Quantum Inf Process (2015) 14:4001–4026
DOI 10.1007/s11128-015-1099-5

Quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio

Nan Jiang[1] · **Jian Wang**[2] · **Yue Mu**[2]

Received: 18 August 2014 / Accepted: 18 August 2015 / Published online: 29 August 2015

Abstract Quantum image processing is one of the most active fields in quantum computation and quantum information processing. Some concepts of quantum images and transformations have emerged in recent years. This paper proposes a quantum algorithm to scale up quantum images based on nearest-neighbor interpolation with integer scaling ratio. Firstly, the novel enhanced quantum representation is improved to the generalized quantum image representation to represent a quantum image with arbitrary size $H \times W$. Then, nearest-neighbor interpolation is used to create new pixels in the enlarged images. Based on them, quantum image scaling up algorithms in the form of circuits are proposed.

Keywords Quantum image processing · Image scaling · Nearest-neighbor interpolation · Quantum image representation

1 Introduction

In 1982, Feynman [1] proposed a novel computation model, named quantum computers which are physical machines that can accept input states which represent a coherent superposition of many different possible inputs and then parallel evolve them into a

This work is supported by the Fundamental Research Funds for the Central Universities under Grants No. 2015JBM027 and the National Scholarship under Grants Nos. 201406545034 and 201507095087.

✉ Jian Wang
wangjian@bjtu.edu.cn

1 College of Computer, Beijing University of Technology, Beijing 100124, China

2 School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

corresponding superposition of outputs. By this way, quantum computers can efficiently solve some problems which are believed to be intractable on any classical computer. A quantum computer will be viewed as a quantum network (or a quantum circuit) composed of quantum logic gates, each gate performing an elementary unitary operation on one, two or more two-state quantum systems called qubits [2].

The developing of quantum computer causes peoples interest to study quantum image processing (QIP) because a computer without images and graphs has already been unacceptable and quantum computers must have the function of image processing. At present, QIP has two main directions: (1) representations of quantum images and (2) QIP algorithms. We introduce them, respectively, in the following.

1.1 Representations of quantum images

In order to process images in quantum computers, the first thing should be done is to store digital images into quantum computers, which is called the representation of quantum images.

In 2003, Venegas-Andraca et al. [3] propose the Qubit Lattice method to store quantum images into two-dimensional arrays of qubits. One qubit holds one pixel. In 2010, they introduce a new method—Entangled Image, for storing binary geometrical shapes in quantum systems [4].

Real Ket uses quantum superposition to store image information [5]. This model performs image quartering iteratively and builds a balanced quadtree for all the pixels. In the quantum image, every pixel is represented as a basis state of a four-dimensional qubit sequence.

In 2011, Le et al. [6] give a flexible representation of quantum images (FRQI)which uses $2n + 1$ qubits to store a $2^n \times 2^n$ image: n qubits for X-coordinate, n qubits for Y-coordinate and 1 qubits for color. FRQI fully exploits the physical properties of qubits and reduces the number of qubits used to store a $2^n \times 2^n$ image from $2^n \times 2^n$ to $2n + 1$. Furthermore, FRQI resolves the real-time computation problem of image processing and provides a flexible method to process any part of a quantum image by using controlled quantum logic gates [7]. However, it cannot retrieve the pixels' value accurately through finite quantum measurements because it store the grayscale information as the probability amplitude of a single qubit.

In 2013, two groups of researchers improve the color information of FRQI. MCQI uses three qubits to represent R, G and B components of color images [8]. A novel enhanced quantum representation of digital images (NEQR) uses the basis state of a qubit sequence with q-qubit to store the grayscale value of every pixel [9]. NEQR inherits the merits of FRQI and the color information of it can be accurately measured. Nevertheless, because NEQR does not improve the location information of FRQI, they only can represent square images in the size of $2^n \times 2^n$ which is not in accordance with the fact that most practical images are rectangular. In order to solve the problem, INEQR is proposed to represent quantum images having $2^{n_1} \times 2^{n_2}$ pixels by modifying the position information of NEQR [10]. The inventors of NEQR provide another representation called QUALPI (quantum log-polar image) in 2013. QUALPI stores images sampled in log-polar coordinates [11].

Table 1 The quantum image representations

Name	Year	Inventor(s)	Features (assume that the image size is $2^n \times 2^n$)
Qubit Lattice [3]	2003	Venegas-Andraca	1. One qubit holds one pixel
			2. Need 2^{2n} qubits
Real Ket [5]	2005	Latorre	1. Use quantum superposition to store image information
Entangled Image [4]	2010	Venegas-Andraca	1. Store binary geometrical shapes in quantum systems
FRQI [6]	2011	Le	1. Need $2n + 1$ qubits
			2. Resolve the real-time computation problem of image processing
			3. Provides a flexible method to process any part of an image
			4. Cannot retrieve the pixels' value accurately
MCQI [8]	2013	Sun	1. Similar to FRQI
			2. Use three qubits to represent R, G and B components of color images
			3. Need $2n + 3$ qubits
NEQR [9]	2013	Zhang	1. Use the basis state to store the grayscale value
			2. Can be accurately measured
			3. only represent square images in the size of $2^n \times 2^n$
			4. Need $2n + q$ qubits
QUALPI [11]	2013	Zhang	1. Store images sampled in log-polar coordinates
QSMC&QSNC [12]	2013	Li	1. Need $2^{2n+1} + 2^q$ qubits
NAQSS [13]	2014	Li	1. Need $2n + 1$ qubits
			2. 1 Qubit represents an image segmentation information
			3. The preparing and the retrieving need a large amount of calculation
INEQR [10]	2014	Jiang	1. Similar to NEQR
			2. Can represent $2^{n1} \times 2^{n2}$ images

Li et al. [12] give QSMC&QSNC method which is extended from Qubit Lattice and uses $2N + m$ qubits to hold an image of N pixels and m different colors. In 2014, NAQSS (normal arbitrary quantum superposition state) is proposed by Li et al. [13]. It uses $n + 1$ qubits to represent an image with 2^n pixels, where n qubits represent colors and coordinates and the remaining 1 qubit represents an image segmentation information. However, the preparing and the retrieving need a large amount of calculation and potentially the values of all the qubits should be changed with only one pixel is altered.

We summarize the quantum image representations in Table 1.

1.2 QIP algorithms

There have been some quantum image processing algorithms. We introduce them by type.

1. Simple geometric transformation: In 2010, some simple geometric transformations such as the two-point swapping, flip, coordinate swapping, orthogonal rotations (90°, 180° and 270°) and their variants for quantum images, specifically those based on the FRQI representation, are proposed using the basic quantum gates, NOT, CNOT and Toffoli gates [7]. It provides a flexible method to process any part of a quantum image.
2. Image translation: Image translation, which maps the position of each picture element into a new position, is a basic image transformation. Ref. [14] gives two types of translation algorithms: entire translation and cyclic translation by providing the quantum circuits.
3. Image scaling: The main aim of image scaling, which has been extensively studied and widely used as a basic image transformation method in classical image processing field, is to resize a digital image. In image scaling, interpolation methods are necessary to produce new pixels (when scaling up) or delete redundant pixels (when scaling down). The commonly used interpolation methods include nearest neighbor, bilinear and bicubic [15,16]. Ref. [10] gives a quantum image scaling method, including scaling up and scaling down, with $2^{r_y} \times 2^{r_x}$ scaling ratio based on nearest-neighbor interpolation.
4. Color transformation: In image processing, operations to process grayscale information play an important role in many complex image algorithms. In Ref. [9], several color transformations are proposed, such as inversion, binaryzation, halving and so on.
5. Image scrambling: The main aim of image scrambling, which is generally used as the preprocessing or postprocessing in the confidentiality storage and transmission, and image information hiding, is to transform a meaningful image into a meaningless or disordered image in order to enhance the image security. Jiang et al. [17,18] give three quantum image scrambling algorithms: Arnold , Fibonacci [17] and Hilbert [19].
6. Image segmentation: Image segmentation subdivides an image into its constituent regions or objects. Venegas-Andraca first mentions quantum image segmentation issue in Ref. [4]. Li et al. [12] give another segmentation scheme based on the extended Grover's quantum search algorithm in 2013. However, it requires repeated tests to determine the parameters' values.
7. Feature extraction: In 2014, Ref. [20] proposes a quantum feature extraction framework based on NEQR. The feature points could be extracted by comparing and thresholding the gradients of the pixels
8. Quantum image watermark and quantum image encryption: Quantum image watermark is to hide a message into a carrier image, and quantum image encryption is to make an image unreadable. Many watermark schemes [21–27] and encryption methods [28–31] have been proposed. However, since we think that they belong to the application of image processing instead of QIP itself, we do not introduce them further. Interested readers can refer to Ref. [21–31].

In this paper, a quantum image scaling up scheme is proposed based on the nearest-neighbor interpolation with integer scaling ratio. The main differences between this paper and Ref. [10] are that:

1. INEQR proposed in Ref. [10] only can represent quantum image sized $2^{n_1} \times 2^{n_2}$, such as 16×4, 128×256 and so on. The representation discussed in this paper can represent arbitrary sized image.
2. The scaling up ratio is improved from $2^{r_y} \times 2^{r_x}$ to $r_y \times r_x$, where $r_y, r_x \in \mathcal{N}$ and $\mathcal{N}$ is the set of nature numbers.

The rest of the paper is organized as follows. A new representation method—the generalized quantum image representation (GQIR)—is presented in Sect. 2. The quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio is discussed in Sect. 3 including the principle, the quantum circuit architecture and the network complexity. An example also is given in this section to describe the scaling up circuit more clearly. Section 4 discusses how to simplify the circuit. Finally, a conclusion is given in Sect. 5.

2 The generalized quantum image representation (GQIR)

As introduced in Sect. 1.1, NEQR has some merits. However, there are two reasons that prompt us to improve NEQR:

- It only can represent square images in the size of $2^n \times 2^n$ which is not in accordance with the fact that most practical images are rectangular, such as 1024×768, 1440×900 and so on.
- In image scaling, if the scaling ratio r is a positive integer, the size of the scaled image will not always be in the form of $2^n \times 2^n$. For example, if a 16×16 image is scaled 3×5 times, the size of the scaled image will be 48×80 which can not be represented in NEQR.

In order to solve the problems, we improve NEQR's location information and call the new method as the generalized quantum image representation (GQIR) which can represent a quantum image sized $H \times W$, where $H, W \in \mathcal{N}$ are arbitrary positive integers.

2.1 GQIR

GQIR which is developed from NEQR can store arbitrary $H \times W$ quantum images. Both the position information and the color information are captured into normalized quantum states: $|0\rangle$ and $|1\rangle$. It uses $h = \lceil \log_2 H \rceil$ qubits for Y-coordinate and $w = \lceil \log_2 W \rceil$ qubits for X-coordinate to represent a $H \times W$ image. However, this will generate a $(2^h - H)$-row and $(2^w - W)$-column redundancy. The redundancy is cased by the intrinsic property of the binary computation and is inevitable. For example, if we want to use binary codes to represent five symbols, the code word length is $\lceil \log_2 5 \rceil = 3$ and only 000, 001, 010, 011, 100 are useful. The remaining three code words 101, 110, 111 are redundancies which are inevitable and will not be processed.

In GQIR, a similar situation exists. Since Hadamard gates are used on the location qubits to let state $|0\rangle$ and state $|1\rangle$ appear with equal probability (the function of Hadamard gate will be introduced in Sect. 3.2), $h + w$ location qubits will generate a

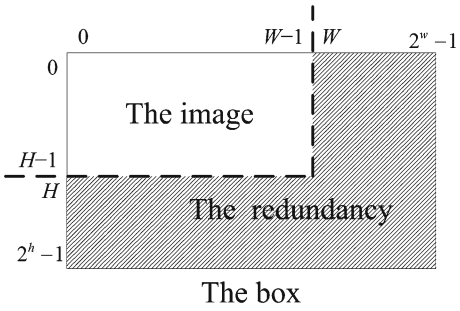


Fig. 1 The diagram of GQIR box

$2^h \times 2^w$ blank image. We call it as a $2^h \times 2^w$ box. However, only $H \times W$ pixels of the box are useful, and other $2^h \times 2^w - H \times W$ "pixels" are redundancies. GQIR puts the effective $H \times W$ pixels, i.e., the image to be represented, in the upper left corner of the box. All the redundant pixels are reserved as $|0\rangle$. Figure 1 is a diagram of GQIR box in which the white part is the image to be represented and the shaded area is the redundancy.

Hence, an GQIR image can be written as below.

$$\begin{aligned} |I\rangle &= \frac{1}{\sqrt{2}^{h+w}} \left(\sum_{Y=0}^{H-1} \sum_{X=0}^{W-1} \otimes_{i=0}^{q-1} |C_{YX}^i\rangle |YX\rangle \right) \\ |YX\rangle &= |Y\rangle|X\rangle = |y_0 y_1 \dots y_{h-1}\rangle |x_0 x_1 \dots x_{w-1}\rangle, \ y_i, x_i \in \{0, 1\} \\ |C_{YX}\rangle &= |C_{YX}^0 C_{YX}^1 \dots C_{YX}^{q-1}\rangle, C_{YX}^i \in \{0, 1\} \end{aligned} \tag{1}$$

where

$$h = \begin{cases} \lceil \log_2 H \rceil, & H > 1 \\ 1, & H = 1 \end{cases} \tag{2}$$

$$w = \begin{cases} \lceil \log_2 W \rceil, & W > 1 \\ 1, & W = 1 \end{cases} \tag{3}$$

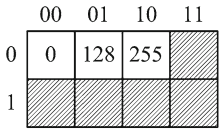
$|YX\rangle$ is the location information and $|C_{YX}\rangle$ is the color information. It needs $h+w+q$ qubits to represent a $H \times W$ image with gray range 2^q. Note that GQIR can represent not only grayscale images but also color images because the color depth q is a variable. In most cases, when $q = 2$, it is a binary image; when $q = 8$, it is a grayscale image; and when $q = 24$, it is a color image.

The box can be written as Eq. (4).

$$|B\rangle = |I\rangle + \frac{1}{\sqrt{2}^{h+w}} \left(\sum_{Y \in \{H, \dots, 2^h - 1\} \text{ or } X \in \{W, \dots, 2^w - 1\}} \otimes_{i=0}^{q-1} |0\rangle |YX\rangle \right) \tag{4}$$

Figure 2 shows a 1×3 grayscale image and its representative expression in GQIR.

$$H = 1, W = 3 \Rightarrow h = 1, w = \lceil \log_2 3 \rceil = 2$$



$$|I\rangle = \frac{1}{\sqrt{2^3}}\left(|00000000\rangle \otimes |000\rangle + |10000000\rangle \otimes |001\rangle + |11111111\rangle \otimes |010\rangle\right)$$

Fig. 2 A example image and its representative expression in GQIR

The 1×3 image is put into a $2^1 \times 2^2 = 2 \times 4$ box. In fact, only three pixels ($Y = 0$ and $X = 00/01/10$) are effective, and others are redundant but irremovable caused by the intrinsic property of binary expression.

2.2 Quantum image preparation

The preparation is the process that transforms quantum computers from the initial state to the desired quantum image state [6]. It is similar to the preparation of NEQR and can be divided into three steps [9]. However, in GQIR, the quantum image preparation is the preparation of the box because the redundancy is irremovable.

Step 0 Prepare $h + w + q$ qubits and set all of them to $|0\rangle$. The initial state can be expressed as in Eq. (5):

$$|\Psi\rangle_0 = |0\rangle^{\otimes h+w+q} \tag{5}$$

Step 1 q identity gates and $h+w$ Hadamard gates are used to construct a blank $2^h \times 2^w$ box. The identity matrix and the Hadamard matrix are shown below.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{6}$$

The whole quantum operation in this step can be expressed as U_1 in Eq. (7):

$$U_1 = I^{\otimes q} \otimes H^{\otimes h+w} \tag{7}$$

U_1 transforms the initial state $|\Psi\rangle_0$ to the intermediate state $|\Psi\rangle_1$.

$$\begin{aligned} |\Psi\rangle_1 &= U_1(|\Psi\rangle_0) = (I|0\rangle)^{\otimes q} \otimes (H|0\rangle)^{\otimes h+w} \\ &= \frac{1}{\sqrt{2^{h+w}}} |0\rangle^{\otimes q} \otimes \sum_{i=0}^{2^{h+w}-1} |i\rangle \\ &= \frac{1}{\sqrt{2^{h+w}}} \sum_{Y=0}^{2^h-1} \sum_{X=0}^{2^w-1} |0\rangle^{\otimes q} |YX\rangle \end{aligned} \tag{8}$$

In simple terms, the effect of identity gate is maintaining the qubit's original state unchanged and the effect of Hadamard gate is letting state $|0\rangle$ and state $|1\rangle$ appear with equal probability. After Step 1, the blank $2^h \times 2^w$ box is gained.

Step 2 Set the pixels' value pixel by pixel. Since there are $H \times W$ pixels in the image, Step 2 is divided into $H \times W$ suboperations to store the color information for every pixel. For pixel (Y, X), the quantum suboperation U_{YX} is shown below.

$$U_{YX} = \left(I \otimes \sum_{ji \neq YX} |ji\rangle\langle ji| \right) + \Omega_{YX} \otimes |YX\rangle\langle YX| \tag{9}$$

where Ω_{YX} is the quantum operation to transform the value of pixel (Y, X) from $|0\rangle^{\otimes q}$ to the desired value as shown in Eq. (10). Hence, the function of U_{YX} is change pixel (Y, X) and others remain unchanged.

$$\Omega_{YX} = \otimes_{i=0}^{q-1} \Omega_{YX}^{i} \tag{10}$$

The function of Ω_{YX}^{i} is setting the value of the ith qubit of pixel (Y, X)'s color information.

$$\Omega_{YX}^{i} : |0\rangle \rightarrow |0 \oplus C_{YX}^{i}\rangle \tag{11}$$

where $\oplus$ is the XOR operation. If $C_{YX}^{i} = 1$, $\Omega_{YX}^{i} : |0\rangle \rightarrow |1\rangle$ is a $(h + w)$-CNOT gate (a CNOT gate with $(h + w)$ control qubits). Otherwise, $\Omega_{YX}^{i} : |0\rangle \rightarrow |0\rangle$ is a quantum identity gate which will do nothing on the quantum state. Hence,

$$\Omega_{YX}|0\rangle^{\otimes q} = \otimes_{i=0}^{q-1}(\Omega_{YX}^{i}|0\rangle) = \otimes_{i=0}^{q-1}|0 \oplus C_{YX}^{i}\rangle = \otimes_{i=0}^{q-1}|C_{YX}^{i}\rangle = |C_{YX}\rangle \tag{12}$$

Act U_{YX} on $|\Psi\rangle_1$

$$\begin{aligned} U_{YX}(|\Psi\rangle_1) &= U_{YX}\left(\frac{1}{\sqrt{2}^{h+w}} \sum_{j=0}^{2^h-1} \sum_{i=0}^{2^w-1} |0\rangle^{\otimes q}|ji\rangle \right) \\ &= \frac{1}{\sqrt{2}^{h+w}} U_{YX}\left(\sum_{ji \neq YX} |0\rangle^{\otimes q}|ji\rangle + |0\rangle^{\otimes q}|YX\rangle \right) \\ &= \frac{1}{\sqrt{2}^{h+w}} U_{YX}\left(\sum_{ji \neq YX} |0\rangle^{\otimes q}|ji\rangle + \Omega_{YX}|0\rangle^{\otimes q}|YX\rangle \right) \\ &= \frac{1}{\sqrt{2}^{h+w}} U_{YX}\left(\sum_{ji \neq YX} |0\rangle^{\otimes q}|ji\rangle + |C_{YX}\rangle|YX\rangle \right) \end{aligned} \tag{13}$$

U_{YX} only sets the color value of its corresponding pixel. In order to set all the $H \times W$ pixels, a quantum operation U_2 is defined below.

$$U_2 = \prod_{Y=0}^{H-1} \prod_{X=0}^{W-1} U_{YX} \tag{14}$$

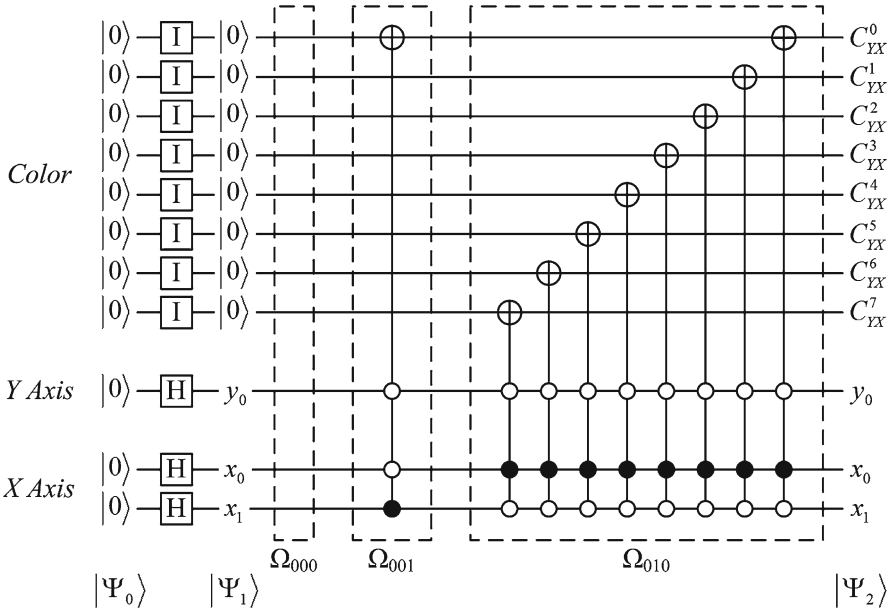


Fig. 3 The quantum circuit of the 1×3 image shown in Fig. 2

Act U_2 to $|\Psi\rangle_1$

$$
\begin{aligned}
|\Psi\rangle_2 &= U_2(|\Psi\rangle_1) \\
&= \frac{1}{\sqrt{2^{h+w}}}\left(\sum_{Y=0}^{H-1}\sum_{X=0}^{W-1} \Omega_{YX}|0\rangle^{\otimes q}|YX\rangle + \sum_{Y\in\{H,\ldots,2^h-1\} \text{ or } X\in\{W,\ldots,2^w-1\}} \otimes_{i=0}^{q-1}|0\rangle|YX\rangle\right) \\
&= \frac{1}{\sqrt{2^{h+w}}}\left(\sum_{Y=0}^{H-1}\sum_{X=0}^{W-1} |C_{YX}\rangle|YX\rangle + \sum_{Y\in\{H,\ldots,2^h-1\} \text{ or } X\in\{W,\cdots,2^w-1\}} \otimes_{i=0}^{q-1}|0\rangle|YX\rangle\right) \qquad (15)
\end{aligned}
$$

In simple terms, the effect of Step 2 is using $(h + w)$-CNOT gates to change the effective $H \times W$ pixels to the desired quantum image states.

Figure 3 shows the detailed quantum circuit for GQIR preparation for the example image shown in Fig. 2. Only three effective pixels $|YX\rangle = |000\rangle, |001\rangle, |010\rangle$ are set to desired value. Others (the reserved pixels) remain unchanged, i.e., are still in their initial state $|0\rangle$.

GQIR is evolved from NEQR. The main merit of it is that it can represent image with arbitrary size instead of $2^n \times 2^n$. Since the type of the location qubits and color qubits in GQIR is the same as NEQR's—in basis state $|0\rangle$ or $|1\rangle$, the improvement does not damage the efficiency and the measurability of the representation. Readers can get more detailed information about them from Section 3.2 and 4.1 of Ref. [9] by replacing n with $\frac{h+w}{2}$.

3 Quantum image scaling up

In this paper, we solve the problem of quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio. The principle of this operation is given first. Then, the scaling up circuit is proposed.

3.1 The principle of image scaling up

Image scaling is the process of resizing a digital image. It has two types: enlarging an image by adding new pixels or lessening an image by deleting redundant pixels. In this process, interpolation is indispensable to determine what the values of the newly added pixels are or which redundant pixels should be deleted.

The commonly used interpolation methods in image processing include nearest neighbor, bilinear and bicubic [15,16]. No matter which interpolation is used, image scaling can be decomposed in two directions: first in one direction and then again in the other direction.

$$I' = S(I, r_x, r_y) = S_y(S_x(I, r_x), r_y) = S_x(S_y(I, r_y), r_x) \tag{16}$$

where S is the scaling function, I is the original image, I' is the scaled image, and r_x and r_y are the horizontal and vertical scaling ratio. S can be decomposed into S_x and S_y, which are the horizontal and vertical scaling function. The principles of S_x and S_y are exactly the same and they are commutative. That is to say image scaling is the combination of two one-dimensional scaling. This paper only discusses scaling up, i.e., $r > 1$.

Figure 4 shows the principles of the three interpolations in one direction, in which the known pixels (the white circles in Fig. 4) are used to estimate the value of the destination pixel (the black circle in Fig. 4), i.e., the newly added pixel.

(1) nearest-neighbor method

The nearest-neighbor method selects the value of the nearest known pixel as the value of the destination pixel. In Fig. 4a,

$$c = c_0$$

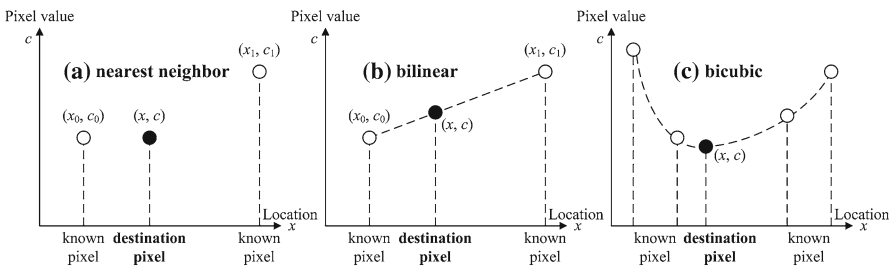


Fig. 4 The principle of interpolation in one direction

because (x_0, c_0) is the nearest neighbor of (x, c).

When implemented on a computer, the essence of nearest-neighbor interpolation is repetition. For example, if the pixel value of a 1×4 original image is

ABCD

and the scaling ratio is 1×3, the scaled up image will be

AAABBBCCCDDD

That is to say, if the scaling ratio is $r_y \times r_x$, the pixel (i, j) in the original image is scaled to $r_y \times r_x$ pixels

$$\begin{array}{ccc} (r_y i + 0, r_x j + 0), & \ldots, & (r_y i + 0, r_x j + (r_x - 1)) \\ (r_y i + 1, r_x j + 0), & \ldots, & (r_y i + 1, r_x j + (r_x - 1)) \\ \ldots, & \ldots, & \ldots \\ (r_y i + (r_y - 1), r_x j + 0), & \ldots, & (r_y i + (r_y - 1), r_x j + (r_x - 1)) \end{array} \tag{17}$$

in the scaled up image (see Fig. 5).

(2) bilinear method

As shown in Fig. 4b, bilinear interpolation estimates the value of the destination pixel (x, c) according to the two distances $x - x_0$ and $x_1 - x$. A straight line is used to connect the two known pixels. Hence,

$$\frac{c - c_0}{x - x_0} = \frac{c_1 - c}{x_1 - x}.$$

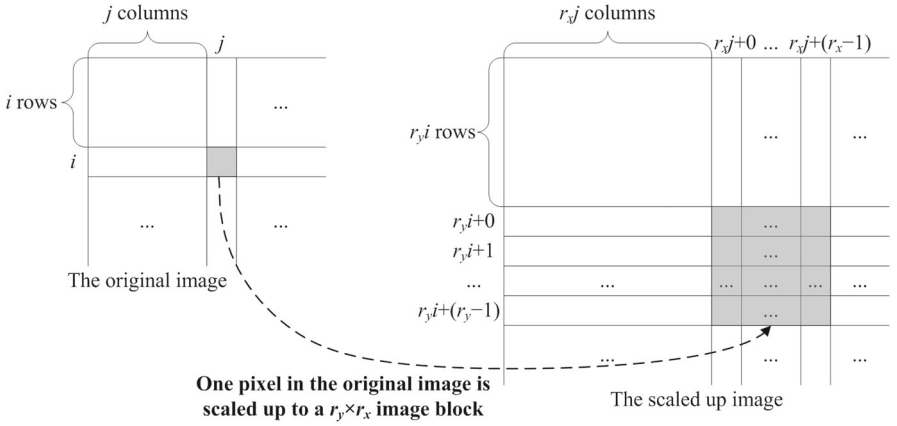


Fig. 5 The essence of image scaling up based on nearest-neighbor interpolation. One pixel in the original image is enlarged to a $r_y \times r_x$ image block in the scaled up image. Since there are i rows and j columns before pixel (i, j) in the original image, there are $r_y i$ rows and $r_x j$ columns before the corresponding image block in the scaled up image. Hence, the image block is shown in Eq. (17)

Fig. 6 The effect of different interpolation methods. **a** Original image. **b** Nearest neighbor. **c** Bilinear. **d** Bicubic

That is to say

$$c = \frac{(c_1 - c_0)x - c_1 x_0 + c_0 x_1}{x_1 - x_0}.$$

(3) bicubic method

As shown in Fig. 4c, bicubic interpolation estimates c according to not only the distances between the destination pixel and known pixels, but also the change rates of known pixels. So, it must use four known pixels to get the change rate. (Because the formula of bicubic interpolation and its derivation are complex and will not be used in the following, we omit them for the sake of simplicity.)

From nearest neighbor to bicubic, as the algorithm complexity is higher and higher, the interpolation effect is more and more smooth. Figure 6 gives an example about the three interpolation methods, in which the original image is enlarged twice. Although the effect of the nearest-neighbor method is crudest, this paper still uses it because it is the simplest one.

3.2 The quantum scaling up scheme

Assume that a $H \times W$ quantum image $|I\rangle$ is scaled up to a $H' \times W'$ quantum image $|I'\rangle$ based on nearest-neighbor interpolation. The scale ratio is $r_y \times r_x$, i.e., $H' = r_y H$ and $W' = r_x W$, where $r_x, r_y \in \mathcal{N}$. According to Fig. 5, the quantum scaling up scheme is shown in Algorithm 1.

Step 1 in Algorithm 1 is similar to Step 0 and Step 1 of Sect. 2.2. $h' + w'$ Hadamard gates are used to change the initial state $|0\rangle^{\otimes h'+w'+q}$ to $\frac{1}{\sqrt{2^{h'+w'}}} \sum_{Y'=0}^{2^{h'}-1} \sum_{X'=0}^{2^{w'}-1} |0\rangle^{\otimes q} |Y'X'\rangle$. The initial value of $|C'_{Y'X'}\rangle = |C'^{0}_{Y'X'} C'^{1}_{Y'X'} \cdots C'^{(q-1)}_{Y'X'}\rangle$ is $|0\rangle^{\otimes q}$. Step 1 indicates that the quantum image scaling up scheme is to generate a new image as the scaled one instead of modifying the original image into a scaled image.

Define an operation

$$U_{(r_y i+k),(r_x j+l)} = \bigotimes_{t=0}^{q-1} U^t_{(r_y i+k),(r_x j+l)}, \tag{18}$$

Algorithm 1 The quantum image scaling up algorithm based on nearest-neighbor interpolation with integer scaling ratio

Input:

The quantum image to be scaled, $|I\rangle = \frac{1}{\sqrt{2}^{h+w}} \left(\sum_{Y=0}^{H-1} \sum_{X=0}^{W-1} |C_{YX}\rangle |YX\rangle\right)$, where $h = \lceil \log_2 H \rceil$ and $w = \lceil \log_2 W \rceil$;

Output:

The scaled up quantum image, $|I'\rangle = \frac{1}{\sqrt{2}^{h'+w'}} \left(\sum_{Y'=0}^{H'-1} \sum_{X'=0}^{W'-1} |C'_{Y'X'}\rangle |Y'X'\rangle\right)$, where $h' = \lceil \log_2 H' \rceil$, $w' = \lceil \log_2 W' \rceil$, $H' = r_y H$ and $W' = r_x W$;

1: Generate a new $2^{h'} \times 2^{w'}$ blank box;
2: **for all** $i = 0$ to $(H-1)$ **do**
3: **for all** $j = 0$ to $(W-1)$ **do**
4: **for all** $k = 0$ to $(r_y - 1)$ **do**
5: **for all** $l = 0$ to $(r_x - 1)$ **do**
6: Set $|C'_{(r_y i+k)(r_x j+l)}\rangle = |C_{ij}\rangle$;
7: **end for**;
8: **end for**;
9: **end for**;
10: **end for**;

where

$$U^t_{(r_y i+k),(r_x j+l)} : |C'^t_{(r_y i+k),(r_x j+l)}\rangle = |0\rangle \rightarrow |0 \oplus C^t_{ij}\rangle = |C^t_{ij}\rangle. \tag{19}$$

$U^t_{(r_y i+k),(r_x j+l)}$ is a $(h+w+h'+w'+1)$-CNOT gate: h-qubit Y, w-qubit X, h'-qubit Y', w'-qubit X' and C^t_{ij} are the control qubits and $|C'^t_{(r_y i+k),(r_x j+l)}\rangle$ is the target qubit. When $Y = i$, $X = j$, $Y' = r_y i + k$ and $X' = r_x j + l$, $|C'^t_{(r_y i+k),(r_x j+l)}\rangle$ is set to $|C^t_{ij}\rangle$. Hence, if $Y = i$, $X = j$, $Y' = r_y i + k$ and $X' = r_x j + l$,

$$U_{(r_y i+k),(r_x j+l)} |0\rangle^{\otimes q} = |C^0_{ij} C^1_{ij} \cdots C^{q-1}_{ij}\rangle, \tag{20}$$

i.e., the pixel $|C'_{(r_y i+k),(r_x j+l)}\rangle$ in the scaled image $|I'\rangle$ is set to desired color information $|C_{ij}\rangle$. That is to say, $U_{(r_y i+k),(r_x j+l)}$ realizes Step 6 in Algorithm 1. Put it in the loop (Step 2–5) in Algorithm 1, i.e., repeat it $r_y r_x H W$ times, will realize the function of image scaling up.

3.3 The notation of $U_{(r_y i+k),(r_x j+l)}$

Equation (19) indicates that $U^t_{(r_y i+k),(r_x j+l)}$ is a $(h+w+h'+w'+1)$-CNOT gate. However, in general, $h+w+h'+w'$ is large and the CNOT gate has too many control qubits. For example, if the size of an image is scaled from 512×512 to 1024×1024, we will get $h = w = 9$ and $h' = w' = 10$ and the CNOT gate will have 39 control qubits. It is not easy to draw so many control qubits in quantum circuits. In order to solve the problem, a circuit module CV(v) is given.

CV(v) is a control module with a n-qubit binary quantum sequence as its input, where $0 \leq v \leq 2^n - 1$.

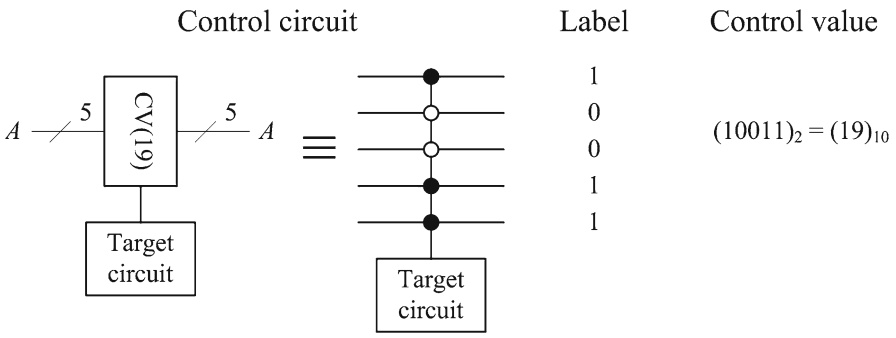
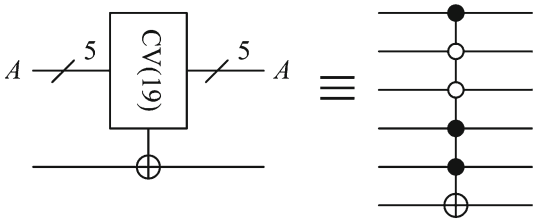


Fig. 7 An example for CV(v) and control value

Fig. 8 A 5-CNOT gate



In order to describe the function of CV(v), we propose the definition of control value firstly. Quantum circuits have two control conditions: ∘ and •. If we use a 1-bit binary label to represent the two control conditions: ∘ as 0 and • as 1, respectively, and concatenate all the binary labels from top to bottom, it can be viewed as a binary number which defined as the control value of the control circuit.

In quantum module CV(v), v is the decimal control value. Figure 7 gives an example for CV(v) and control value. The circuit has $n = 5$ control qubits and the control value is 19.

Note that the quantum module CV(v) is only a control module. No matter whatever the target circuit is, CV(v) is used to control when the target circuit will be triggered. The target circuit can be NOT gate, SWAP gate, or other more complex circuits. For example, if the target circuit is a NOT gate, Fig. 7 is externalized as Fig. 8. It is a 5-CNOT gate.

Hence, $U^t_{(r_y i+k),(r_x j+l)}$ is shown in Fig. 9a: When $Y = i$, $X = j$, $Y' = r_y i + k$ and $X' = r_x j + l$, $|C'^t_{(r_y i+k),(r_x j+l)}\rangle$ is set to $|C^t_{ij}\rangle$.

Figure 9b gives the circuit of $U_{(r_y i+k),(r_x j+l)} = \otimes_{t=0}^{q-1} U^t_{(r_y i+k),(r_x j+l)}$. We also use $U_{(r_y i+k),(r_x j+l)}$ to note the circuit module. However, notice that the module $U_{(r_y i+k),(r_x j+l)}$ has nothing to do with i, j, $r_y i + k$ and $r_x j + l$. We simplify $U_{(r_y i+k),(r_x j+l)}$ as U. It has four CV control modules which correspond to the loop (Step 2–5) in Algorithm 1. The module U with four CV control modules accomplishes Step 6.

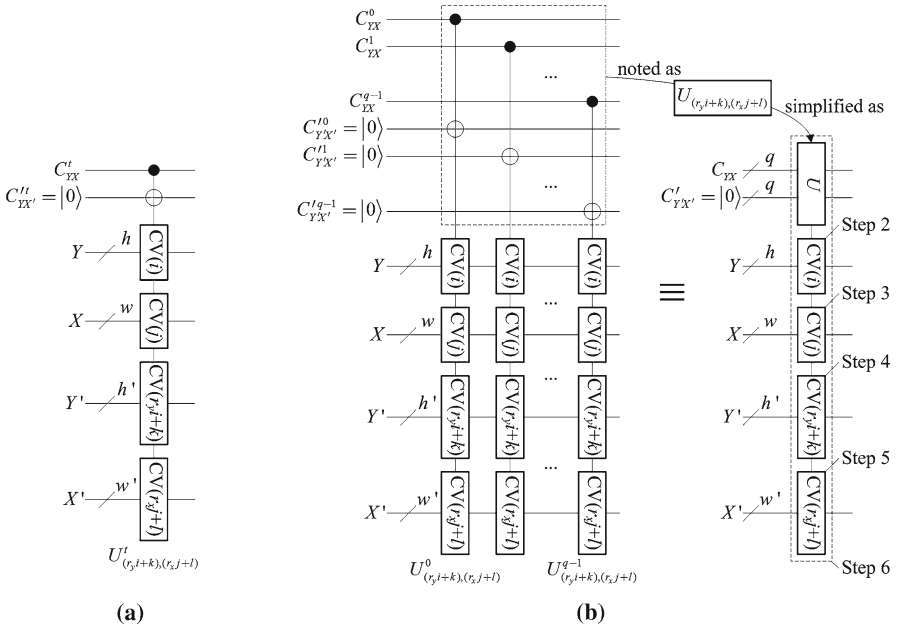


Fig. 9 The circuit modules $U^t_{(r_y i+k),(r_x j+l)}$ and $U_{(r_y i+k),(r_x j+l)}$

3.4 The quantum scaling up circuit

As discussed in Sect. 3.3, the module U with four CV control modules accomplishes Step 6. Next it should be repeated $r_y r_x HW$ times to realize the whole scaling up circuit. However, the repetition is accompanied by the change of parameters:

- ⋆ the control value of CV(i) is changed from 0 to $H-1$;
- ⋆ the control value of CV(j) is changed from 0 to $W-1$;
- ⋆ the control value of CV($r_y i+k$) is changed from $r_y i+0$ to $r_y i+(r_y-1)$;
- ⋆ the control value of CV($r_x j+l$) is changed from $r_x j+0$ to $r_x j+(r_x-1)$,

which are corresponding to Step 2–5 in Algorithm 1, respectively.

The structure of the quantum scaling up circuit is shown in Fig. 10.

In the first part in Fig. 10, $h'+w'$, Hadamard gates are used to derive $y'_0, y'_1, \ldots, y'_{h'-1}$ and $x'_0, x'_1, \ldots, x'_{w'-1}$, which is the Step 1 in Algorithm 1. The remaining part realizes Step 2–6 in Algorithm 1.

3.5 An example

This section gives a simple example to describe the scaling up circuit more clearly. Suppose the gray range of a 1×2 image is 2^1, i.e., $H=1$, $W=2$ and $q=1$. Hence $h=1$ and $w=1$. The scaling ratio is $r_y \times r_x = 5\times 3$. Then, the size of the scaled image is 5×6 and $h'=\lceil \log_2 5\rceil = 3$, $w'=\lceil \log_2 6\rceil = 3$. Figures 11 and 12 show the circuit and its effect.

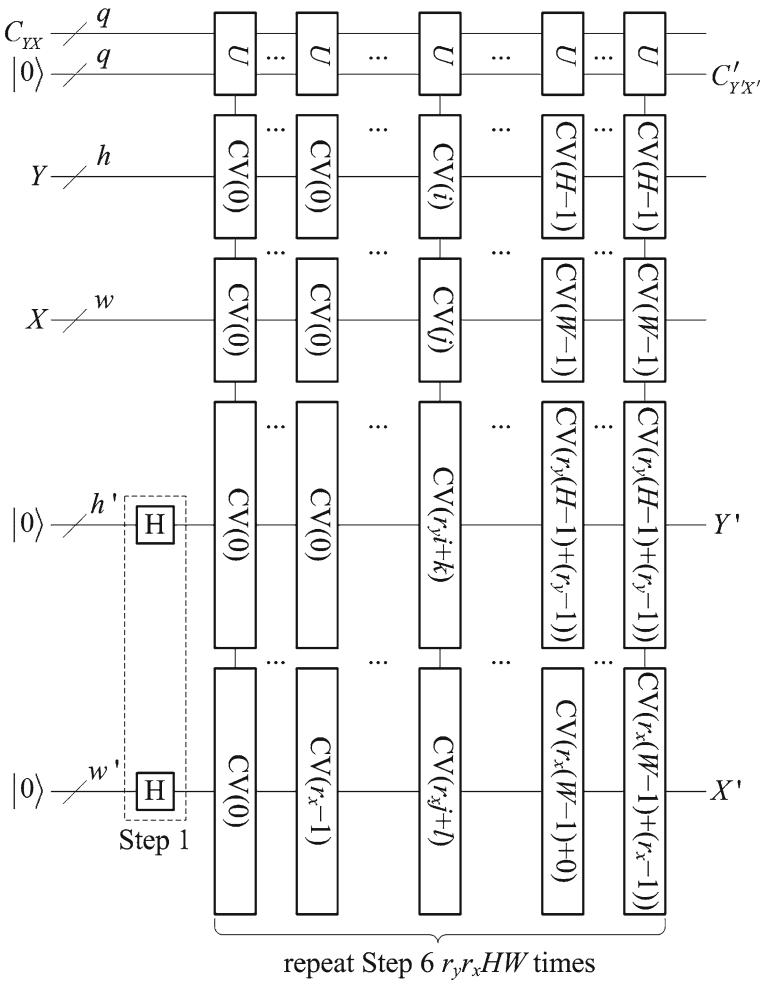


Fig. 10 The image scaling up circuit

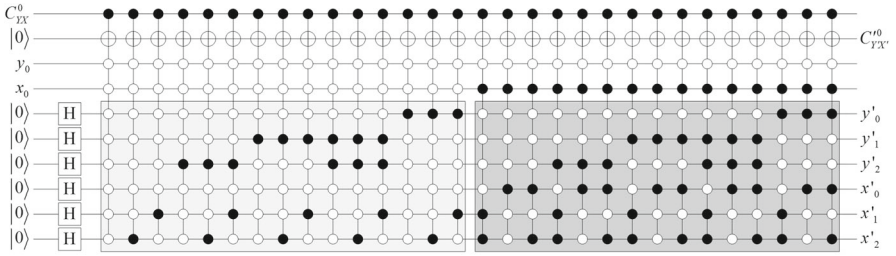


Fig. 11 The image scaling up circuit for the example

Figure 11 has 30 layers. The first 15 layers (in the light gray background) enlarge pixel (0,0) in the original image to a 5 × 3 block in the scaled image (the light gray image block in Fig. 12), and the second 15 layers (in the dark gray background) enlarge

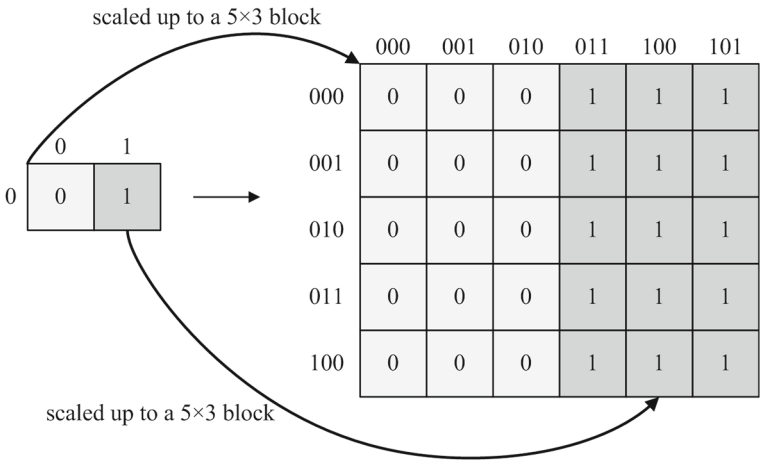


Fig. 12 The effect of the image scaling up circuit

pixel (0,1) in the original image to a 5×3 block in the scaled image (the dark gray image block in Fig. 12).

3.6 Network 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.

Ref. [34] points out that

- a n-Control-NOT gate ($n \geq 3$) is equivalent to $2(n-1)$ Toffoli gates and 1 Control-NOT gate as shown in Fig. 13a; and
- one Toffoli gate can be simulated by six Control-NOT gates.

Hence, the network complexity of a n-Control-NOT gate ($n \geq 3$) is $12n-11$. According to Fig. 13b, when the control value of the n-Control-NOT gate is not "11…1", the network complexity of the n-Control-NOT gate ($n \geq 3$) is

$$(12n-11)+2 = 12n-9 \tag{21}$$

Figure 9b indicates that the module U can be decomposed into q sublayers and each sublayer has $h+w+h'+w'+1$ control qubits. According to Eq. (21), the network complexity of module U is

$$q(12(h+w+h'+w'+1)-9) = q(12(h+w+h'+w')+3).$$

According to Fig. 10, the image scaling up circuit has $r_y r_x HWU$ modules. Hence the network complexity of image scaling up is

$$r_y r_x HWq(12(h+w+h'+w')+3) \tag{22}$$

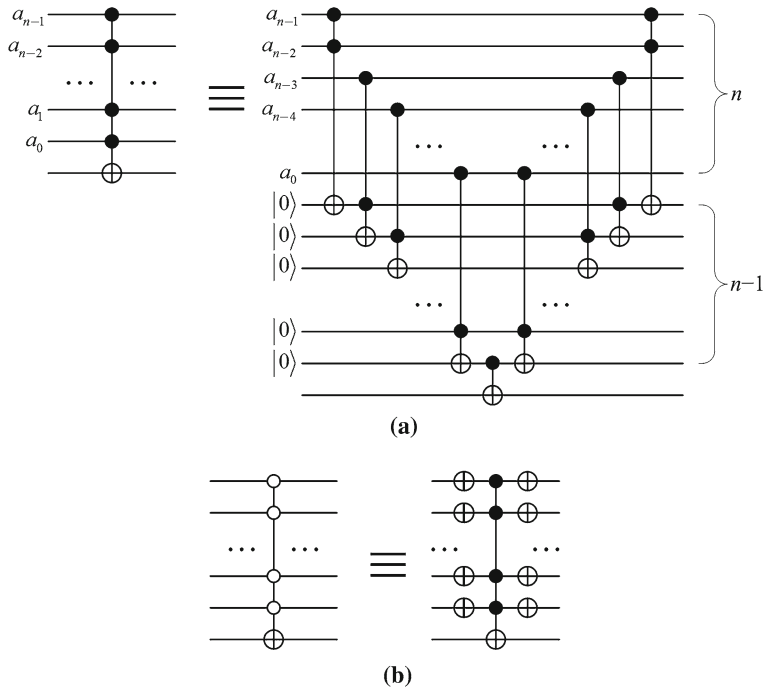


Fig. 13 Equivalence relation between logic gates. **a** n-Control-NOT gate. **b** Decomposition of "$\circ$"

For example, the network complexity of Fig. 11 is

$$5 \times 3 \times 1 \times 2 \times 1 \times (12(1 + 1 + 3 + 3) + 3) = 2970.$$

4 Simplify the circuit

The network complexity of the quantum image scaling up algorithm is not a small number [see Eq. (22)]. In this section, the simplification of the circuit is discussed, which is based on the minimization of Boolean expressions.

4.1 The principle

As stated in Sect. 3.3, control value is a binary string. There is a way to transform the control value (a binary string) to a Boolean minterm by considering each position in the binary string as a Boolean variable. If x is the Boolean variable at a position in the string and the value of that position is 1, then the lateral x is used in the minterm; otherwise, the lateral $\overline{x}$ is used. For example, the binary string 000 and 101 are equivalent to $\overline{x}_0\overline{x}_1\overline{x}_2$ and $x_0\overline{x}_1x_2$, respectively [6]. With this method, if there are two 3-CNOT gates and their control value are 000 and 001, then the corresponding Boolean expression is

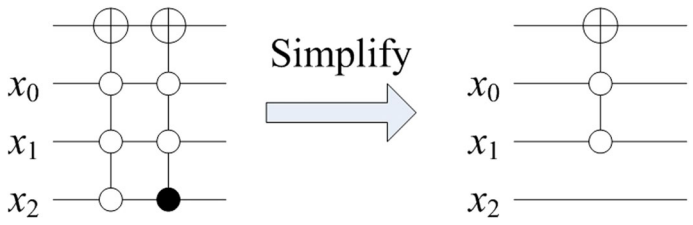


Fig. 14 Simplify a quantum gate

$$e = \overline{x}_0\overline{x}_1\overline{x}_2 + \overline{x}_0\overline{x}_1 x_2 = \overline{x}_0\overline{x}_1(\overline{x}_2 + x_2) = \overline{x}_0\overline{x}_1 \tag{23}$$

This denotes that two 3-CNOT gates are simplified to one 2-CNOT gate (see Fig. 14). As a consequence, the network complexity is dropped from $2(12 \times 3 - 9) = 54$ to $6 + 2 = 8$.

Equation (23) tells us that if only one control qubit of two n-CNOT gates is different and others are the same, the two n-CNOT gates can be simplified to one $(n-1)$-CNOT gate. In the following, the scaling up circuit is simplified based on the principle.

4.2 Simplify the scaling up circuit

Firstly, an operation set Λ consists of Step 2–10 of Algorithm 1.

$$\begin{aligned}\Lambda &= \bigcup_{i=0}^{H-1}\bigcup_{j=0}^{W-1}\bigcup_{k=0}^{r_y-1}\bigcup_{l=0}^{r_x-1}\bigcup_{t=0}^{q-1} U^t_{(r_y i+k),(r_x j+l)} \\ &= \bigcup_{t=0}^{q-1}\left(\bigcup_{i=0}^{H-1}\bigcup_{j=0}^{W-1}\bigcup_{l=0}^{r_x-1}\left(\bigcup_{k=0}^{r_y-1} U^t_{(r_y i+k),(r_x j+l)}\right)\right)\end{aligned} \tag{24}$$

If we fix the value of t, i, j and l and change k from 0 to $r_y - 1$, the corresponding circuit of expression $\bigcup_{t=0}^{q-1}\left(\bigcup_{i=0}^{H-1}\bigcup_{j=0}^{W-1}\bigcup_{l=0}^{r_x-1}\left(\bigcup_{k=0}^{r_y-1} U^t_{(r_y i+k),(r_x j+l)}\right)\right)$ is shown in Fig. 15. The fixed t indicates that all the $U^t_{(r_y i+k),(r_x j+l)}$ operate the tth qubit of color information. The fixed i, j and l indicate that the control values on Y, X and X' of the r_y $(h + w + h' + w' + 1)$-CNOT gates are all the same. Only the control value on Y' is increasing and the step is 1. This phenomenon makes it possible to simplify the circuit.

Theorem 1 gives the upper limit of the circuit's network complexity.

Theorem 1 *If N is used to denote the scaling up circuit's network complexity, then*

$$N \le \begin{cases} r_y r_x HWq\left(6a - \frac{9}{2}\right), & \text{if one of } r_y \text{ and } r_x \text{ are even number} \\ r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_x HWq\left(6a + \frac{15}{2}\right), & \text{if both } r_y \text{ and } r_x \text{ are odd number, and } r_y \ge r_x \\ r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_y HWq\left(6a + \frac{15}{2}\right), & \text{if both } r_y \text{ and } r_x \text{ are odd number, and } r_y < r_x \end{cases} \tag{25}$$

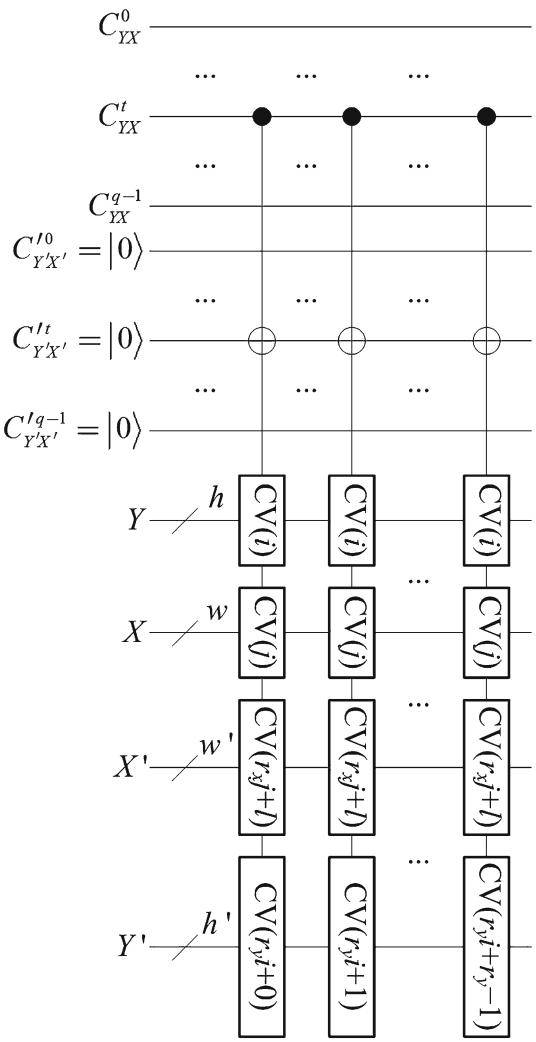


Fig. 15 Fix t, i, j and l and change k from 0 to $r_y - 1$

where $a = (h + w + h' + w')$.

Proof In order to simplify the entire scaling up circuit, we firstly study partial circuit as shown in Fig. 15 in which t, i, j and l are fixed.

Due to that the control values on Y, X and X' of the r_y $(h + w + h' + w' + 1)$-CNOT gates are all the same, their corresponding Boolean expressions are also the same. We use Y, X and X' to name them, respectively, i.e., the Boolean expression of $U^t_{(r_y i+k),(r_x j+l)}$ is

$$e_k = YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2}\widehat{y}'_{h'-1} \tag{26}$$

where $\widehat{y}'_i$ indicates that it may be y'_i or $\overline{y}'_i$.

In order to simplify the circuit based on Eq. (26), three cases are discussed.

Case 1. $r_y i$ is an even number, and r_y also is an even number.

Because $r_y i$ is an even number, the bottom control conditions of CV($r_y i + 0$), CV($r_y i + 2$), $\cdots$, CV($r_y i + r_y - 2$) are "$\circ$", i.e.,

$$e_k = YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2}\overline{y}'_{h'-1}, 0 \le k \le r_y - 2 \text{ and } k \text{ is an even number}$$

Hence,

$$e_{k+1} = YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2}y'_{h'-1}$$

Therefore,

$$\begin{aligned} e_k + e_{k+1} &= YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2}\overline{y}'_{h'-1} + YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2}y'_{h'-1} \\ &= YXX'\widehat{y}'_0\widehat{y}'_1 \cdots \widehat{y}'_{h'-2} \end{aligned}$$

That is to say, two $(h + w + h' + w' + 1)$-CNOT gates ($U^t_{(r_y i+k),(r_x j+l)}$ and $U^t_{(r_y i+k+1),(r_x j+l)}$) are simplified to one $(h + w + h' + w')$-CNOT gate. We call the two $(h + w + h' + w' + 1)$-CNOT gates as a pair. Therefore, the network complexity is dropped from

$$2(12(h + w + h' + w' + 1) - 9) = 24(h + w + h' + w') + 6$$

to

$$12(h + w + h' + w') - 9.$$

Because r_y is an even number, there are exactly $\frac{r_y}{2}$ pairs in Fig. 15. Hence, after simplification, the network complexity of Fig. 15 is

$$\frac{r_y}{2}(12(h + w + h' + w') - 9) = 6r_y(h + w + h' + w') - \frac{9r_y}{2}. \tag{27}$$

Case 2. $r_y i$ is an even number, and r_y is an odd number.

Because $r_y i$ is an even number, as stated in Case 1, for each pair, the network complexity is dropped to

$$12(h + w + h' + w') - 9.$$

However, r_y is an odd number which results in $\frac{r_y-1}{2}$ pairs and a remainder $(h + w + h' + w' + 1)$-CNOT gate (the last gate in Fig. 15). Hence, after simplification, the network complexity of Fig. 15 is

$$\frac{r_y-1}{2}(12(h+w+h'+w')-9)+12(h+w+h'+w'+1)-9$$
$$=6(r_y+1)(h+w+h'+w')-\frac{9r_y}{2}+\frac{15}{2}. \tag{28}$$

Case 3. $r_y i$ is an odd number, and r_y also is an odd number.

Because $r_y i$ is an odd number, the bottom control conditions of CV($r_y i+0$) are "•". Under this circumstance, $U^t_{(r_y i+0),(r_x j+l)}$ and $U^t_{(r_y i+1),(r_x j+l)}$ cannot be defined as a pair because maybe there is carry which makes more than one bit differences between $r_y i+0$ and $r_y i+1$. However, $U^t_{(r_y i+1),(r_x j+l)}$ and $U^t_{(r_y i+2),(r_x j+l)}$ can be defined as a pair because the bottom control conditions of CV($r_y i+1$) are "∘". Similarly, the network complexity of a simplified pair is

$$12(h+w+h'+w')-9.$$

Similar to Case 2, there are still $\frac{r_y-1}{2}$ pairs and a remainder $(h+w+h'+w'+1)$-CNOT gate (the first gate in Fig. 15). Hence, after simplification, the network complexity of Fig. 15 is

$$6(r_y+1)(h+w+h'+w')-\frac{9r_y}{2}+\frac{15}{2}. \tag{29}$$

To the entire scaling up circuit, according to Eq. (24), the partial circuit shown in Fig. 15 is repeated $qHWr_x$ times. Therefore, the network complexity of the entire scaling up circuit is divided into two cases.

(1) r_y is an even number

No matter what the value of i, $r_y i$ is an even number because r_y is even. Hence, according to Case 1 [Eq. (27)], the network complexity of the entire circuit is

$$qHWr_x\left(6r_y(h+w+h'+w')-\frac{9r_y}{2}\right)$$
$$=r_y r_x HWq\left(6a-\frac{9}{2}\right). \tag{30}$$

(2) r_y is an odd number

Because i is changed from 0 to $H-1$, Case 2 and Case 3 appear alternately. However, no matter $r_y i$ is even or odd, the network complexity is the same [see Eqs. (28) and (29)]. Hence, the network complexity of the entire circuit is

$$qHWr_x\left(6(r_y+1)(h+w+h'+w')-\frac{9r_y}{2}+\frac{15}{2}\right)$$
$$=r_y r_x HWq\left(6a-\frac{9}{2}\right)+r_x HWq\left(6a+\frac{15}{2}\right). \tag{31}$$

Equations (30) and (31) are gained from the angle of Y'. If we exchange r_y and r_x, the result will be gained from the angle of X'.

(1) r_x is an even number

$$N = r_y r_x HWq\left(6a - \frac{9}{2}\right). \tag{32}$$

(2) r_x is an odd number

$$N = r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_y HWq\left(6a + \frac{15}{2}\right). \tag{33}$$

Equations (30) and (32) are identical, and they are smaller than Eqs. (31) and (33). Hence, if one of r_y and r_x are even numbers,

$$N = r_y r_x HWq\left(6a - \frac{9}{2}\right).$$

If both r_y and r_x are odd numbers, from Eqs. (31) and (33),

$$N = \begin{cases} r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_x HWq\left(6a + \frac{15}{2}\right), & \text{if } r_y \geq r_x \\ r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_y HWq\left(6a + \frac{15}{2}\right), & \text{if } r_y < r_x \end{cases}$$

□

Figure 11 is used as an example to explain Theorem 1 further. In this example, $r_y = 5 > r_x = 3$. So the circuit is simplified from the angle of Y'. In order to observe the process of simplification more clearly, the location of Y' and X' is exchanged and all the 9-CNOT gates are reordered according to the control value on X' and Y' from small to large (see Fig. 16a). Figure 16b gives the simplified circuit. The network complexity of the simplified circuit is

$$r_y r_x HWq\left(6a - \frac{9}{2}\right) + r_x HWq\left(6a + \frac{15}{2}\right) = 1638.$$

In fact, Theorem 1 only gives the upper limit of the complexity because maybe the simplified circuit can be simplified further. For example, the circuit shown in Fig. 16b can be simplified further to Fig. 16c whose network complexity is

$$2 \times ((12 \times 6 - 9) + (12 \times 7 - 9) + (12 \times 8 - 9) + (12 \times 9 - 9)) = 648.$$

From Fig. 16a–c, the network complexity is dropped from 2970 to 1638 to 648 by two steps. Although the principle of the two steps are the same, the first step is easy to describe (has been described in Theorem 1) and the second step has too many cases and cannot be provided quantitative equation easily. Hence, we will study the second step in future.

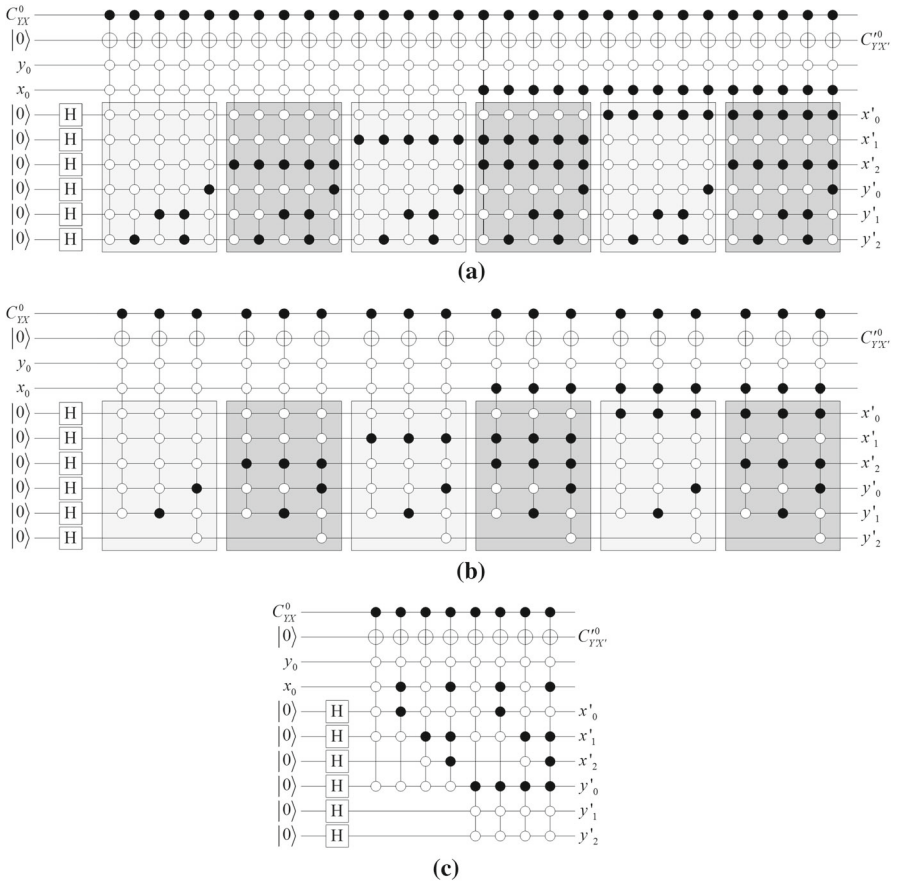


Fig. 16 The simplification of the scaling up circuit. **a** The reordered circuit. **b** The simplified circuit. **c** The simplified further circuit

5 Conclusion

This paper gives an improved quantum image representation GQIR and proposes a quantum algorithm to scale up quantum images based on nearest-neighbor interpolation with integer magnification. It has two main contributions:

(1) A new representation method—the generalized quantum image representation (GQIR)—is proposed to improve NEQR. It inherits all the merits of NEQR and can represent quantum images with arbitrary size $H \times W$.
(2) The quantum image scaling up algorithm based on nearest neighbor interpolation with integer scaling ratio is studied. Image scaling is a widely used geometric transformation. This paper proposes the quantum scheme with integer scaling ratio for the first time. The scheme is to generate a new image as the scaled one instead of modifying the original image into a scaled image. The basic idea of it is

repeating every pixel's color information $r_y \times r_x$ times to the new image by using multi-control-qubit-CNOT gates. The circuit's simplification is also discussed.

The future works may include: (1) give quantum scaling down circuits; (2) improve scaling ratio from integers to real numbers; (3) give quantum scaling circuits based on other interpolation methods, such as bilinear and bicubic; and (4) provide quantitative analysis about the complete simplification.

References

1. Feynman, R.P.: Simulating physics with computers. Int. J. Theor. Phys. **21**(6/7), 467–488 (1982)
2. Vlatko, V., Adriano, B., Artur, E.: Quantum networks for elementary arithmetic operations. Phys. Rev. A **54**(1), 147–153 (1996)
3. 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)
4. Venegas-Andraca, S.E., Ball, J.L.: Processing images in entangled quantum systems. Quantum Inf. Process. **9**(1), 1–11 (2010)
5. Latorre, J.I.: Image compression and entanglement. arXiv:quant-ph/0510031 (2005)
6. Le, P.Q., Dong, F.Y., Hirota, K.: A flexible representation of quantum images for polynomial preparation, image compression and processing operations. Quantum Inf. Process. **10**(1), 63–84 (2011)
7. 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)
8. Sun, B., Iliyasu, A.M., Yan, F., Dong, F.Y., Hirota, K.: An RGB multi-channel representation for images on quantum computers. J. Adv. Comput. Intell. Intell. Inform. **17**(3), 404–417 (2013)
9. 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)
10. Jiang, N., Wang, L.: Quantum image scaling using nearest neighbor interpolation. Quantum Inf. Process. **14**(5), 1559–1571 (2015)
11. Zhang, Y., Lu, K., Gao, Y.H., Xu, K.: A novel quantum representation for log-polar images. Quantum Inf. Process. **12**(9), 3103–3126 (2013)
12. Li, H.S., Zhu, Q.X., Lan, S., Shen, C.Y., Zhou, R.G., Mo, J.: Image storage, retrieval, compression and segmentation in a quantum system. Quantum Inf. Process. **12**(6), 2269–2290 (2013)
13. Li, H.S., Zhu, Q.X., Zhou, R.G., Lan, S., Yang, X.J.: Multi-dimensional color image storage and retrieval for a normal arbitrary quantum superposition state. Quantum Inf. Process. **13**(4), 991–1011 (2014)
14. Wang, J., Jiang, N., Wang, L.: Quantum image translation. Quantum Inf. Process. **14**(5), 1589–1604 (2015)
15. Gonzalez, R., Woods, R.: Digital Image Processing, 3rd edn. Prentice Hall, New Jersey (2007)
16. http://www.mathworks.cn/cn/help/images/ref/imresize.html (2014)
17. Jiang, N., Wu, W.Y., Wang, L.: The quantum realization of Arnold and Fibonacci image scrambling. Quantum Inf. Process. **13**(5), 1223–1236 (2014)
18. Jiang, N., Wang, L.: Analysis and improvement of the quantum Arnold image scrambling. Quantum Inf. Process. **13**(7), 1545–1551 (2014)
19. Jiang, N., Wang, L., Wu, W.Y.: Quantum Hilbert image scrambling. Int. J. Theor. Phys. **53**(7), 2463–2484 (2014)
20. Zhang, Y., Lu, K., Xu, K., Gao, Y.H., Wilson, R.: Local feature point extraction for quantum images. Quantum Inf. Process. **14**(5), 1573–1588 (2015)
21. 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)
22. 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)
23. Zhang, W.W., Gao, F., Liu, B., et al.: A quantum watermark protocol. Int. J. Theory Phys. **52**, 504–513 (2013)

24. Yang, Y.G., Jia, X., Xu, P., Tian, J.: Analysis and improvement of the watermark strategy for quantum images based on quantum Fourier transform. Quantum Inf. Process. **12**(8), 2765–2769 (2013)
25. Song, X.H., Wang, S., Liu, S., El-Latif, Ahmed A.Abd, Niu, X.M.: A dynamic watermarking scheme for quantum images using quantum wavelet transform. Quantum Inf. Process. **12**(12), 3689–3706 (2013)
26. Song, X.H., Wang, S., Liu, S., El-Latif, Ahmed. A. Abd., Niu, X.M.: Dynamic watermarking scheme for quantum images based on Hadamard transform. Multimed. Syst. **20**(4), 379–388 (2014)
27. Jiang, N., Wang, L.: A quantum image information hiding algorithm based on Moiré pattern. Int. J. Theor. Phys. **54**(3), 1021–1032 (2015)
28. Wang, S., Song, X.H., Niu, X.M.: A novel encryption algorithm for quantum images based on quantum wavelet transform and diffusion. Adv. Intell. Syst. Comput. **298**, 243–250 (2014)
29. Hua, T., Chen, J., Pei, D., et al.: Quantum image encryption algorithm based on image correlation decomposition. Int. J. Theor. Phys. **54**(2), 526–537 (2015)
30. Zhou, Ri-Gui, Wu, Qian, Zhang, Man-Qun, et al.: A quantum image encryption algorithm based on quantum image geometric transformations. Pattern Recognit. **321**, 480–487 (2012)
31. Zhou, Ri-Gui, Wu, Qian, Zhang, Man-Qun, et al.: Quantum image encryption and decryption algorithms based on quantum image geometric transformations. Int. J. Theor. Phys. **52**, 1802–1817 (2013)
32. Wootters, W.K., Zurek, W.H.: A single quantum cannot be cloned. Nature **299**, 802–803 (1982)
33. Yuen, H.P.: Amplification of quantum states and noiseless photon amplifiers. Phys. Lett. A **113**(8), 405–407 (1986)
34. Nielson, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
35. Iliyasu, A.M., Le, P.Q., Dong, F.Y., Hirota, K.: A framework for representing and producing movies on quantum computers. Int. J. Quantum Inf. **9**(6), 1459–1497 (2011)