} = \begin{pmatrix} H_k & 4^k E_k + H_k^{\mathrm{T}} \\ (4^{k+1} + 1)E_k - H_k^{\mathrm{ud}} & (3 \times 4^k + 1)E_k - (H_k^{\mathrm{lr}})^{\mathrm{T}} \end{pmatrix}$$

$$H_{k+2} = \begin{pmatrix} H_{k+1} & (4^{k+2}+1)E_{k+1} - H_{k+1}^{\mathrm{lr}} \\ 4^{k+1}E_{k+1} + H_{k+1}^{\mathrm{T}} & (3\times 4^k+1)E_{k+1} - \left(H_{k+1}^{\mathrm{T}}\right)^{\mathrm{lr}} \end{pmatrix}$$

Then, introducing H_{k+1} into H_{k+2} and according to Theorem 1, we have

$$H_{k+2}=\begin{pmatrix} H_k & 4^kE_k+H_k^{\mathrm{T}} & (15\times 4^k+1)E_k-\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}} & (16\times 4^k+1)E_k-H_k^{\mathrm{lr}} \\ (4^{k+1}+1)E_k-H_k^{\mathrm{ud}} & (3\times 4^k+1)E_k-\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} & 13\times 4^kE_k-\left[\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}}\right]^{\mathrm{lr}} & 12\times 4^kE_k+\left(H_k^{\mathrm{ud}}\right)^{\mathrm{lr}} \\ 4^{k+1}E_k-H_k^{\mathrm{T}} & (8\times 4^k+1)E_k-\left(H_k^{\mathrm{ud}}\right)^{\mathrm{T}} & 8\times 4^kE_k+\left[\left(H_k^{\mathrm{ud}}\right)^{\mathrm{T}}\right]^{\mathrm{lr}} & (12\times 4^k+1)E_k-\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}} \\ (11\times 4^k+1)E_k-H_k & 9\times 4^kE_k+H_k^{\mathrm{lr}} & (7\times 4^k+1)E_k-H_k & 5\times 4^kE_k+H_k^{\mathrm{lr}} \end{pmatrix}$$

$$H_{k+2}^{\mathrm{lr}}=\begin{pmatrix} (16\times 4^k+1)E_k-H_k & (15\times 4^k+1)E_k-H_k^{\mathrm{T}} & 4^kE_k+(H_k^{\mathrm{T}})^{\mathrm{lr}} & H_k^{\mathrm{T}} \\ 12\times 4^kE_k+H_k^{\mathrm{ud}} & 13\times 4^kE_k-(H_k^{\mathrm{lr}})^{\mathrm{T}} & (3\times 4^k+1)E_k-[(H_k^{\mathrm{lr}})^{\mathrm{T}}]^{\mathrm{lr}} & (4^{k+1}+1)E_k-(H_k^{\mathrm{ud}})^{\mathrm{lr}} \\ (12\times 4^k+1)E_k-H_k^{\mathrm{T}} & 8\times 4^kE_k+(H_k^{\mathrm{ud}})^{\mathrm{T}} & (8\times 4^k+1)E_k-[(H_k^{\mathrm{ud}})^{\mathrm{T}}]^{\mathrm{lr}} & 4^{k+1}E_k-(H_k^{\mathrm{T}})^{\mathrm{lr}} \\ 5\times 4^kE_k+H_k & (7\times 4^k+1)E_k-H_k^{\mathrm{lr}} & 9\times 4^kE_k+H_k & (11\times 4^k+1)E_k-H_k^{\mathrm{lr}} \end{pmatrix}$$

Obviously $H_{k+2}+H_{k+2}^{\mathrm{lr}}=(4^{k+2}+1)E_{k+2}$.

(3) Based on the above two points, Lemma 1 holds.

□

Lemma 2 *If n is an odd number and $n>0$,*

$$H_n + H_n^{\mathrm{ud}} = (4^n+1)E_n.$$

Proof Use mathematical induction.

(1) When $n=1$, $H_1=\begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$, obviously $H_1+H_1^{\mathrm{ud}}=(4^1+1)E_1$.

(2) Assume that when $n=k$, $H_k+H_k^{\mathrm{ud}}=(4^k+1)E_k$, where k is an odd number. According to (1),

$$H_{k+1} = \begin{pmatrix} H_k & (4^{k+1}+1)E_k + H_k^{\mathrm{lr}} \\ 4^kE_k - H_k^{\mathrm{T}} & (3\times 4^k+1)E_k - \left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}} \end{pmatrix}$$

$$H_{k+2} = \begin{pmatrix} H_{k+1} & 4^{k+1}E_{k+1} + H_{k+1}^{\mathrm{T}} \\ (4^{k+2}+1)E_{k+1} - H_{k+1}^{\mathrm{ud}} & (3\times 4^{k+1}+1)E_{k+1} - \left(H_{k+1}^{\mathrm{lr}}\right)^{\mathrm{T}} \end{pmatrix}$$

Then, introducing H_{k+1} into H_{k+2} and according to Theorem 1,

$$H_{k+2}=\begin{pmatrix} H_k & (4^{k+1}+1)E_k+H_k^{\mathrm{lr}} & 4^{k+1}E_k+H_k^{\mathrm{T}} & 5\times 4^kE_k+H_k \\ 4^kE_k+H_k^{\mathrm{T}} & (3\times 4^k+1)E_k-\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}} & (8\times 4^k+1)E_k-\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} & (7\times 4^k+1)E_k-\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} \\ (15\times 4^k+1)E_k-\left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} & 13\times 4^kE_k+\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{ud}} & 8\times 4^kE_k+H_k^{\mathrm{T}} & 9\times 4^kE_k+H_k \\ (16\times 4^k+1)E_k-H_k^{\mathrm{ud}} & 12\times 4^kE_k+\left(H_k^{\mathrm{lr}}\right)^{\mathrm{ud}} & (12\times 4^k+1)E_k-\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} & (11\times 4^k+1)E_k-\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} \end{pmatrix}$$

$$H_{k+2}^{\mathrm{ud}}=\begin{pmatrix} (16\times 4^k+1)E_k-H_k & 12\times 4^kE_k+H_k^{\mathrm{lr}} & (12\times 4^k+1)E_k-\left[\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}}\right]^{\mathrm{ud}} & (11\times 4^k+1)E_k-\left(\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}}\right)^{\mathrm{ud}} \\ (15\times 4^k+1)E_k-H_k^{\mathrm{T}} & 13\times 4^kE_k+\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}} & 8\times 4^kE_k+\left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} & 9\times 4^kE_k+H_k^{\mathrm{ud}} \\ 4^kE_k+\left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} & (3\times 4^k+1)E_k-\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{ud}} & 8\times 4^kE_k+H_k^{\mathrm{T}} & (7\times 4^k+1)E_k-\left(\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}}\right)^{\mathrm{ud}} \\ H_k^{\mathrm{ud}} & (4^{k+1}+1)E_k+\left(H_k^{\mathrm{lr}}\right)^{\mathrm{ud}} & 4^{k+1}E_k+\left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} & 5\times 4^kE_k+H_k^{\mathrm{ud}} \end{pmatrix}$$

Then,

$$H_{k+2} + H_{k+2}^{\mathrm{ud}} = (4^{k+2}+1)E_{k+2} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & H_k^{\mathrm{T}} - \left[\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}}\right]^{\mathrm{ud}} & H_k - \left(\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}}\right)^{\mathrm{ud}} \\ \mathbf{0} & \mathbf{0} & \left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} - \left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} & H_k^{\mathrm{ud}} - \left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} \\ \mathbf{0} & \mathbf{0} & H_k^{\mathrm{T}} - \left[\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}}\right]^{\mathrm{ud}} & H_k - \left(\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}}\right)^{\mathrm{ud}} \\ \mathbf{0} & \mathbf{0} & \left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} - \left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} & H_k^{\mathrm{ud}} - \left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} \end{pmatrix}$$

According to Theorem 1,

$$\left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} = \left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} \Leftrightarrow \left(H_k^{\mathrm{T}}\right)^{\mathrm{ud}} - \left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}} = \mathbf{0} \Leftrightarrow H_k^{\mathrm{T}} - \left[\left(H_k^{\mathrm{lr}}\right)^{\mathrm{T}}\right]^{\mathrm{ud}} = \mathbf{0}$$

and

$$H_k^{\mathrm{ud}} = \left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} \Leftrightarrow H_k^{\mathrm{ud}} - \left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}} = \mathbf{0} \Leftrightarrow H_k - \left\{\left[\left(H_k^{\mathrm{T}}\right)^{\mathrm{lr}}\right]^{\mathrm{T}}\right\}^{\mathrm{ud}} = \mathbf{0}$$

Hence, $H_{k+2} + H_{k+2}^{\mathrm{ud}} = (4^{k+2}+1)E_{k+2}$.

(3) Based on the above two points, Lemma 2 holds.

□

3.2 The Modified Recursive Algorithm for Generating Hilbert Scanning Matrix

In order to realize the Hilbert image scrambling in quantum computer, the recursive generation algorithm was modified for the Hilbert scanning matrix, i.e., (1).

Theorem 2

$$H_{n+1} = \begin{cases} \begin{pmatrix} H_n & (H_n + 4^n E_n)^{\mathrm{T}} \\ (H_n + 3 \times 4^n E_n)^{\mathrm{pp}} & (H_n + 2 \times 4^n E_n)^{\mathrm{T}} \end{pmatrix}, & n \; is \; even. \\ \begin{pmatrix} H_n & (H_n + 3 \times 4^n E_n)^{\mathrm{pp}} \\ (H_n + 4^n E_n)^{\mathrm{T}} & (H_n + 2 \times 4^n E_n)^{\mathrm{T}} \end{pmatrix}, & n \; is \; odd. \end{cases} \tag{2}$$

where, n is a positive integer, the initial matrix is $H_1 = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ *and* $E_n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$.

Proof What needs to do is to prove that (1) and (2) are equivalent. Note that, (1) or (2) and no matter n is even or odd, H_{n+1} is divided into 4 parts; the 4 parts in (2) are equal to the 4 parts in (1) needs to be proven.

(1) When n is an even number, the following 4 issues needs to be proven:

(i) $H_n = H_n$
(ii) $4^n E_n + H_n^{\mathrm{T}} = (H_n + 4^n E_n)^{\mathrm{T}}$
(iii) $(4^{n+1} + 1)E_n - H_n^{\mathrm{ud}} = (H_n + 3 \times 4^n E_n)^{\mathrm{pp}}$
(iv) $(3 \times 4^n + 1)E_n - \left(H_n^{\mathrm{lr}}\right)^{\mathrm{T}} = (H_n + 2 \times 4^n E_n)^{\mathrm{T}}$

For (i) and (ii)

They are obviously correct.

For (iii)

According to Lemma 1 and Theorem 1,

$$\begin{aligned}
&H_n + H_n^{\mathrm{lr}} = (4^n + 1)E_n \\
\Rightarrow &\left(H_n + H_n^{\mathrm{lr}}\right)^{\mathrm{pp}} = H_n^{\mathrm{pp}} + H_n^{\mathrm{ud}} = \left[(4^n + 1)E_n\right]^{\mathrm{pp}} = (4^n + 1)E_n \\
\Rightarrow &(4^n + 1)E_n - H_n^{\mathrm{ud}} = H_n^{\mathrm{pp}} \\
\Rightarrow &(4^n + 1)E_n - H_n^{\mathrm{ud}} + 3 \times 4^n E_n = H_n^{\mathrm{pp}} + 3 \times 4^n E_n \\
\Rightarrow &(4^{n+1} + 1)E_n - H_n^{\mathrm{ud}} = \left(H_n + 3 \times 4^n E_n\right)^{\mathrm{pp}}
\end{aligned}$$

For (iv)

According to Lemma 1 and Theorem 1,

$$\begin{aligned}
&H_n + H_n^{\mathrm{lr}} = (4^n + 1)E_n \\
\Rightarrow &\left(H_n + H_n^{\mathrm{lr}}\right)^{\mathrm{T}} = H_n^{\mathrm{T}} + \left(H_n^{\mathrm{lr}}\right)^{\mathrm{T}} = \left[(4^n + 1)E_n\right]^{\mathrm{T}} = (4^n + 1)E_n \\
\Rightarrow &(4^n + 1)E_n - \left(H_n^{\mathrm{lr}}\right)^{\mathrm{T}} = H_n^{\mathrm{T}} \\
\Rightarrow &(4^n + 1)E_n - \left(H_n^{\mathrm{lr}}\right)^{\mathrm{T}} + 2 \times 4^n E_n = H_n^{\mathrm{T}} + 2 \times 4^n E_n \\
\Rightarrow &(3 \times 4^n + 1)E_n - \left(H_n^{\mathrm{lr}}\right)^{\mathrm{T}} = \left(H_n + 2 \times 4^n E_n\right)^{\mathrm{T}}
\end{aligned}$$

Hence, when n is an even number,

$$H_{n+1} = \begin{pmatrix} H_n & (H_n + 4^n E_n)^{\mathrm{T}} \\ (H_n + 3 \times 4^n E_n)^{\mathrm{pp}} & (H_n + 2 \times 4^n E_n)^{\mathrm{T}} \end{pmatrix}$$

(2) When n is an odd number, the following 4 issues needs to be proven:

(i) $H_n = H_n$
(ii) $(4^{n+1}+1)E_n - H_n^{\mathrm{lr}} = (H_n + 3\times 4^n E_n)^{\mathrm{pp}}$
(iii) $4^n E_n + H_n^{\mathrm{T}} = (H_n + 4^n E_n)^{\mathrm{T}}$
(iv) $(3\times 4^n + 1)E_n - \left(H_n^{\mathrm{T}}\right)^{\mathrm{lr}} = (H_n + 2\times 4^n E_n)^{\mathrm{T}}$

For (i) and (iii)

They are obviously correct.

For (ii)

According to Lemma 2 and Theorem 1,

$$\begin{aligned}
&H_n + H_n^{\mathrm{ud}} = (4^n+1)E_n \\
&\Rightarrow \left(H_n + H_n^{\mathrm{ud}}\right)^{\mathrm{pp}} = H_n^{\mathrm{pp}} + H_n^{\mathrm{lr}} = \left[(4^n+1)E_n\right]^{\mathrm{pp}} = (4^n+1)E_n \\
&\Rightarrow (4^n+1)E_n - H_n^{\mathrm{lr}} = H_n^{\mathrm{pp}} \\
&\Rightarrow (4^n+1)E_n - H_n^{\mathrm{lr}} + 3\times 4^n E_n = H_n^{\mathrm{pp}} + 3\times 4^n E_n \\
&\Rightarrow (4^{n+1}+1)E_n - H_n^{\mathrm{lr}} = \left(H_n + 3\times 4^n E_n\right)^{\mathrm{pp}}
\end{aligned}$$

For (iv)

According to Lemma 2 and Theorem 1,

$$\begin{aligned}
&H_n + H_n^{\mathrm{ud}} = (4^n+1)E_n \\
&\Rightarrow \left(H_n + H_n^{\mathrm{ud}}\right)^{\mathrm{T}} = H_n^{\mathrm{T}} + \left(H_n^{\mathrm{T}}\right)^{\mathrm{lr}} = \left[(4^n+1)E_n\right]^{\mathrm{T}} = (4^n+1)E_n \\
&\Rightarrow (4^n+1)E_n - \left(H_n^{\mathrm{T}}\right)^{\mathrm{lr}} = H_n^{\mathrm{T}} \\
&\Rightarrow (4^n+1)E_n - \left(H_n^{\mathrm{T}}\right)^{\mathrm{lr}} + 2\times 4^n E_n = H_n^{\mathrm{T}} + 2\times 4^n E_n \\
&\Rightarrow (3\times 4^n+1)E_n - \left(H_n^{\mathrm{T}}\right)^{\mathrm{lr}} = \left(H_n + 2\times 4^n E_n\right)^{\mathrm{T}}
\end{aligned}$$

Hence, when n is an odd number,

$$H_{n+1} = \begin{pmatrix} H_n & (H_n + 3\times 4^n E_n)^{\mathrm{pp}} \\ (H_n + 4^n E_n)^{\mathrm{T}} & (H_n + 2\times 4^n E_n)^{\mathrm{T}} \end{pmatrix}$$

According to the above two points, Theorem 2 is proved. □

In the following, the quantum Hilbert image scrambling is based on Theorem 2 (or (2)).

4 Quantum Circuits for Hilbert Image Scrambling

In this section, a quantum circuit will be given in order to scramble a quantum image. Its input is the $2^n \times 2^n$ original quantum image and its output is the Hilbert scrambled quantum image. The images are represented using FRQI.

4.1 The Circuit Flow

According to (2), as n increases, the recursive generation algorithm can be divided into three basic operations: initialization, odd and even. Being aware of the fact that even and odd are carried out alternately. The process can be described in Fig. 5.

Accordingly, the quantum circuit can be composed of three basic circuits. We also call them Initialization, Even and Odd.

However, the equation and the circuits have two key differences.

1. The result of (2) is the Hilbert scanning matrix H_n. It needs an extra step – put the pixel $(\lfloor(H_n(i,j)-1)/2^n\rfloor, (H_n(i,j)-1)\mathrm{mod}2^n)$ of the original image into the position (i,j) in the scrambled image – to scramble images. But in circuits, the output is the scrambled image directly.
2. In (2), the size of the Hilbert scanning matrix H_n increases gradually: from 2 × 2 to 4 × 4, to 8 × 8, to 16 × 16 and so on. But in circuits, the sizes of the input image and the output image are all $2^n \times 2^n$. We can not and should not change the size of the image in the circuits.

The method to solve the first problem is to operate the image directly instead of the Hilbert scanning matrix. The countermeasure to the second problem is to partition the image into blocks and operate each block according to (2). Note that the partition operation needs to be operated many times:

The 1st time it partition the image into $2^{n-1} \times 2^{n-1}$ blocks sized 2 × 2;

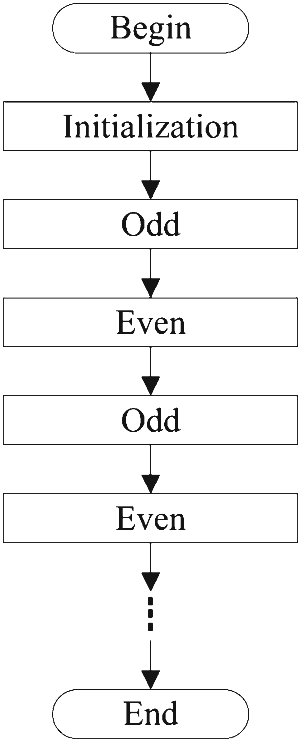


Fig. 5 The recursive generation algorithm flowchart

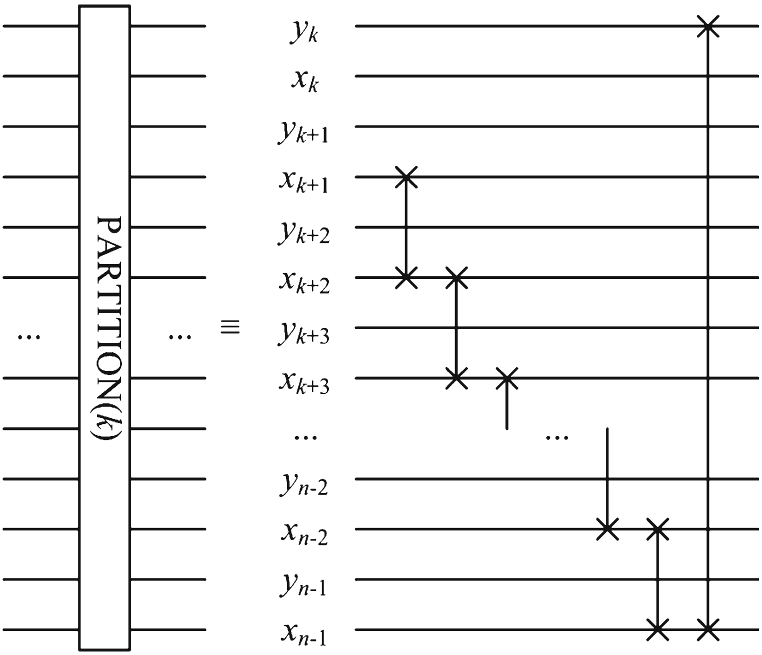


Fig. 6 PARTITION(k)

The 2nd time it partition the image into $2^{n-2} \times 2^{n-2}$ blocks sized 4×4;
The 3rd time it partition the image into $2^{n-3} \times 2^{n-3}$ blocks sized 8×8;
$\cdots$;
The nth time it partition the image into 1 block sized $2^n \times 2^n$.

They correspond to the gradually increasing size of the Hilbert scanning matrix H_n.

Hence, the three basic circuits Initialization, Even and Odd are also compositive. We call the three parts composing the three basic circuits as the three circuit modules.

4.2 The Three Circuit Modules

In this section, k is an integer and $0 \le k \le n-1$.

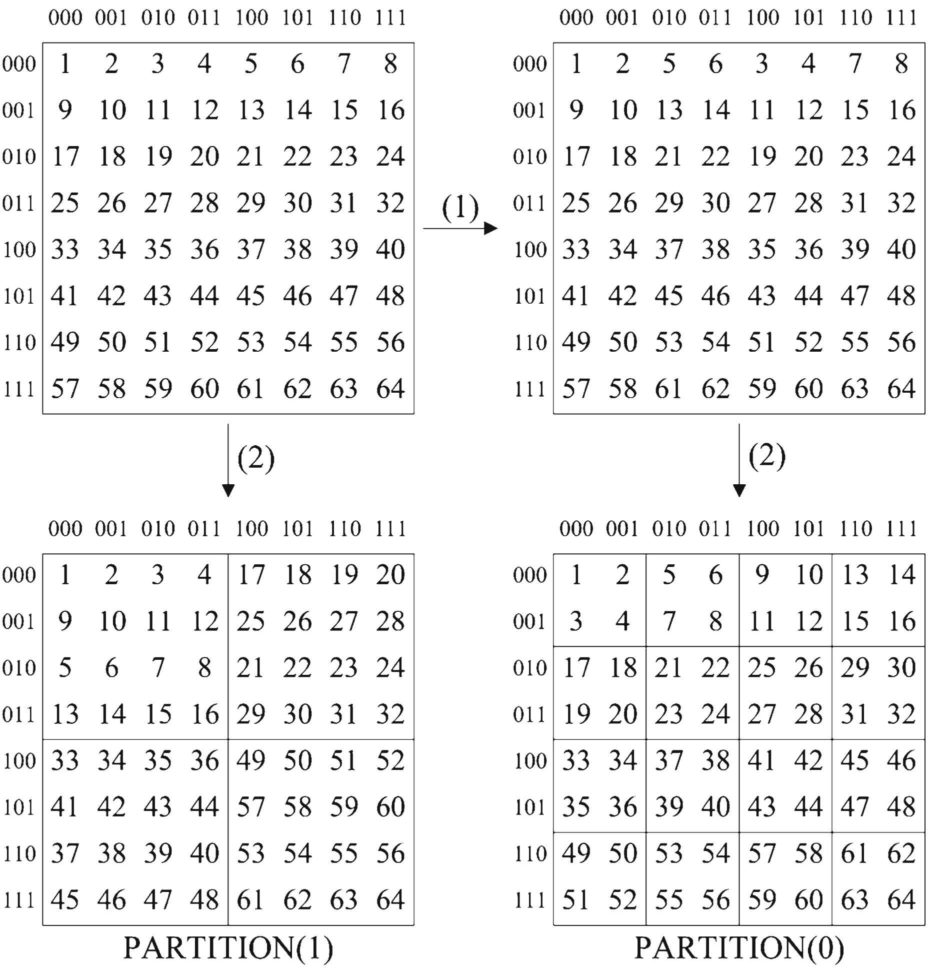
4.2.1 Module PARTITION(k)

The PARTITION(k) module can divide the $2^n \times 2^n$ input image into $2^{n-k-1} \times 2^{n-k-1}$ blocks sized $2^{k+1} \times 2^{k+1}$.

The PARTITION(k) quantum circuit is

(1) Swap x_{k+1} and x_{k+2}, x_{k+2} and x_{k+3}, $\cdots$, x_{n-2} and x_{n-1}.
(2) Swap x_{n-1} and y_k.

As shown in Fig. 6. Because we need not to change the pixels' color, the inputs of the circuit only include the location information $|i\rangle = |y\rangle|x\rangle$ as shown in Section 2.1 (the same as following circuits).



	000	001	010	011	100	101	110	111
000	1	2	3	4	5	6	7	8
001	9	10	11	12	13	14	15	16
010	17	18	19	20	21	22	23	24
011	25	26	27	28	29	30	31	32
100	33	34	35	36	37	38	39	40
101	41	42	43	44	45	46	47	48
110	49	50	51	52	53	54	55	56
111	57	58	59	60	61	62	63	64

	000	001	010	011	100	101	110	111
000	1	2	5	6	3	4	7	8
001	9	10	13	14	11	12	15	16
010	17	18	21	22	19	20	23	24
011	25	26	29	30	27	28	31	32
100	33	34	37	38	35	36	39	40
101	41	42	45	46	43	44	47	48
110	49	50	53	54	51	52	55	56
111	57	58	61	62	59	60	63	64

	000	001	010	011	100	101	110	111
000	1	2	3	4	17	18	19	20
001	9	10	11	12	25	26	27	28
010	5	6	7	8	21	22	23	24
011	13	14	15	16	29	30	31	32
100	33	34	35	36	49	50	51	52
101	41	42	43	44	57	58	59	60
110	37	38	39	40	53	54	55	56
111	45	46	47	48	61	62	63	64

PARTITION(1)

	000	001	010	011	100	101	110	111
000	1	2	5	6	9	10	13	14
001	3	4	7	8	11	12	15	16
010	17	18	21	22	25	26	29	30
011	19	20	23	24	27	28	31	32
100	33	34	37	38	41	42	45	46
101	35	36	39	40	43	44	47	48
110	49	50	53	54	57	58	61	62
111	51	52	55	56	59	60	63	64

PARTITION(0)

Fig. 7 Two examples for PARTITION(k)

For example, assume $n = 3$, the input of PARTITION(k) is S_3, the function of quantum circuit PARTITION(0) and PARTITION(1) is shown in Fig. 7.

4.2.2 Module O(k)

Assume $A,\ B,\ C,\ D$ are matrixes sized $2^{k-1} \times 2^{k-1}$, the function of O(k) is that transform $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ into $\begin{pmatrix} A & D^{\mathrm{pp}} \\ B^{\mathrm{T}} & C^{\mathrm{T}} \end{pmatrix}$ which is similar to the form in (2) when n is odd number.

The O(k) (k is odd) quantum circuit is

(1) Swap x_k and y_k. Add a C-NOT Gate, and set x_k as as control qubit, y_k as target qubit.
(2) Add CSWAP Gates whose control qubit is y_k to swap x_0 and y_0, x_1 and y_1, $\cdots$, and x_{k-1} and y_{k-1}.
(3) Add 01-NOT Gates whose 0-control qubit is y_k and 1-control qubit is x_k to reverse $x_{k-1}, x_{k-2}, \cdots, x_0, y_{k-1}, y_{k-2}, \cdots, y_0$.

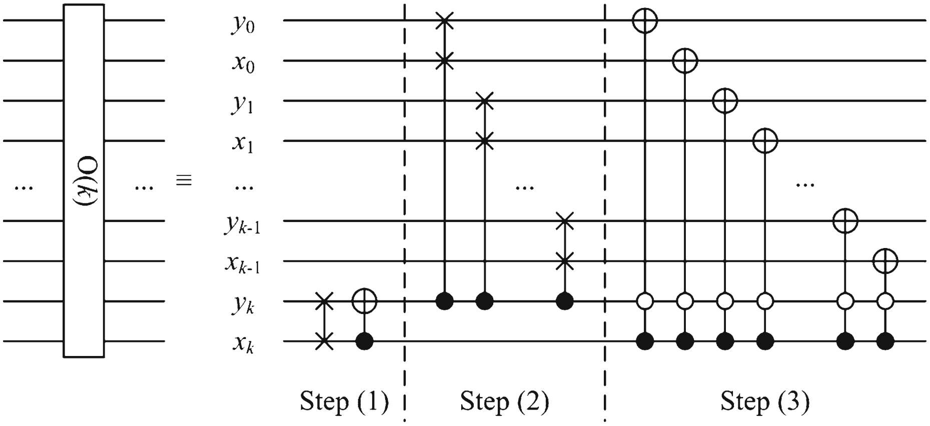


Fig. 8 O(k)

As shown in Fig. 8.

The result of every step is shown in Fig. 9.

4.2.3 *Module E(k)*

The function of E(k) is that transform $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ into $\begin{pmatrix} A & B^{\mathrm{T}} \\ D^{\mathrm{pp}} & C^{\mathrm{T}} \end{pmatrix}$ which is similar to the form in (2) when n is even number.

The E(k) (k is even) quantum circuit is

(1) Add a C-NOT Gate, and set y_k as as control qubit, x_k as target qubit.
(2) Add CSWAP Gates whose control qubit is x_k to swap x_0 and y_0, x_1 and y_1, $\cdots$, and x_{k-1} and y_{k-1}.
(3) Add 01-NOT Gates whose 0-control qubit is x_k and 1-control qubit is y_k to reverse $x_{k-1}, x_{k-2}, \cdots, x_0, y_{k-1}, y_{k-2}, \cdots, y_0$.

As shown in Fig. 10.

The result of every step is shown in Fig. 11.

The unitarity of the three circuit modules roots in the unitarity of basic quantum gates showed in Section 2.3.

4.3 The Three Basic Circuits

The three basic circuits Initialization, Odd and Even are composed of the three circuit modules.

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \xrightarrow{Step(1)} \begin{pmatrix} A & D \\ B & C \end{pmatrix} \xrightarrow{Step(2)} \begin{pmatrix} A & D \\ B^{\mathrm{T}} & C^{\mathrm{T}} \end{pmatrix} \xrightarrow{Step(3)} \begin{pmatrix} A & D^{\mathrm{pp}} \\ B^{\mathrm{T}} & C^{\mathrm{T}} \end{pmatrix}$$

Fig. 9 The function of O(k)

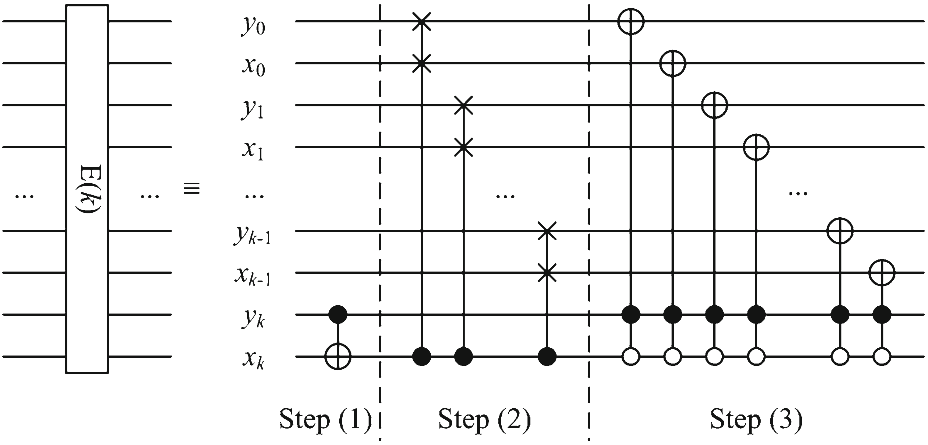


Fig. 10 E(k)

4.3.1 *Initialization*

The Initialization quantum circuit shown in Fig. 12 is used to generate the initial matrix $H_1 = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ as mentioned in Theorem 2. PARTITION(0) divides the $2^n \times 2^n$ input image into $2^{n-1} \times 2^{n-1}$ blocks sized 2×2. Then, for each block $\left(\text{denoted as} \begin{pmatrix} a & b \\ c & d \end{pmatrix}\right)$, when $y_0 = 1$, x_0 is reversed and c and d swaped. The initialized block $\begin{pmatrix} a & b \\ d & c \end{pmatrix}$ is corresponding to the initial matrix $H_1 = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ in Theorem 2.

4.3.2 *Odd(k)*

The Odd(k) (k is an odd number and $1 \leq k \leq n - 1$) quantum circuit is shown in Fig. 13. Note that when $k = n$, the PARTITION module disappears and Odd(k) becomes O(k).

The PARTITION(k) module divide the input image into blocks and the O(k) module change every block into the form of $\begin{pmatrix} A & D^{\mathrm{pp}} \\ B^{\mathrm{T}} & C^{\mathrm{T}} \end{pmatrix}$ respectively.

4.3.3 *Even(k)*

The Even(k) (k is an even number and $2 \leq k \leq n - 1$) quantum circuit is shown in Fig. 14. Note that when $k = n$, the PARTITION module disappears and Even(k) becomes E(k).

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \xrightarrow{Step(1)} \begin{pmatrix} A & B \\ D & C \end{pmatrix} \xrightarrow{Step(2)} \begin{pmatrix} A & B^{\mathrm{T}} \\ D & C^{\mathrm{T}} \end{pmatrix} \xrightarrow{Step(3)} \begin{pmatrix} A & B^{\mathrm{T}} \\ D^{\mathrm{pp}} & C^{\mathrm{T}} \end{pmatrix}$$

Fig. 11 The function of E(k)

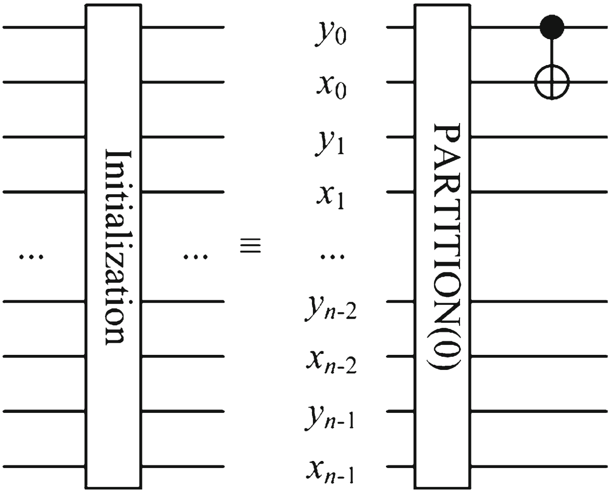


Fig. 12 The Initialization quantum circuit

The PARTITION(k) module divide the input image into blocks and the E(k) module change every block into the form of $\begin{pmatrix} A & B^{\mathrm{T}} \\ D^{\mathrm{pp}} & C^{\mathrm{T}} \end{pmatrix}$ respectively.

4.4 The Integrated Hilbert Image Scrambling Quantum Circuit

According to (2) and Fig. 5, the integrated Hilbert image scrambling quantum circuit is shown in Fig. 15. The meaning of the final basic circuit Even$(n-1)$/Odd$(n-1)$ is that if $n-1$ is an even number, the final basic circuit is Even$(n-1)$; otherwise, it is Odd$(n-1)$.

Fig. 13 The Odd(k) quantum circuit

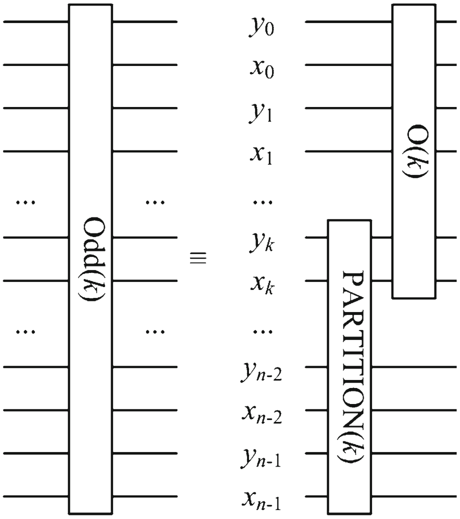
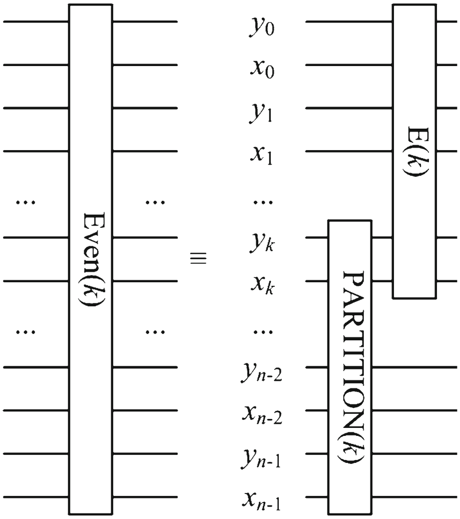


Fig. 14 The Even(k) quantum circuit



4.5 The Inverse Circuit

Because the quantum gates are unitary and no information is lost, the integrated Hilbert image scrambling quantum circuit is unitary. If we reverse the action of it (i.e., if we apply each gate of the circuit in the reversed order) with the scrambled image as input, the output will produce the rebuild image.

4.6 A Simple Example

Assume that $n = 3$, the Hilbert image-scrambling circuit is shown in Fig. 16.

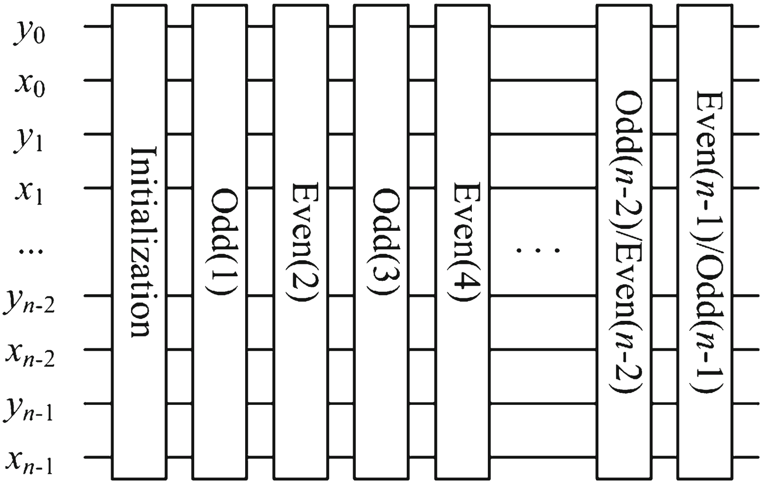


Fig. 15 The integrated Hilbert image scrambling quantum circuit

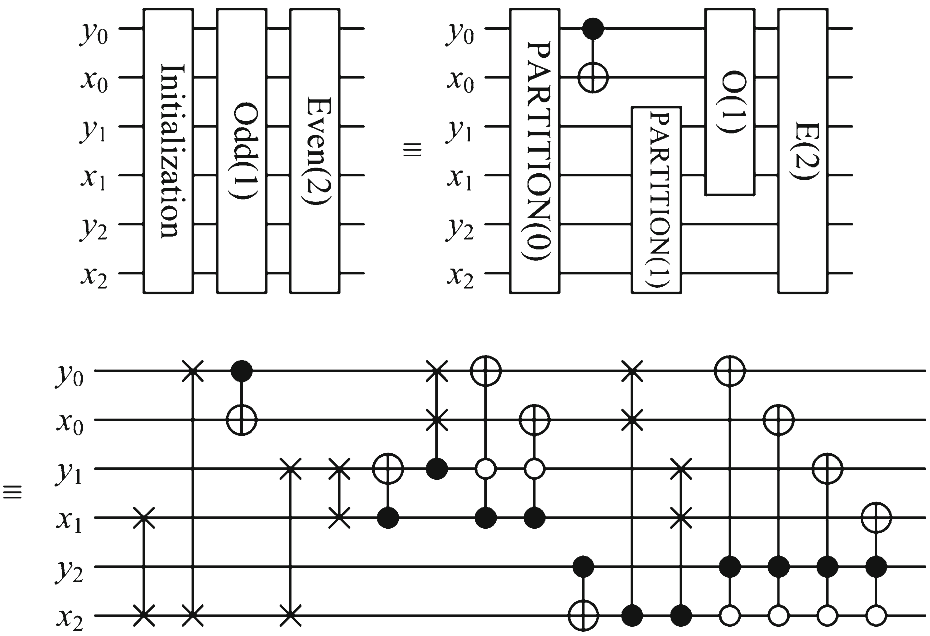


Fig. 16 The quantum circuits for the simple example

The image has $2^3 \times 2^3 = 64$ pixels. If the pixel values of the original image are "$1, 2, 3, \cdots, 63, 64$", the image processing procedure can be shown in Fig. 17. PARTITION(0) blocks the image into 2×2 subimages which is $\begin{pmatrix} i & i+1 \\ i+2 & i+3 \end{pmatrix}$. The C-NOT gate swaps the last two pixel of every subimage. PARTITION(1) blocks the image into 4×4 subimages. O(1) and E(2) complete the scrambling.

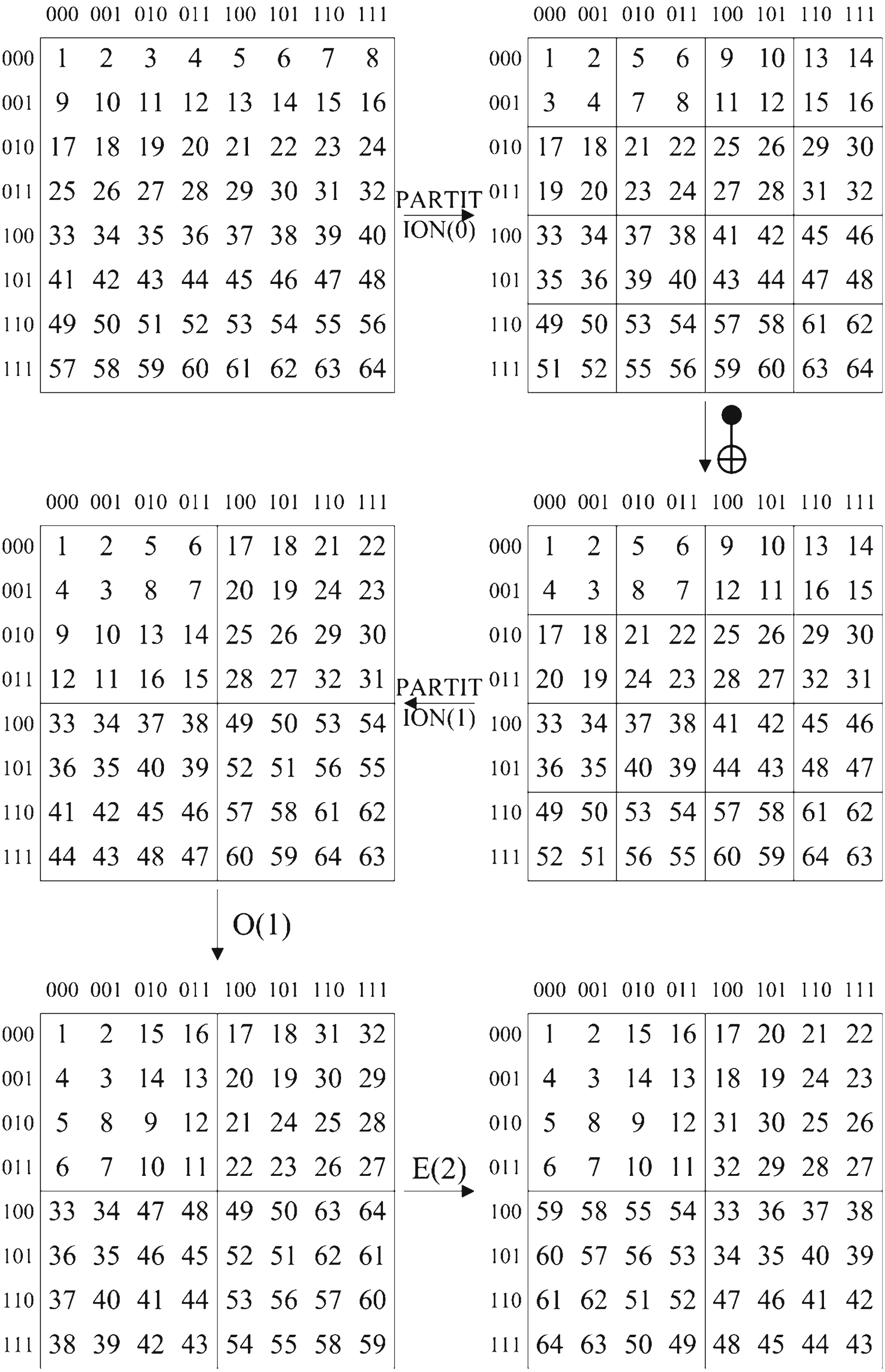
If we input the scrambled image from the right of the circuit shown in Fig. 16, the original image will be got from the left.

5 Circuit Complexity

The network complexity depends very much on what is considered to be an elementary gate. In this section, we choose the Control-NOT to be our basic unit, then the Swap gate can be simulated by 3 Control-NOT gates, the Toffoli gate can be simulated by 6 Control-NOT gates, the CSWAP gate can be simulated by 18 Control-NOT gates, and the 01-NOT gate can be simulated by 8 Control-NOT gates [1].

According to Figs. 6, 8, and 10, the numbers of elementary gates in PARTITION(k) is $3 \times (n - k - 1) = 3n - 3k - 3$, O($k$) is $3 + 1 + 18 \times k + 8 \times 2k = 34k + 4$, and E($k$) is $1 + 18 \times k + 8 \times 2k = 34k + 1$ respectively.

Consequently, according to Figs. 12–14, the complexity of Initialization is $(3n - 3) + 1 = 3n - 2$, Odd(k) is $(3n - 3k - 3) + (34k + 4) = 3n + 31k + 1$, and Even($k$) is $(3n - 3k - 3) + (34k + 1) = 3n + 31k - 2$.



	000	001	010	011	100	101	110	111
000	1	2	3	4	5	6	7	8
001	9	10	11	12	13	14	15	16
010	17	18	19	20	21	22	23	24
011	25	26	27	28	29	30	31	32
100	33	34	35	36	37	38	39	40
101	41	42	43	44	45	46	47	48
110	49	50	51	52	53	54	55	56
111	57	58	59	60	61	62	63	64

	000	001	010	011	100	101	110	111
000	1	2	5	6	9	10	13	14
001	3	4	7	8	11	12	15	16
010	17	18	21	22	25	26	29	30
011	19	20	23	24	27	28	31	32
100	33	34	37	38	41	42	45	46
101	35	36	39	40	43	44	47	48
110	49	50	53	54	57	58	61	62
111	51	52	55	56	59	60	63	64

	000	001	010	011	100	101	110	111
000	1	2	5	6	17	18	21	22
001	4	3	8	7	20	19	24	23
010	9	10	13	14	25	26	29	30
011	12	11	16	15	28	27	32	31
100	33	34	37	38	49	50	53	54
101	36	35	40	39	52	51	56	55
110	41	42	45	46	57	58	61	62
111	44	43	48	47	60	59	64	63

	000	001	010	011	100	101	110	111
000	1	2	5	6	9	10	13	14
001	4	3	8	7	12	11	16	15
010	17	18	21	22	25	26	29	30
011	20	19	24	23	28	27	32	31
100	33	34	37	38	41	42	45	46
101	36	35	40	39	44	43	48	47
110	49	50	53	54	57	58	61	62
111	52	51	56	55	60	59	64	63

	000	001	010	011	100	101	110	111
000	1	2	15	16	17	18	31	32
001	4	3	14	13	20	19	30	29
010	5	8	9	12	21	24	25	28
011	6	7	10	11	22	23	26	27
100	33	34	47	48	49	50	63	64
101	36	35	46	45	52	51	62	61
110	37	40	41	44	53	56	57	60
111	38	39	42	43	54	55	58	59

	000	001	010	011	100	101	110	111
000	1	2	15	16	17	20	21	22
001	4	3	14	13	18	19	24	23
010	5	8	9	12	31	30	25	26
011	6	7	10	11	32	29	28	27
100	59	58	55	54	33	36	37	38
101	60	57	56	53	34	35	40	39
110	61	62	51	52	47	46	41	42
111	64	63	50	49	48	45	44	43

Fig. 17 The image processing procedure for the simple example

Therefore, according to Fig. 15, the whole circuit complexity is $(3n-2)+\sum_{k\text{ is odd}}(3n+31k+1)+\sum_{k\text{ is even}}(3n+31k-2)\approx 18n^2+18n-2$. It scales squarely with the size of the circuit's input n.

6 Conclusion

A Hilbert image scrambling strategy for quantum images is proposed in this paper. The scheme was based on the modified recursive generation algorithm for Hilbert scanning matrix in order to transform images into Hilbert scrambled version by quantum circuits. The circuits can be divided into three basic circuits: Initialization, Odd and Even. The realization of them in quantum computers can promote QIP. The complexity of the networks is square.

Acknowledgments This work is supported by the Beijing Municipal Education Commission Science and Technology Development Plan under Grants No. KM201310005021, KZ201210005007, the Fundamental Research Funds for the Central Universities under Grants No. 2012JBM041, and the Graduate Technology Fund of BJUT under Grants No. YKJ-2013-10282.

References

1. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Phys. Rev. A **54**(1), 147–153 (1996)
2. Feynman, R.: Quantum mechanical computers. Found. Phys. **16**(6), 507–531 (1986)
3. Nielson, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
4. Williams, C.: Explorations in quantum computing. In: Texts in Computer Science (2011)
5. Caraiman, S., Manta, V.: Image processing using quantum computing. In: International Conference on 16th System Theory, Control and Computing (ICSTCC), pp. 1–6 (2012)
6. Beach, G., Lomont, C., Cohen, C.: Quantum image processing (QuIP). In: 32nd Applied Imagery Pattern Recognition Workshop, pp. 39–44 (2003)
7. Venegas-Andraca, S.E., Ball, J.L.: Processing images in entangled quantum system. J. Quantum. Inf. Process. **9**(1), 1–11 (2010)
8. Venegas-Andraca, S.E., Bose, S.: Storing, processing and retrieving an image using quantum mechanics. In: Proceedings of the SPIE Conference on Quantum Information and Computation, pp. 137–147 (2003)
9. Latorre, J.I.: Image Compression and Entanglement. (2005). arXiv:quantph/0510031
10. Le, P.Q., Dong, F., Hirota, K.: A flexible representation of quantum images for polynomial preparation, image compression and processing operations. Quantum Inf. Process. **10**(1), 63–84 (2011)
11. Zhang, Y., Lu, K., Gao, Y.-H., Wang, M.: NEQR: a novel enhanced quantum representation of digital images. Quantum Inf. Process. **12**(12), 2833–2860 (2013)
12. Fijany, A., Williams, C.: Quantum Wavelet Transform: Fast Algorithm and Complete Circuits (1998). arXiv:quant-ph/9809004
13. Klappenecker, A., Roetteler, M.: Discrete cosine transforms on quantum computers. In: IEEER8-EURASIP Symposium on Image and Signal Processing and Analysis (ISPA01), pp. 464–468. Pula (2001)
14. Tseng, C., Hwang, T.: Quantum circuit design of 8 × 8 discrete cosine transforms using its fast computation flow graph. In: IEEE International Symposium on Circuits and Systems, pp. 828–831 (2005)
15. Lomont, C.: Quantum Convolution and Quantum Correlation Algorithms are Physically Impossible (2003). arXiv:quant-ph/0309070
16. Zhang, W.-W., Gao, F., Liu, B., et al.: A watermark strategy for quantum images based on quantum Fourier transform. Quantum Inf. Process. **12**(4), 793–803 (2013)
17. Weiwei, Z., Fei, G., Bing, L., et al.: A quantum watermark protocol. Int. J. Theor. Phys. **52**, 504–513 (2013)
18. Iliyasu, A.M., Le, P.Q., Dong, F., Hirota, K.: Watermarking and authentication of quantum images based on restricted geometric transformations. Inf. Sci. **186**, 126–149 (2012)

19. Hu, M.-Y., Tian, X.-L., Xia, S.-W., et al.: Image scrambling based on 3-D Hilbert curve. In: 3rd International Congress on Image and Signal Processing, vol. 1, pp. 147–149 (2010)
20. Guo, J.-M., Yang, Y., Wang, N.: Chaos-based gray image watermarking algorithm. In: International Conference on Uncertainty Reasoning and Knowledge Engineering, vol.1, pp. 158–160 (2011)
21. Lien, B.K.: Robust data hiding by Hilbert curve decomposition. In: IEEE Intelligent Information Hiding and Multimedia Signal Processing, pp. 937–940 (2009)
22. Sun Y.-Y., Kong R.-Q., Wang X.-Y.: An image encryption algorithm utilizing Mandelbrot set. In: International Workshop on Chaos-Fractal Theory and Its Applications, pp. 170–173 (2010)
23. Moreno, J., Otazu, X.: Image compression algorithm based on Hilbert scanning of embedded quadtrees: an introduction of the hi-set coder. In: IEEE International Conference on Multimedia and Expo, pp. 1–6 (2011)
24. Lin, X.-H., Cai, L.-D.: Scrambling research of digital image based on Hilbert curve. Chinese J. Stereology Image Anal. **9**(4), 224–227 (2004)
25. Shang, Z., Ren, H., Zhang, J.: A block location scrambling algorithm of digital image based on Arnold transformation. In: 9th International Conference for Young Computer Scientists, pp. 2942–2947 (2008)
26. Zou, W.-G., Huang, J.-Y., Zhou C.-Y.: Digital image scrambling technology based on two dimension Fibonacci transformation and its periodicity. In: 3rd International Symposium on Information Science and Engineering, pp. 415–418 (2010)
27. Wen M.-G., Huang S.-C., Han C.-C.: An information hiding scheme using magic squares. In: IEEE International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 556–560 (2010)
28. Wang, S., Xu, X.-S.: A new algorithm of Hilbert scanning matrix and its MATLAB program. J. Image Graph. **11**(1), 119–122 (2006)
29. Le, P.Q., Iliyasu, A.M., Dong, F.Y., Hirota, K.: Fast geometric transformation on quantum images. IAENG Int. J. Appl. Math. **40**(3), 113–123 (2010)
30. Jiang, N., Wu, W.-Y., Wang, L.: The quantum realization of Arnold and Fibonacci image scrambling. Quantum Inf. Process. accepted.
31. Peano, G.: Sur une courbe, qui remplit toute une aire plane. Math. Ann. **36**(1), 157–160 (1890)
32. Hilbert, D.: Ueber die stetige Abbildung einer Linie auf ein Flächenstück. Math. Ann. **38**(3), 459–460 (1891)
33. Kamata, S., Eason, R.O., Bandou, Y.: A new algorithm for N-dimensional Hilbert scanning. IEEE Trans. Image Process. **8**(7), 964–973 (1999)