

Integer multiple quantum image scaling based on NEQR and bicubic interpolation

Shuo Cai(蔡硕), Ri-Gui Zhou(周日贵), Jia Luo(罗佳), and Si-Zhe Chen(陈思哲)

Citation: Chin. Phys. B, 2024, 33 (**4**): 040302. DOI: 10.1088/1674-1056/ad1b40
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Gray code based gradient-free optimization algorithm for parameterized quantum circuit

Anqi Zhang(张安琪), Chunhui Wu(武春辉), and Shengmei Zhao(赵生妹)
Chin. Phys. B, 2024, 33 (**2**): 020311. DOI: 10.1088/1674-1056/ad09cd

Quantum algorithm for minimum dominating set problem with circuit design

Haoying Zhang(张皓颖), Shaoxuan Wang(王绍轩), Xinjian Liu(刘新建), Yingtong Shen(沈颖童), and Yukun Wang(王玉坤)
Chin. Phys. B, 2024, 33 (**2**): 020310. DOI: 10.1088/1674-1056/ad02e5

Proposal for sequential Stern-Gerlach experiment with programmable quantum processors

Meng-Jun Hu(胡孟军), Haixing Miao(缪海兴), and Yong-Sheng Zhang(张永生)
Chin. Phys. B, 2024, 33 (**2**): 020303. DOI: 10.1088/1674-1056/ad09ca

Quantum color image scaling based on bilinear interpolation

Chao Gao(高超), Ri-Gui Zhou(周日贵), and Xin Li(李鑫)
Chin. Phys. B, 2023, 32 (**5**): 050303. DOI: 10.1088/1674-1056/ac8f35

Novel quantum secret image sharing scheme

Gao-Feng Luo(罗高峰), Ri-Gui Zhou(周日贵), Wen-Wen Hu(胡文文)
Chin. Phys. B, 2019, 28 (**4**): 040302. DOI: 10.1088/1674-1056/28/4/040302

Integer multiple quantum image scaling based on NEQR and bicubic interpolation

Shuo Cai(蔡硕)[1,2], Ri-Gui Zhou(周日贵)[1,2,†], Jia Luo(罗佳)[1,2,4,‡], and Si-Zhe Chen(陈思哲)[3]

[1] *College of Information Engineering, Shanghai Maritime University, Shanghai* 201306, *China*
[2] *Research Center of Intelligent Information Processing and Quantum Intelligent Computing, Shanghai* 201306, *China*
[3] *College of Merchant Marine, Shanghai Maritime University, Shanghai* 201306, *China*
[4] *School of Mathematics and Computational Science, Shangrao Normal University, Shangrao* 334001, *China*

(Received 27 June 2023; revised manuscript received 10 December 2023; accepted manuscript online 5 January 2024)

As a branch of quantum image processing, quantum image scaling has been widely studied. However, most of the existing quantum image scaling algorithms are based on nearest-neighbor interpolation and bilinear interpolation, the quantum version of bicubic interpolation has not yet been studied. In this work, we present the first quantum image scaling scheme for bicubic interpolation based on the novel enhanced quantum representation (NEQR). Our scheme can realize synchronous enlargement and reduction of the image with the size of $2^n \times 2^n$ by integral multiple. Firstly, the image is represented by NEQR and the original image coordinates are obtained through multiple CNOT modules. Then, 16 neighborhood pixels are obtained by quantum operation circuits, and the corresponding weights of these pixels are calculated by quantum arithmetic modules. Finally, a quantum matrix operation, instead of a classical convolution operation, is used to realize the sum of convolution of these pixels. Through simulation experiments and complexity analysis, we demonstrate that our scheme achieves exponential speedup over the classical bicubic interpolation algorithm, and has better effect than the quantum version of bilinear interpolation.

Keywords: quantum image processing, image scaling, bicubic interpolation, quantum circuit

PACS: 03.67.–a, 03.67.Ac, 07.05.Pj, 03.67.Lx **DOI:** 10.1088/1674-1056/ad1b40

1. Introduction

Quantum computing[1] is a novel computing paradigm proposed by Feynman in 1982, which utilizes the principles of quantum mechanics and exploits quantum superposition, entanglement, and coherence to achieve quantum parallel computing. Quantum computing has shown remarkable advantages over classical computing in some areas. In 1994, Shor proposed a quantum integer factorization algorithm[2] that can crack the widely used RSA encryption scheme in polynomial time. In 1996, Grover proposed a quantum search algorithm[3] that brings a second acceleration to unstructured data search. It can be seen that the powerful parallel computing power of quantum computing far exceeds that of classical computers.

Quantum image processing (QImP) is an emerging discipline that combines mathematics, computer science, and graphics to address two issues: quantum image representation (QImR) and quantum image processing algorithms. Storing classical computing images in a quantum computer is known as quantum image representation. In 2003, Venegas-Andraca and Bose[4] proposed the qubit lattice representation method, which is similar to classical methods in that images are stored as matrices and each qubit stores only one pixel. This is the first attempt at quantum image characterization. Then, The real ket representation method using quantum superposition states for storage was proposed by Latorre[5] in 2005. In 2011, Le *et al.*[6] proposed a flexible quantum image representation (FRQI), which stores color information as angles and coordinate information in an n-qubit sequence. In 2013, Zhang *et al.*[7] proposed a new enhanced quantum representation (NEQR), it stores grayscale information in qubits as binary strings. Although more qubits are used, NEQR solves the problem of FRQI's inability to accurately measure grayscale information within a finite number of times and is convenient for color operation. However, NEQR can only store square images of size $2^n \times 2^n$, and to address this issue, Jiang and Wang[8] proposed an improved new enhanced quantum representation (INEQR) in 2015, which can store non-square image of size $2^{n_1} \times 2^{n_2}$. In the same year, Jiang *et al.*[9] further proposed a generalized quantum image representation (GQIR) that can represent images of arbitrary size.

On the basis of the quantum image representation, many quantum image processing algorithms have been proposed, such as geometric transformation,[10] image scaling,[8,9,11–18] edge detection,[19] image filtering,[20] image watermarking,[21] feature extraction,[22] and image segmentation.[23] Among them, image scaling is one of the most commonly used and basic operations in image processing, which can adjust the size of digital image. Interpolation methods are used to create new pixels when scaling up images or to delete redundant pixels when scaling down images. Com-

[†]Corresponding author. E-mail: rgzhou@shmtu.edu.cn
[‡]Corresponding author. E-mail: luojia@shmtu.edu.cn

© 2024 Chinese Physical Society and IOP Publishing Ltd

mon interpolation methods include nearest-neighbor interpolation, bilinear interpolation and bicubic interpolation.[24] So far, nearest-neighbor interpolation and bilinear interpolation are mainly used to achieve quantum image scaling. In 2015, Jiang and Wang[8] proposed an integer ratio scaling circuit based on INEQR and nearest-neighbor interpolation, which is the first quantum image processing algorithm that can change the size of an image. In the same year, they also proposed an integer ratio scaling circuit based on GQIR.[9] In 2017, Zhou *et al.*[12] improved FRQI and proposed a nearest-neighbor interpolation circuit for color images, and Zhou *et al.* first implemented bilinear interpolation based on NEQR.[13] Therefore, quantum image scaling algorithms based on bilinear interpolation have been widely used[14–18] in recent years. However, no quantum image scaling algorithm based on bicubic interpolation has been proposed yet. In this work, a bicubic interpolation algorithm is proposed based on NEQR for the first time to fill the gap in its quantum version. This algorithm enables integer multiple scaling of square quantum images with superior visual fidelity over quantum bilinear interpolation, and achieves an exponential advantage over classical cubic convolution interpolation in terms of computational complexity.

The rest of this paper is organized as follows. In Section 2, the theoretical basis is introduced. In Section 3 the relevant quantum modules are presented, and the complete image scaling circuit design using bicubic interpolation is given in Section 4. In Section 5 the complexity of the entire circuit is analyzed and comparisons are made. In Section 6 the simulation experiment is carried out and the experimental results are analyzed. Finally, some conclusions are drawn from the present study and future work is also mentioned in Section 7.

2. Theoretical basis

This section mainly introduces the theoretical background of our work, including the quantum image representation method, the bicubic interpolation algorithm used, and the adaptations for the quantum version.

2.1. Novel enhanced quantum representation (NEQR)

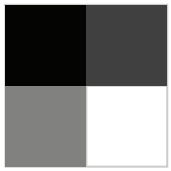
In 2013, Zhang *et al.*[7] proposed the NEOR for an image of size $2^n \times 2^n$, with a grayscale range of 2^q. The NEQR is described as follows:

$$|I\rangle = \frac{1}{2^n}\sum_{Y=0}^{2^n-1}\sum_{X=0}^{2^n-1}|f(Y,X)\rangle|YX\rangle = \frac{1}{2^n}\sum_{Y=0}^{2^n-1}\sum_{X=0}^{2^n-1}\otimes_{i=0}^{q-1}|c_{YX}^i\rangle|YX\rangle, \tag{1}$$

where

$$f(Y,X) \in \{0,1,\ldots,2^q-1\};$$
$$|c_{YX}\rangle = |c_{YX}^0 c_{YX}^1 \ldots c_{YX}^{q-1}\rangle,\ \ c_{YX}^i \in \{0,1\}, \tag{2}$$
$$|YX\rangle = |Y\rangle|X\rangle = |y_0 y_1 \ldots y_{n-1}\rangle|x_0 x_1 \ldots x_{n-1}\rangle;$$
$$y_i, x_i \in \{0,1\}. \tag{3}$$

The $|f(Y,X)\rangle$ is the grayscale information, with q qubits, and $|Y,X\rangle$ is the position information, with $2n$ qubits. The NEQR requires $q+2n$ qubits to represent all the image information. Figure 1 shows a grayscale image of size 2×2 with a pixel grayscale range of 255 and its corresponding quantum expression. It requires 8 qubits to represent the grayscale information and 2 qubits to represent the position information.



$$|I\rangle = \frac{1}{2}(|0\rangle\otimes|00\rangle+|64\rangle\otimes|10\rangle+|128\rangle\otimes|01\rangle+|255\rangle\otimes|11\rangle)$$
$$= \frac{1}{2}\Big(\begin{matrix}(|00000000\rangle\otimes|00\rangle+|01000000\rangle\otimes|10\rangle\\ +(|10000000\rangle\otimes|01\rangle+|11111111\rangle\otimes|11\rangle\end{matrix}\Big)$$

Fig. 1. An example of 2×2 image and its NEQR.

2.2. Cubic convolution interpolation

In classical image processing, bicubic interpolation is one of the most commonly used interpolation methods. Comparing with nearest-neighbor interpolation and bilinear interpolation, the image effect obtained by bicubic interpolation is good, but the time complexity of the algorithm also increases correspondingly. The bicubic interpolation method used in this work is based on the cubic convolution interpolation in Ref. [24], which is the most commonly used image scaling algorithm.

In cubic convolution interpolation, the interpolation in the vertical dimension and in the horizontal dimension can be separated. The interpolation function can be decomposed as follows:

$$I' = S(I,r_x,r_y) = S_y(S_x(I,r_x),r_y) = S_x(S_y(I,r_y),r_x), \tag{4}$$

where I' is the interpolated image, S is the scaling function, I is the original image, and r_y and r_x are the vertical scaling ratio and horizontal scaling ratio, respectively. The function S can be decomposed into S_y and S_x, which are the vertical scaling function and horizontal scaling function. The principles of S_y and S_x are exactly the same and can be interchanged. Therefore, no matter which dimension the interpolation function is performed in first, the final interpolated image result is exactly the same.

In the interpolation process of image scaling, for an original image I of size $H\times W$ and an interpolated image I' of size $H'\times W'$, the correspondence between the coordinates of the interpolated image (Y',X') and the coordinates of the original image (Y,X) is shown in Eqs. (5) and Fig. 2. By mapping to the original image coordinates, the corresponding 16 neighborhood pixel values can be obtained as shown in Eq. (7).

$$X = \left\lfloor X'\times\frac{W}{W'}\right\rfloor,\quad Y = \left\lfloor Y'\times\frac{H}{H'}\right\rfloor, \tag{5}$$
$$w = \frac{W}{W'}X' - X,\quad h = \frac{H}{H'}Y' - Y, \tag{6}$$

$$\boldsymbol{B}=\begin{bmatrix} f(Y-1,X-1) & f(Y-1,X) & f(Y-1,X+1) & f(Y-1,X+2) \\ f(Y,X-1) & f(Y,X) & f(Y,X+1) & f(Y,X+2) \\ f(Y+1,X-1) & f(Y+1,X) & f(Y+1,X+1) & f(Y+1,X+2) \\ f(Y+2,X-1) & f(Y+2,X) & f(Y+2,X+1) & f(Y+2,X+2) \end{bmatrix}. \tag{7}$$

We define matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ as the vertical scaling function matrix and horizontal scaling function matrix respectively, as shown in below:

$$\boldsymbol{A}=[S(1+h),S(h),S(1-h),S(2-h)], \tag{8}$$
$$\boldsymbol{C}=[S(1+w),S(w),S(1-w),S(2-w)]^{\mathrm{T}}, \tag{9}$$

The pixel value in (Y',X') of the interpolated image can be expressed as follows:

$$f'(Y',X')=\boldsymbol{A}\cdot\boldsymbol{B}\cdot\boldsymbol{C}=\sum_{i=-1}^{2}\sum_{j=-1}^{2}(Y+i,X+j)\times S(|i-h|)S(|j-w|). \tag{10}$$

According to Ref. [24], the convolution kernel of the cubic convolution function is shown below:

$$S(x)=\begin{cases} 1-(a+3)x^2+(a+2)|x|^3, & 0\le|x|\le 1, \\ -4a+8a|x|-5ax^2+a|x|^3, & 1<|x|\le 2, \\ 0, & 2<|x|. \end{cases} \tag{11}$$

In image processing, a is often taken as -0.5. At this time, the convolution kernel of the cubic convolution function can be transformed into Eq. (12), and the corresponding function graph is shown in Fig. 2. The corresponding weight matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ in the y direction and x direction can be obtained through this convolution kernel formula. Substituting $\boldsymbol{A}$ and $\boldsymbol{C}$ into Eq. (10), the pixel values of the interpolated image can be obtained as follows:

$$S(x)=\begin{cases} 1-\frac{5}{2}x^2+\frac{3}{2}|x|^3, & 0\le|x|\le 1, \\ 2-4|x|+\frac{5}{2}x^2-\frac{1}{2}|x|^3, & 1<|x|\le 2, \\ 0, & 2<|x|. \end{cases} \tag{12}$$

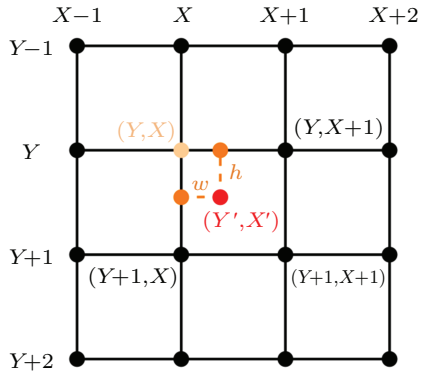


Fig. 2. Mapping relationship and neighborhood pixels of coordinates in two images.

2.3. Preparation of quantum image processing algorithm based on NEQR

Since the quantum image representation used in this paper is NEQR, which can only represent square images, and most quantum reversible circuits currently only support positive integer operations. In order to facilitate the implementation of the quantum bicubic interpolation, it is necessary to transform the above formula in Subsection 2.2 so that the quantum image scaling circuit can be implemented using quantum reversible circuits.

For an original image I of size $2^n\times 2^n$, the size of the interpolated image is $2^{n+m}\times 2^{n+m}$. Obviously, the scaling ratio is 2^m. The coordinate mapping relation between the interpolated image and the original image can be derived from Eqs. (5) and (6) as follows:

$$X=\left\lfloor\frac{X'}{2^m}\right\rfloor,\quad Y=\left\lfloor\frac{Y'}{2^m}\right\rfloor, \tag{13}$$
$$w=\frac{X'}{2^m}-X,\quad h=\frac{Y'}{2^m}-Y. \tag{14}$$

Next, we consider the negative numbers that will appear in the calculation process, and transform the formula to make the final result the same. The pixel values in matrix $\boldsymbol{B}$ must be positive integers. According to Fig. 3, if $x\in(1,2)$, then $S(x)<0$, that is, the values of $\boldsymbol{A}[0,0]$, $\boldsymbol{A}[0,3]$, $\boldsymbol{C}[0,0]$, and $\boldsymbol{C}[3,0]$ are negative and the values of other elements are positive. Hence, to avoid multiplication operations between positive numbers and negative numbers in Eq. (10), we simplify the operations into the addition and subtraction of the outcomes of four matrix multiplications as indicated in Eq. (15).

$$f'(Y',X')=\boldsymbol{A}_1\cdot\boldsymbol{B}_1\cdot\boldsymbol{C}_1+\boldsymbol{A}_2\cdot\boldsymbol{B}_4\cdot\boldsymbol{C}_2-\boldsymbol{A}_1\cdot\boldsymbol{B}_2\cdot\boldsymbol{C}_2-\boldsymbol{A}_2\cdot\boldsymbol{B}_3\cdot\boldsymbol{C}_1, \tag{15}$$

where

$$\boldsymbol{A}_1=[S(h),S(1-h)], \tag{16}$$
$$\boldsymbol{A}_2=[|S(1+h)|,|S(2-h)|], \tag{16}$$
$$\boldsymbol{C}_1=[S(w),S(1-w)]^{\mathrm{T}}, \tag{17}$$
$$\boldsymbol{C}_2=[|S(1+w)|,|S(2-w)|]^{\mathrm{T}}, \tag{17}$$
$$\boldsymbol{B}_1=\begin{bmatrix} f(Y,X) & f(Y,X+1) \\ f(Y+1,X) & f(Y+1,X+1) \end{bmatrix}, \tag{18}$$
$$\boldsymbol{B}_2=\begin{bmatrix} f(Y,X-1) & f(Y,X+2) \\ f(Y+1,X) & f(Y+1,X+2) \end{bmatrix}, \tag{19}$$
$$\boldsymbol{B}_3=\begin{bmatrix} f(Y-1,X) & f(Y-1,X+1) \\ f(Y+2,X) & f(Y+2,X+1) \end{bmatrix}, \tag{20}$$
$$\boldsymbol{B}_4=\begin{bmatrix} f(Y-1,X-1) & f(Y-1,X+2) \\ f(Y+2,X) & f(Y+2,X+2) \end{bmatrix}. \tag{21}$$

Equations (15)–(21) ensure that there is no negative number operation in the entire calculation process. The formula is then simplified to eliminate any decimals in the operation.

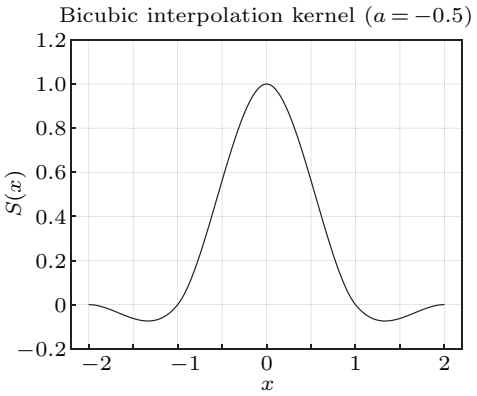


Fig. 3. Schematic diagram of cubic convolution kernel function.

Only matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ need to be analyzed. Since matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are the same except for parameters, we only analyze the changes in $\boldsymbol{A}$ as shown in Eq. (22) and matrix $\boldsymbol{C}$ is similar.

$$\begin{aligned}\boldsymbol{A} &= [S_2(1+h), S_1(h), S_1(1-h), S_2(2-h)]\\ &= \Big[2-4(1+h)+\frac{5}{2}(1+h)^2-\frac{1}{2}(1+h)^3,\\ &\quad 1-\frac{5}{2}h^2+\frac{3}{2}h^3, 1-\frac{5}{2}(1-h)^2+\frac{3}{2}(1-h)^3,\\ &\quad 2-4(2-h)+\frac{5}{2}(2-h)^2-\frac{1}{2}(2-h)^3\Big]\\ &= \Big[-\frac{1}{2}h(h-1)^2, \frac{1}{2}(h-1)(3h^2-2h-2),\\ &\quad -\frac{1}{2}h(3h^2-4h-1), \frac{1}{2}h^2(h-1)\Big]. \qquad (22)\end{aligned}$$

By expanding w and h 2^m times, we obtain y and x respectively:

$$\begin{aligned}&x = 2^m w = X' - 2^m X, \quad y = 2^m w = Y' - 2^m Y,\\ &y, x \in [0, 2^m). \qquad (23)\end{aligned}$$

In the matrix, the values of h and w may not be integral, but y and x must be integral. Equations (10) and (15) are then processed as follows:

$$\begin{aligned}f'(Y', X') &= \boldsymbol{A}\cdot\boldsymbol{B}\cdot\boldsymbol{C}\\ &= \frac{1}{2^{6m}}(2\times 2^{3m}\boldsymbol{A})\cdot\boldsymbol{B}\cdot(2\times 2^{3m}\boldsymbol{C})\\ &= \frac{1}{2^{6m+2}}\{2^{3m+1}\boldsymbol{A}_1\cdot\boldsymbol{B}_1\cdot 2^{3m+1}\boldsymbol{C}_1\\ &\quad + 2^{3m+1}\boldsymbol{A}_2\cdot\boldsymbol{B}_4\cdot 2^{3m+1}\boldsymbol{C}_2\\ &\quad - 2^{3m+1}\boldsymbol{A}_1\cdot\boldsymbol{B}_2\cdot 2^{3m+1}\boldsymbol{C}_2\\ &\quad - 2^{3m+1}\boldsymbol{A}_2\cdot\boldsymbol{B}_3\cdot 2^{3m+1}\boldsymbol{C}_1\}. \qquad (24)\end{aligned}$$

Let

$$\begin{aligned}\boldsymbol{A}'_1 &= 2^{3m+1}\boldsymbol{A}_1 = [(2^m-y)(2^{2m+1}+2^{m+1}y-3y^2),\\ &\quad y(2^{2m}+2^{m+2}y-3y^2)], \qquad (25)\\ \boldsymbol{C}'_1 &= 2^{3m+1}\boldsymbol{C}_1 = [(2^m-x)(2^{2m+1}+2^{m+1}x-3x^2),\\ &\quad x(2^{2m}+2^{m+2}x-3x^2)]^{\mathrm{T}}, \qquad (26)\\ \boldsymbol{A}'_2 &= |2^{3m+1}A_2| = [y(2^m-y)^2, y^2(2^m-y)], \qquad (27)\\ \boldsymbol{C}'_2 &= |2^{3m+1}C_2| = [x(2^m-x)^2, x^2(2^m-x)]^{\mathrm{T}}. \qquad (28)\end{aligned}$$

According to Eqs. (25)–(28), the final result can be expressed as Eq. (29). Now, it can be ensured that all operations involve only positive integers.

$$\begin{aligned}f'(Y', X') &= \frac{1}{2^{6m+2}}(f'_1(Y', X') + f'_3(Y', X')\\ &\quad - f'_2(Y', X') - f'_4(Y', X'))\\ &= \frac{1}{2^{6m+2}}(\boldsymbol{A}'_1\cdot\boldsymbol{B}_1\cdot\boldsymbol{C}'_1 + \boldsymbol{A}'_2\cdot\boldsymbol{B}_4\cdot\boldsymbol{C}'_2\\ &\quad - \boldsymbol{A}'_1\cdot\boldsymbol{B}_2\cdot\boldsymbol{C}'_2 - \boldsymbol{A}'_2\cdot\boldsymbol{B}_3\cdot\boldsymbol{C}'_1). \qquad (29)\end{aligned}$$

3. Quantum modules

3.1. Multiple CNOT module

The multiple CNOT module[13] replicates n-qubit number $|X\rangle = |x_{n-1}\ldots x_1 x_0\rangle$ to $|0\rangle^{\otimes n}$ by using n CNOT gates. Its quantum circuit is shown in Fig. 4.

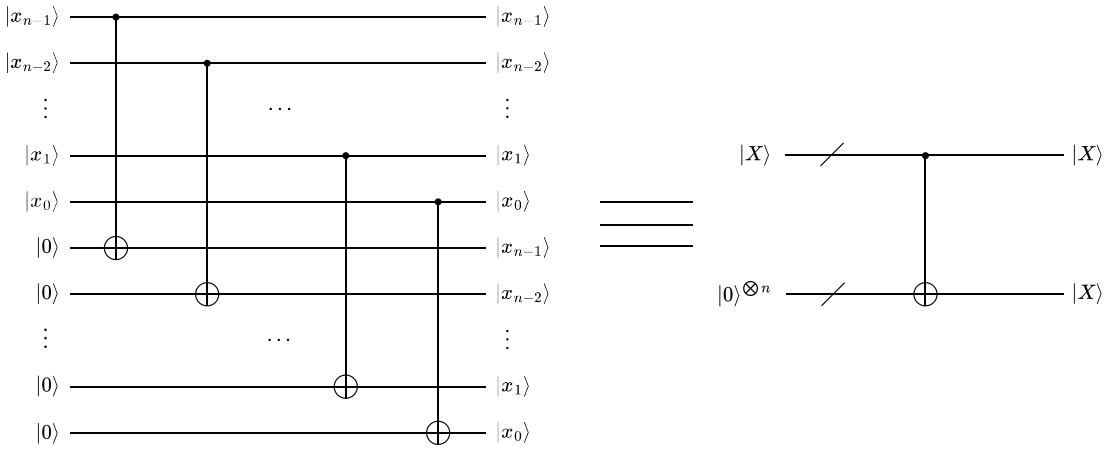


Fig. 4. Circuit of multiple CNOT module.

3.2. Quantum operation $\Omega_{Y,X}$

The function of the quantum operation[7] is to set the pixel value at position (Y,X) in the quantum image. The quantum circuit is shown in Fig. 5. For more information about quantum operation $\Omega_{Y,X}$, please refer to Ref. [7].

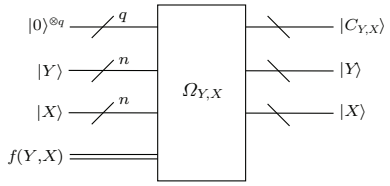


Fig. 5. Circuit of quantum operation.

3.3. Adding one module U_1 and subtracting one module U_2

The function of the adding one module[25] (U_1) is to add one to $x_{n-1}x_{n-2}\ldots x_1x_0$ only when $x_{n-1}x_{n-2}\ldots x_1x_0 \neq 1$, otherwise it does not change.

The function of the subtracting one module (U_2) is to subtract one from $x_{n-1}x_{n-2}\ldots x_1x_0$ only when $x_{n-1}x_{n-2}\ldots x_1x_0 \neq 0$, otherwise it does not change.

U_1 and U_2 are shown in Eqs. (30), (31), and Fig. 6.

$$U_1|x_{n-1}\ldots x_1x_0\rangle = \begin{cases} |x_{n-1}\ldots x_1x_0+1\rangle, & x_{n-1}\times\cdots\times x_1\times x_0 \neq 1, \\ |x_{n-1}\ldots x_1x_0\rangle, & x_{n-1}\times\cdots\times x_1\times x_0 = 1, \end{cases} \tag{30}$$

$$U_2|x_{n-1}\ldots x_1x_0\rangle = \begin{cases} |x_{n-1}\ldots x_1x_0-1\rangle, & x_{n-1}\times\cdots\times x_1\times x_0 \neq 0, \\ |x_{n-1}\ldots x_1x_0\rangle, & x_{n-1}\times\cdots\times x_1\times x_0 = 0, \end{cases} \tag{31}$$

where n is an integer greater than 1 and $x_0,x_1\ldots x_{n-1}\in\{0,1\}$.

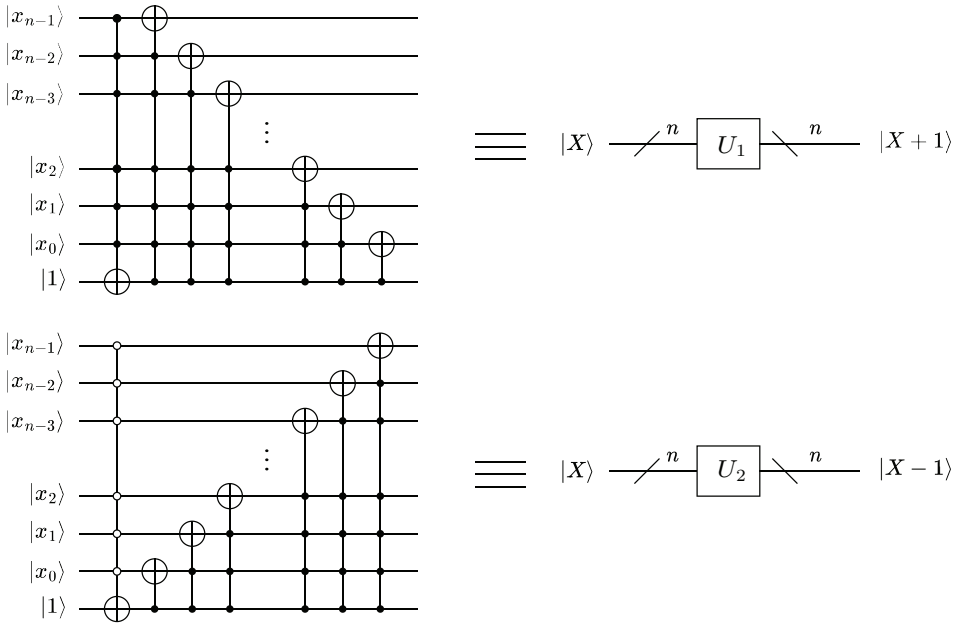


Fig. 6. Circuit of U_1 module and U_2 module.

3.4. Quantum adder module

The addition of n-qubit X and Y is performed by the quantum adder (QA) module.[26] Its quantum circuit is shown in Fig. 7. It consists of quantum Fourier transform (QFT), rotation module, and inverse QFT (IQFT). It requires only one auxiliary qubit for addition, improving computational performance.

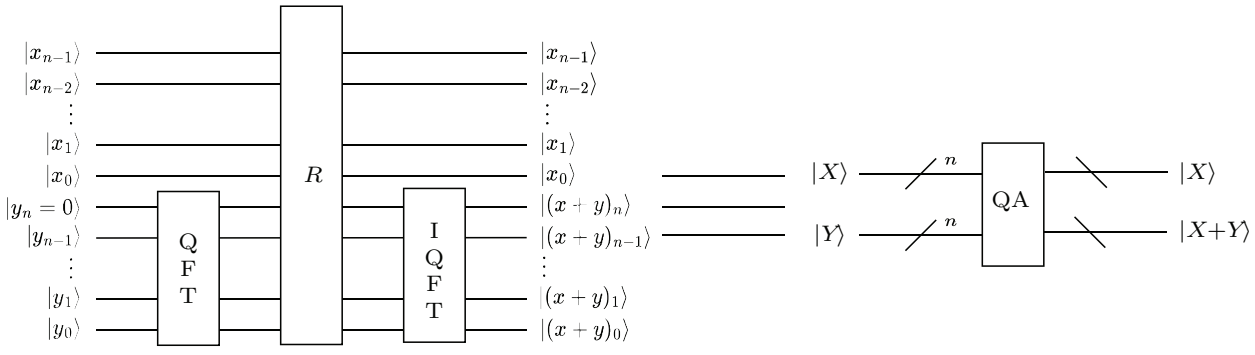


Fig. 7. Circuit of QA module.

3.5. Reversible parallel subtractor module

A reversible parallel subtractor (PS) module is implemented by a reversible half subtractor (RHS) and $n-1$ reversible full subtractors (RFSs). The function of n-qubit X minus n-qubit Y is implemented. Figure 8 shows the detailed circuit of the PS module. For more detailed information, refer to Refs. [27,28], please.

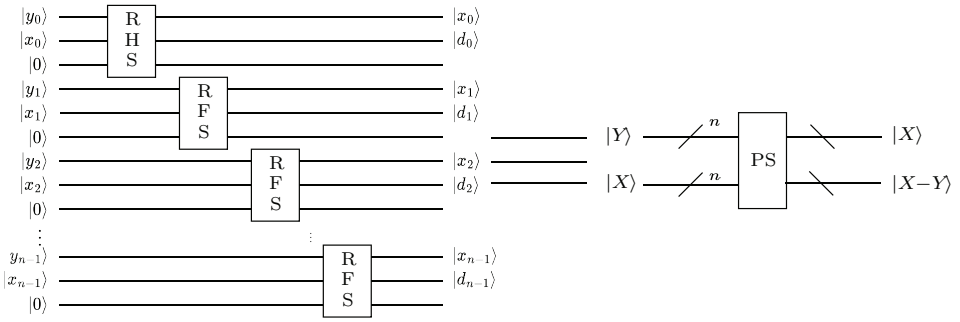


Fig. 8. Circuit of PS module.

3.6. T-count multiplier module

The T-count multiplier (TM) module performs the multiplication of n-qubit X and Y. The inputs of the module are the multiplicand $|X\rangle$ and the multiplier $|Y\rangle$, both of which are n qubits. The output of the module is the product $|P\rangle$ of $|X\rangle$ and $|Y\rangle$ on the right side. The TM module consists of Toffoli gate array and Ctr-Add circuit.

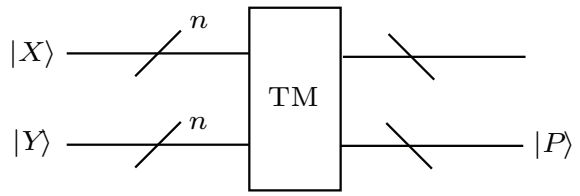


Fig. 9. Simplified circuit of TM module.

Figure 9 shows the simplified circuit of the TM module. For more detailed information, readers can refer to Ref. [29], please.

3.7. Multiply by 2 module and divide by 2 module

The function of the multiply by 2 module[30] is to double the value of a non-negative integer, that is, to implement $|X\rangle \to |2X\rangle$,where $|X\rangle = |x_{n-1}\ldots x_1x_0\rangle$. This module can be implemented by continuous exchange between adjacent qubits. Before performing the operation, an auxiliary qubit needs to be added at the highest position to store the information of the original highest qubit x_{n-1}. The result of the multiply by 2 module transformation is obtained as follows: $|2X\rangle = |x_{n-1}\ldots x_1x_00\rangle$.

The function of the divide by 2 module is to halve the value of a non-negative integer. The change in quantum state before and after the unitary operation is $|x_{n-1}x_{n-2}\ldots x_1x_0\rangle \to |0x_{n-1}x_{n-2}\ldots x_1\rangle$. According to the characteristics of binary, moving each qubit to the right in turn and setting the highest position to 0 can achieve this module.

The circuit of the multiply by 2 module and the divide by 2 module are shown in Fig. 10.

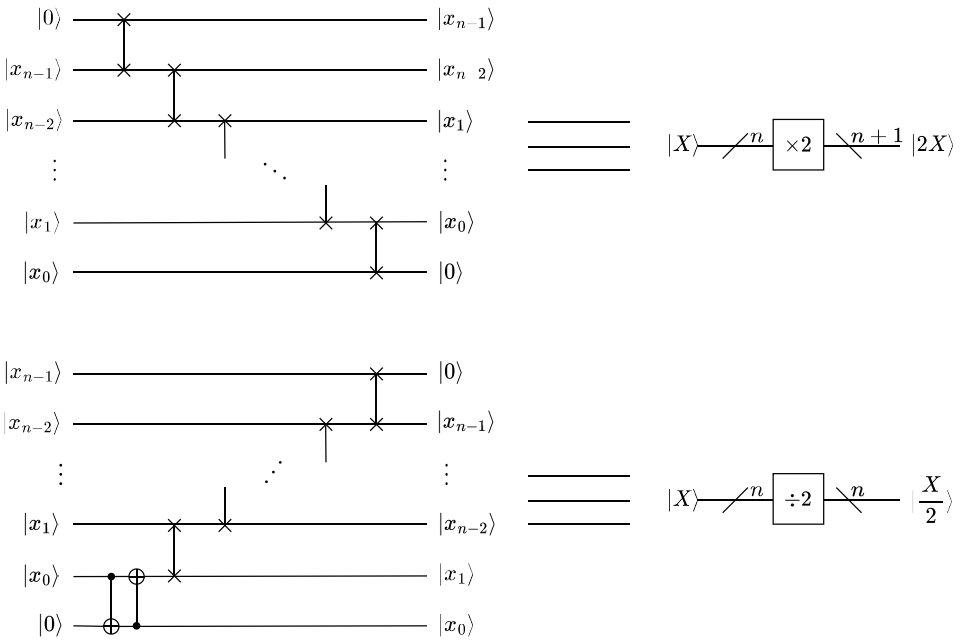


Fig. 10. Multiply by 2 module and divide by 2 module.

To implement multiplication by 2^m or division by 2^m based on the multiply by 2 module and the divide by 2 module, most of the current quantum image processing papers either connect m divide by 2 modules in series or use a quantum divider module directly, but both methods are highly complex. We have modified the multiply by 2 module and divide by 2 module mentioned above to implement multiply by 2^m module function and divide by 2^m module function directly. The circuit is shown in Fig. 11.

In the multiply by 2^m module, to prevent qubit from overflowing, m qubits need to be added before the start of the circuit. Then the exchange gates in the figure are followed in turn to exchange qubits, with an interval being m. To enlarge an n-qubit number, n exchange gates are required. In the exchange process, each state change is

$$|0\rangle^{\otimes m}|x_{n-1}x_{n-2}\ldots x_0\rangle \to |x_{n-1}\rangle|0\rangle^{\otimes m}|x_{n-2}\ldots x_0\rangle \to \cdots$$
$$\to |x_{n-1}x_{n-2}\ldots\rangle|0\rangle^{\otimes m}|x_0\rangle \to |x_{n-1}x_{n-2}\ldots x_0\rangle|0\rangle^{\otimes m}. \quad (32)$$

Similarly, in the divide by 2^m module, the state change is $|x_{n-1}x_{n-2}\ldots x_0\rangle|0^{\otimes m}\rangle \to |0^{\otimes m}\rangle|x_{n-1}x_{n-2}\ldots x_0\rangle$. Due to the reversibility of quantum computing, swapping the input state and output state of the multiply by 2^m module can also achieve the function of divide by 2^m module.

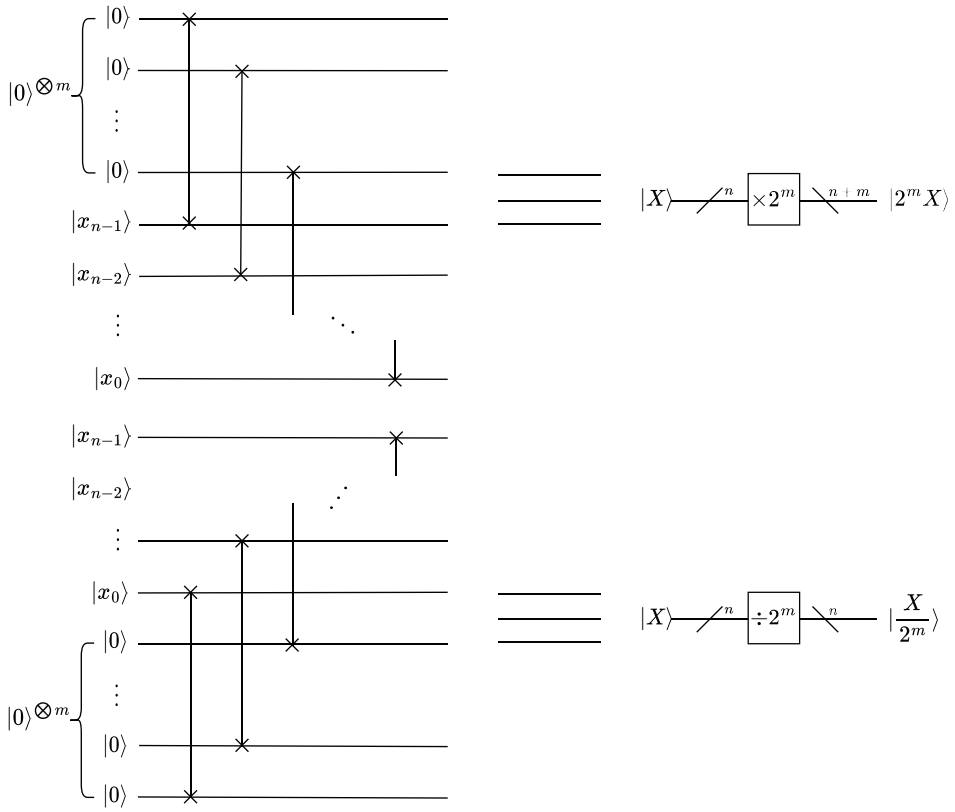


Fig. 11. Multiply by 2^m module and divide by 2^m module.

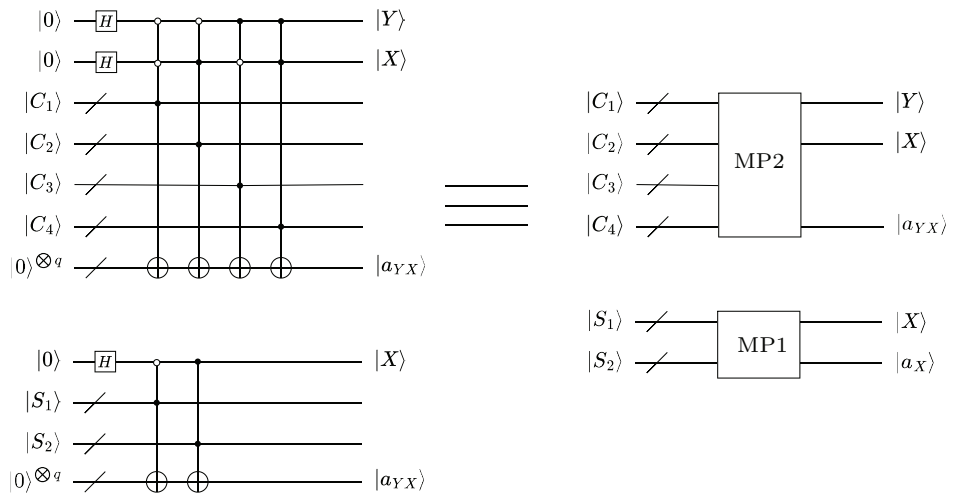


Fig. 12. Matrix preparation circuit.

3.8. Quantum matrix multiplier module

The convolution operations are implemented by replacing the complex arithmetic module with the quantum matrix multiplier in Ref. [31].

3.8.1. Matrix preparation module

Before performing matrix operations, the weights and neighborhood pixel values obtained through the above quantum modules need to be prepared into the matrix. The matrix representation used is as follows:

$$|T\rangle = \frac{1}{\sqrt{2}^{M+N}} \sum_{Y=0}^{M-1} \sum_{X=0}^{N-1} |t_{YX}\rangle |YX\rangle. \tag{33}$$

As can be seen, this expression is similar to the GQIR or NEQR quantum image representation of quantum image representation. A circuit for preparing the quantum matrix can be easily designed based on the preparation of GQIR. Since this work only involves 1×2 matrices, 2×2 matrices and 2×1 matrices. We only employ matrix preparation modules MP1 and MP2 to generate a 1×2 matrix $[S_1, S_2]$ and a 2×2 matrix $[C_1, C_2; C_3, C_4]$ respectively. The circuit is shown in Fig. 12.

3.8.2. Quantum matrix multiplier composition module

The quantum comparator[32] (CMP) is used to compare two binary numbers a and b. The outputs are two 1-qubit values, which indicate the comparison results. The change of the output result state is as follows:

$$|ab00\rangle \to |abc_1c_2\rangle, \tag{34}$$

$$c_1c_2 = \begin{cases} 10, & a > b, \\ 01, & a < b, \\ 00, & a = b. \end{cases} \tag{35}$$

The element-wise multiplication of two matrices is performed by the matching member multiplication module (MS). The results obtained by the MS module are accumulated step by step by the superposition state quantum adder module (ASS). Figure 13 illustrates the simplified circuits for these modules. The quantum matrix multiplier is described in detail in Ref. [32].

4. Quantum circuit realization of image scaling

4.1. Quantum image scaling-up scheme

Now, we suppose that the original image size is $2^n \times 2^n$ and the interpolated image size is $2^{n+m} \times 2^{n+m}$, where $n, m > 0$. According to the content described in Subsection 2.3, the entire circuit can be divided into the following steps.

Step 1 Map the position information (Y', X') of the interpolated image to the position information (Y, X) of the original image

The Y', X', Y, and X can be represented by the following expressions:

$$\begin{aligned} |Y'\rangle &= |Y'_{n+m-1} \ldots Y'_m \ldots Y'_1 Y'_0\rangle, \\ |X'\rangle &= |X'_{n+m-1} \ldots X'_m \ldots X'_1 X'_0\rangle, \\ |Y\rangle &= |Y_{n-1} \ldots Y_1 Y_0\rangle, \\ |X\rangle &= |X_{n-1} \ldots X_1 X_0\rangle. \end{aligned} \tag{36}$$

According to Eq. (13), it can be obtained that the high n qubits of (Y', X') correspond to (Y, X), and n CNOT gates are used to copy the high n qubits of the interpolation coordinates to obtain the mapped original image position information. Then, y and x correspond to the low m qubits, and m CNOT gates are used to copy the low m qubits. The circuit is shown in Fig. 14.

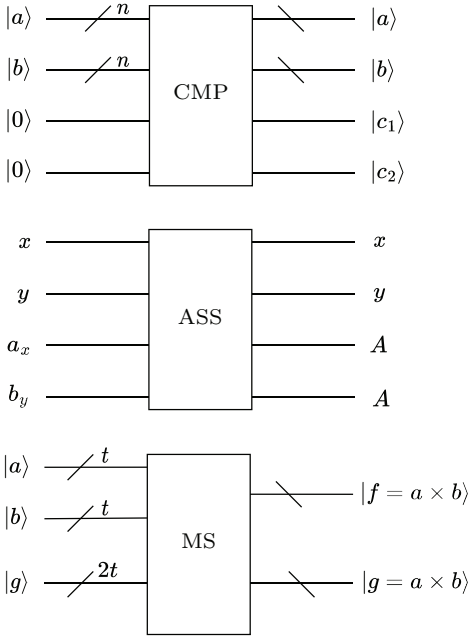


Fig. 13. Quantum matrix multiplier composition module.

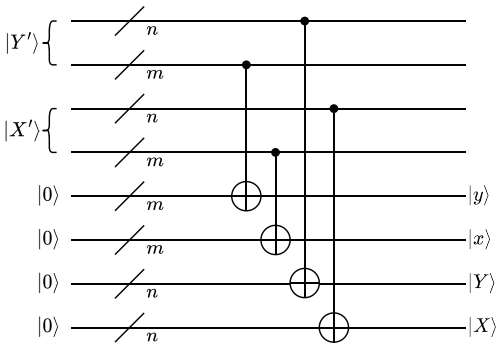


Fig. 14. Circuit of coordinate mapping.

Step 2 Obtain all the weight information in the Y and X directions

To obtain A'_1, A'_2, C'_1, and C'_2 in Eqs. (25)–(28), y and x need to be substituted to obtain the weight information A_y

and $\boldsymbol{C}_x$ in the Y direction and X direction respectively. The elements of the matrices $\boldsymbol{A}_y$ and $\boldsymbol{C}_x$ are given by the following equations:

$$\begin{aligned}\boldsymbol{A}_y = &\left[y(2^m-y)^2,(2^m-y)(2^{2m+1}+2^{m+1}y-3y^2),\right.\\ &\left.y(2^{2m}+2^{m+2}y-3y^2),y^2(2^m-y)\right],\end{aligned} \tag{37}$$

$$\begin{aligned}\boldsymbol{C}_x = &\left[x(2^m-x)^2,(2^m-x)(2^{2m+1}+2^{m+1}x-3x^2),\right.\\ &\left.x(2^{2m}+2^{m+2}x-3x^2),x^2(2^m-x)\right]^{\mathrm{T}}.\end{aligned} \tag{38}$$

The elements in $\boldsymbol{A}'_1$, $\boldsymbol{A}'_2$, $\boldsymbol{C}'_1$, and $\boldsymbol{C}'_2$ are obtained by extracting the corresponding elements from the two matrices given above. The values of the four elements in $\boldsymbol{A}_y$ are shown by taking y for example. The calculation circuit is shown in Fig. 15.

The calculation method of $\boldsymbol{C}_x$ is similar, except that the input parameter changes from y to x. According to the convolution kernel function diagram as shown in Fig. 3, $|S(x)|<=1$. Then, after scaling up $|\boldsymbol{A}|$ and $|\boldsymbol{C}|$ by 2^{3m+1} times, the maximum element value in Eqs. (37) and (38) is 2^{3m+1}, and only $3m+2$ qubits are needed for representation.

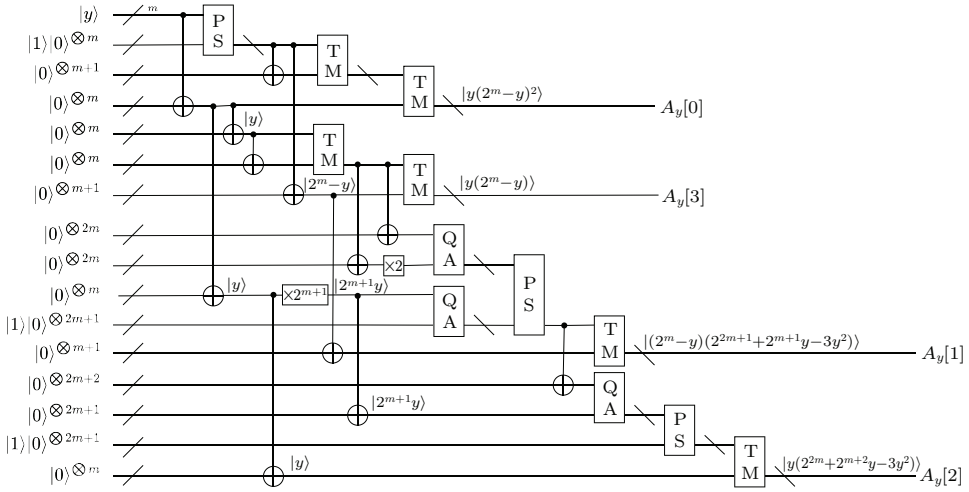


Fig. 15. Weight calculation circuit.

It should be noted that when performing multiplication operations, the numbers of the input qubit and the output qubit must be equal, so the numbers with fewer digits need to be padded with zeros at high digits. As a result, the final output result may have more digits, which is inevitable. Fortunately, the qubit values above the $3m+2$ qubits must be zero, so we can simply replicate the lowest $3m+2$ qubits when generating the quantum matrix. The circuit is shown in Fig. 15. And as shown in Fig. 16, the simplified circuit is referred to as the AC module.

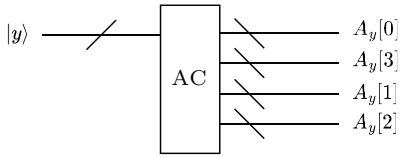


Fig. 16. AC module.

Step 3 Obtain the 16-neighborhood information in the area where the original image position is located

Bilinear interpolation only requires four neighboring pixels, so it directly copies the position information using multiple CNOT modules. However, bicubic interpolation needs 16 neighboring pixels, which would increase the number of qubits significantly if each quantum operation $\Omega_{Y,X}$ is copied. Here we note that the input coordinates $|YX\rangle$ are still $|YX\rangle$ after processing by the quantum operation $\Omega_{Y,X}$. This means that these two input parts can be reused, equivalent to auxiliary qubits in quantum computing. Figure 17 shows the full quantum neighborhood pixel extraction circuit, consisting of 16 quantum operations.

Figure 18 depicts the simplified quantum circuit, denoted by Ω_{16}.

Regarding the processing of boundary pixels, if the calculated (Y,X) is a point on the boundary, then there will be null values around it. Here, the points that exceed the boundary are set to be equal to the nearest boundary point adjacent to them. It can be implemented by using the U_1 module and the U_2 module.

Step 4 Prepare weight information and neighborhood information into 8 matrices and perform matrix multiplication

Taking the matrix multiplication of $f'_1(Y',X')=\boldsymbol{A}'_1\cdot\boldsymbol{B}_1\cdot\boldsymbol{C}'_1$ for example, the remaining three matrix multiplications are also in this mode. First, its structure is analyzed. Matrix multiplication is a 1×2 matrix $\boldsymbol{A}'_1$ multiplied by a 2×2 matrix $\boldsymbol{B}_1$ to obtain a 1×2 intermediate matrix $\boldsymbol{G}$, and then this intermediate matrix is multiplied by a 2×1 matrix $\boldsymbol{C}'_1$ to obtain the final result. However, in the process of processing, since the 1×2 matrix $\boldsymbol{G}$ can be regarded as a vector with 2 elements, $\boldsymbol{G}\cdot\boldsymbol{C}'_1$ can be regarded as two arrays with two elements multiplied. Instead of using matrix multiplication, in Ref. [23] a

simplified array multiplication method is used, and the quantum preparation process of 1×2 arrays and matrices are also the same. Therefore, the entire circuit can be divided into two parts. The first part completes $\boldsymbol{A}'_1 \cdot \boldsymbol{B}_1$ and generates an intermediate result $\boldsymbol{G}$ and the second part treats $\boldsymbol{G}$ and $\boldsymbol{C}'_1$ as arrays to achieve array multiplication. It should be noted that when preparing weight information into a quantum matrix, only the lower $3m+2$ qubits need to be copied by using a 3-CNOT gate because all digits higher than $3m+2$ are 0. The circuit is shown in Fig. 19.

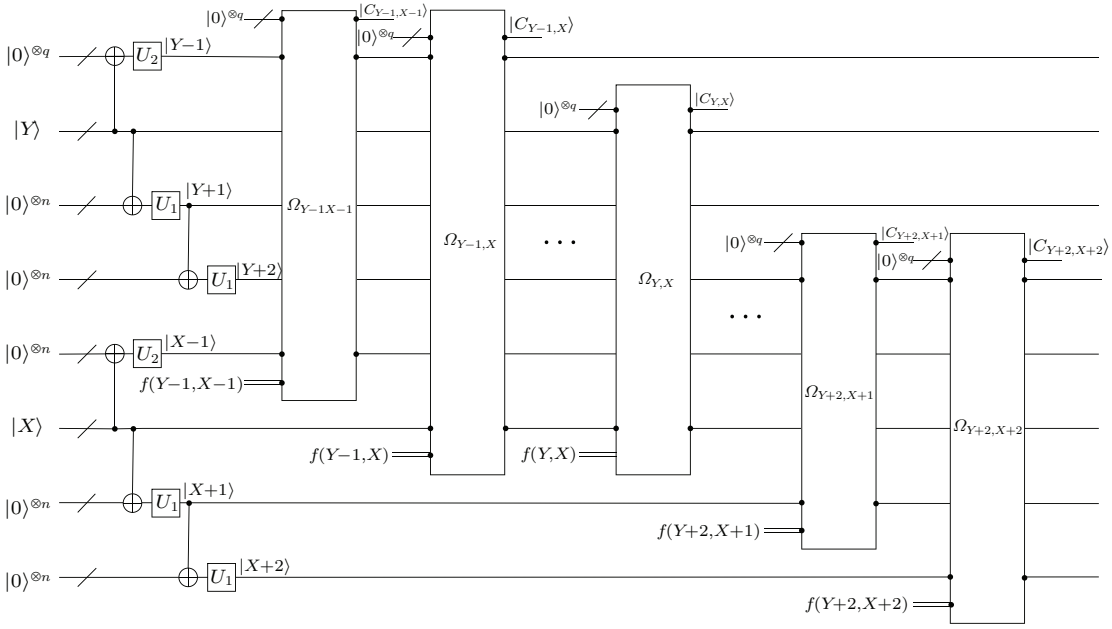


Fig. 17. Neighborhood pixel extraction circuit.

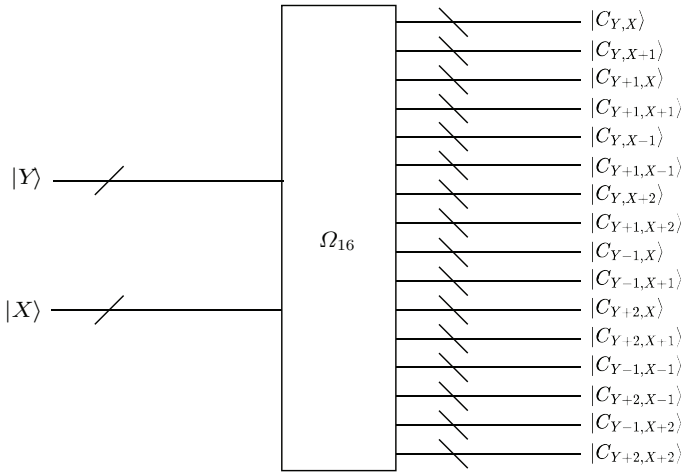


Fig. 18. Simple neighborhood pixel extraction circuit.

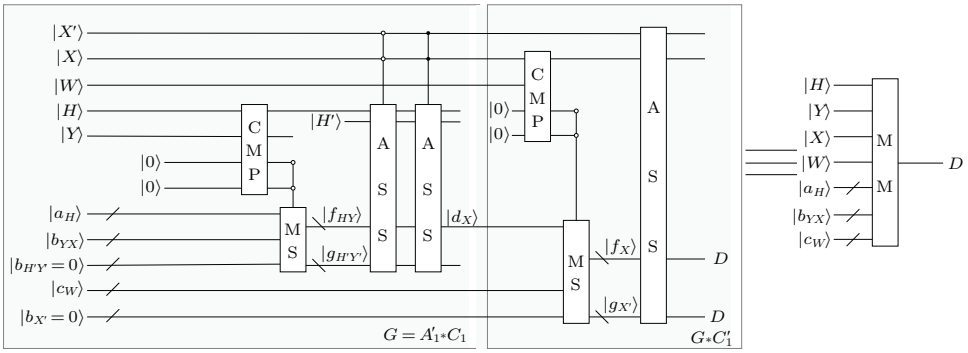


Fig. 19. Matrix operation circuit.

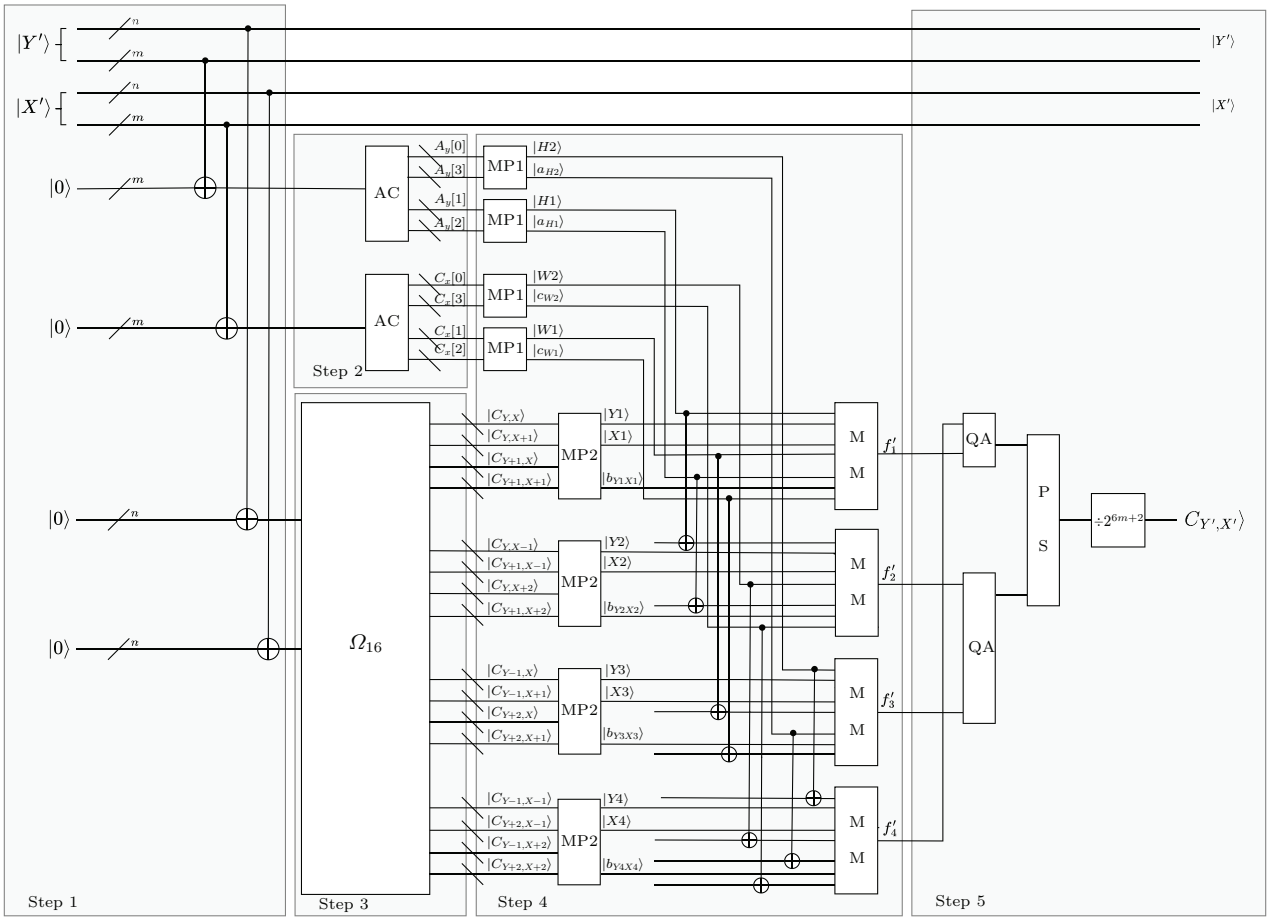


Fig. 20. Quantum image scaling-up scheme.

Step 5 Obtain the pixel intensity values of the interpolated image I'

By the above steps, the matrix multiplication results of the four matrices in Eq. (29) are obtained as $f'_1(Y',X')$, $f'_2(Y',X')$, $f'_3(Y',X')$, $f'_4(Y',X')$. Then quantum adders and quantum subtractors are used according to the formula, and finally we can obtain the pixel value intensity $f'(Y',X')$ at position (Y',X') by using the divide by 2^{6m+2} module. The complete circuit is shown in Fig. 20.

4.2. Quantum image scaling-down scheme

When shrinking quantum images, because this paper is based on NEQR, it can only be shrunk by integer. According to the coordinate mapping formula, the values of the h and w are both zero, so $\boldsymbol{A} = [0,1,0,0]$, $\boldsymbol{C} = [0,1,0,0]^{\mathrm{T}}$. At this time, $f'(Y',X') = \boldsymbol{A}\cdot\boldsymbol{B}\cdot\boldsymbol{C} = f(Y,X)$. In this case, there is no need to perform calculations like in the case of quantum scaling-up circuit. The quantum circuit is shown in Fig. 21.

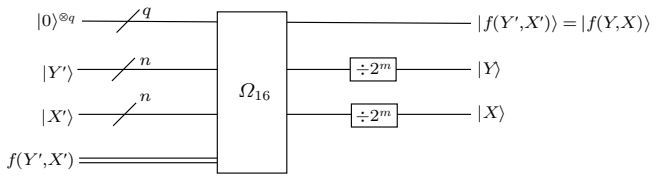


Fig. 21. Quantum image scaling-down scheme.

It should be noted that in this bicubic interpolation method, when the shrinkage ratio is an integer, the scaling algorithm is the nearest-neighbor interpolation algorithm. Therefore, the designed circuit is also equivalent to the nearest-neighbor interpolation, which is determined by the formula of cubic convolution itself. When performing integer scaling in practice, classical image processing needs to use other scaling algorithms more suitable for integer scaling, which is not involved in this work.

5. Network complexity analysis

Quantum circuit complexity, unlike its classical counterpart, is primarily determined by the number of elementary gates, consisting of CNOT gates and arbitrary 2×2 unitary operator, each with a unit time complexity.[34] In this section, the complexity of each module and the entire circuit is analyzed and compared with the counterparts in previous work.

5.1. Complexity analysis of each module

Multiple CNOT module In order to duplicate n qubits, n CNOT gates are required, with a circuit complexity being $O(n)$.

U_1 module and U_2 module Both use roughly the same elementary quantum gates, consisting of an n-CNOT gate,

a CNOT gate, a Toffoli gate, and $(n-2)$ m-CNOT ($m = 3,4,\ldots,n$) gates. It is known that an m-CNOT gate is composed of $2m-1$ Toffoli gates and a CNOT gate.[34] In summary, the above circuit is composed of $6n^2+7n-18$ CNOT gates, with a circuit complexity being $O(n^2)$.

Quantum operation Ω_{YX} For a grayscale image with a grayscale range of 2^q and a size of $2^n \times 2^n$, a single quantum operation $\Omega_{Y,X}$[7] has a network complexity of $O(q \cdot 2n)$.

Arithmetic module According to Refs. [26–29], the PS module has a circuit complexity of $O(n)$, while the QA module and the TM module have a complexity of $O(n^2)$ each.

Multiply by 2 and divide by 2, multiply by 2^m and divide by 2^m modules For an n-qubit number, only n swap gates are needed, a swap gate is composed of 3 CNOT gates.[34] Hence, these modules have a circuit complexity of $O(n)$.

Matrix preparation module We only analyze the network complexity of the 1×2 and 2×2 matrix preparation modules. If all elements to be prepared in the matrix are n-qubit numbers, then the MP1 module requires $2n$ Toffoli gates and the MP2 module requires $4n$ 3-CNOT gates. Therefore, the network complexity of the matrix preparation module is $O(n)$.

Matrix multiplication calculator module AC Reference [31] gave a time complexity of $O(MS\log N)$ for multiplying two matrices of size MN by NS. However, the matrix size is very small in this work. In this case, the network complexity within the matrix regarding the size of the numbers needs to be considered . Herein the range of numbers in the matrix is assumed to be 2^q.

CMP module According to Ref. [32], for comparing two n-qubit numbers, the circuit complexity is $O(n^2)$.

ASS module For a quantum superposition state with n elements, the ASS[33] module has a circuit complexity of $O(\log(n))$. The circuit complexity of the ASS module is related to array size logarithmically, but also to the range of element values. For a quantum superposition state with a value range of 2^q, the network complexity is $O(q)$.

MS module For performing MS module operations on two q-qubit numbers, the MS module consists of one multiplier module[35] and n SS modules. The module circuit complexity is $O(q^3)$.

5.2. Complexity analysis of quantum image scaling-up scheme

Step 1 Obtain position mapping: $2(n+m)$ CNOT gates are required, with a circuit complexity of $O(m+n)$.

Step 2 Obtain weight information: An AC module has 12 multiple CNOT gates, 3 PS modules, 3 QA modules and 6 TM modules. The circuit complexity of 12 multiple CNOT gates is $O(m+m+m+m+m+1+m+m+1+m+1+2m+2m+m+2m+2m+2) \approx O(m)$. Then, the network complexity of 3 PS modules is $O(m+2m+2m+2) \approx O(m)$ and the network complexity of 3 QA modules is $O((2m)^2+(2m+2)^2+(2m+2)^2) \approx O(m^2)$. Finally, the time complexity of the 6 TM number multiplication modules is $O((m+1)^2+(2m+2)^2+m^2+(2m)^2+(2m+2)^2+(2m+2)^2) \approx O(m^2)$.

Therefore, the circuit complexity of the entire weight calculation module AC is $O(m^2)$. To obtain eight weights, two AC modules are required to calculate the weights in the y direction and x direction, respectively.

Step 3 Acquiring neighborhood information: A neighborhood information acquisition module requires four U_1 modules and two U_2 modules, the circuit complexity is $O(6n^2)$. The six multiple CNOT gates have a complexity of $O(6n)$. Additionally, 16 quantum operation circuits are required. Hence, the overall network complexity of this step is $O(6n^2+6n+16q \cdot 2n) \approx O(n^2+qn)$.

Step 4 Calculate the result of each matrix: There are four 2×2 matrix preparation circuits with a complexity of $O(4q)$ and four 1×2 matrix preparation circuits with a complexity of $O(4(3m+2))$. Four CNOT gates and $4(3m+2)$ multiple CNOT gates have a circuit complexity of $O(4(3m+2)+4)$. There are also four MM modules in the end, and the MM module is analyzed below. Firstly, X, W, H, Y are all single qubits. a_H and c_W are weights with $3m+2$ qubits. b_{TX} is a pixel value occupying q qubits. Then the time complexity of two CMP modules is $O(1)$. The input qubits of the MS module must be equal. Let $\alpha = \max(3m+2,q)$, so the circuit complexity of the first MS module is $O(\alpha^3)$ and each output qubit number of the first MS module must be 2α. The circuit complexity of the second MS module is $O((2\alpha)^3)$. The circuit complexities of the first two ASS modules are both $O(2\alpha)$ and the circuit complexity of the last ASS module is $O(4\alpha)$. Hence, the overall complexity of the circuit for this step is $O(4(3m+2)+4q+4(3m+2)+4+\alpha^3+(2\alpha)^3+2\alpha+4\alpha) \approx O(\alpha^3)$.

Step 5 Get the final result: This step has two QA modules, a PS module and a divide by 2^{6m+2} module. The circuit complexity is $O((4\alpha)^2+(4\alpha)^2+4\alpha+4\alpha) \approx O(\alpha^2)$.

In summary, the overall complexity of the quantum image scaling-up circuit is given below:

$$O(m+n+m^2+n^2+qn+\alpha^3+\alpha^2) \approx O(n^2+qn+\alpha^3). \quad (39)$$

5.3. Complexity analysis of quantum image scaling-down scheme

Compared with the scaling-up circuit, the scaling-down circuit is much simple and only requires a quantum operation $\Omega_{Y,X}$ with a complexity of

$$O(q \cdot 2n+2n) \approx O(qn). \quad (40)$$

5.4. Complexity discussion

Table 1 shows the comparison of time complexity among different scaling methods applied to a quantum image of size $2^n \times 2^n$ and gray scale range 2^q. Obviously, compared with classical image scaling based on bicubic interpolation, our scheme has achieved exponential acceleration. Moreover, it has lower complexity than Zhou *et al.*'s quantum bilinear interpolation image scaling method based on NEQR.

A circuit with cubic time complexity is designed to calculate the pixel values for the quantum scaling-up circuit. However, limited by the complexity of bicubic interpolation itself, a large number of quantum modules are also required, which leads to a very complex final circuit that needs a lot of auxiliary qubits. Although this is currently unavoidable, we are still trying to minimize the number of auxiliary qubits as much as possible.

Table 1. Comparison of complexity among several image scaling methods.

Method	Time or circuit complexity
Classical bicubic interpolation	$O(2^{2n})$
Zhou *et al.*'s[13] quantum bilinear interpolation	$O(2^{n+3})$
Our quantum bicubic interpolation scheme	$O(n^2+qn+\alpha^3)$

6. Simulation-based experiments and analysis

In this section, the cubic convolution interpolation method proposed in this paper is compared with the nearest-neighbor interpolation method[11] and the bilinear interpolation[13] method. Because current physical quantum computers lack universality and quantum computing virtual machines cannot support many qubits, a series of simulation experiments are carried out in the MATLAB R2018b environment for evaluation. Among them, complex vectors represent quantum states and unitary matrices represent unitary transformations.

6.1. Image scaling-up simulation

A 64×64 Lena grayscale image is chosen as the original image for the quantum magnification circuit simulation experiment as shown in Fig. 22(c). We use three interpolation methods to magnify it by integer multiples of 2 and 4, respectively. The magnified images are displayed in the figure. It can be seen that the bilinear interpolation and bicubic interpolation in Figs. 22(d), 22(e) and 22(f), 22(g) are subjectively significantly better than the nearest-neighbor interpolation in Figs. 22(a) and 22(b). Additionally, for Fig. 22(f) *versus* Fig. 22(d), Fig. 22(g) *versus* Fig. 22(e), the former has less jagged edges, smoother image, and better effect than the latter.

6.2. Image scaling-down simulation

For image scaling-down, due to the limitation of the method used in this paper,[24] the integer scaling ratio is equivalent to the nearest-neighbor interpolation. We compare it directly with nearest-neighbor interpolation. A 128×128 Lena image is selected as the original image and shrunk twice as shown in Fig. 23. It can be seen that figures 23(b) and 23(c) have the same effect.



Fig. 22. Simulated image scale-up by using different interpolation method. (a) Nearest (128×128), (b) nearest (256×256), (c) original (64×64), (d) bilinear (128×128), (e) bilinear (256×256), (f) bicubic (128×128), (g) bicubic (256×256).



Fig. 23. Simulatied image scale-down using different interpolation methods: (a) original (128×128), (b) nearest (64×64), (c) bicubic (64×64).

6.3. Comparison of peak signal-to-noise ratios (PSNRs)

It is noted that our comparison is based on human visual perception, which can be subjective. To provide a more objective assessment of the image scaling performance, we use the peak signal-to-noise ratio as a metric. PSNR is a widely

adopted measure of image quality in image processing. It is defined for two images of size $2^n \times 2^n$ as follows:

$$\mathrm{PSNR} = 20 \cdot \log_{10}(\mathrm{MAX}_I/\sqrt{\mathrm{MSE}}), \tag{41}$$

$$\mathrm{MSE} = \frac{1}{2^{2n}} \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} [I(i,j) - K(i,j)]^2, \tag{42}$$

where MAX_I represents the maximum pixel value in the image, which is usually 255, and MSE denotes the mean squared error between the original image I and the interpolated image K. The higher the PSNR value, the lower the distortion of K is.

The PSNR is a common metric for measuring the reconstruction quality of lossy compressed images. It should be noted that the calculation of PSNR requires two images of equal size. Therefore, before computing the PSNR, the interpolated image needs to be rescaled inversely proportional to its original size. The algorithm proposed in this work performs integer scaling on quantum images, which preserves their original pixels. Unlike other methods, our algorithm does not introduce blurring or distortion when scaling up and then scaling down an image, making PSNR an unsuitable metric for evaluation. Instead, we scale down and then scale up an image to measure the quality of our algorithm. Despite the increased distortion and reduced PSNR caused by this process, our results show that our quantum bicubic convolution interpolation algorithm outperforms other quantum interpolation methods.

Using different quantum interpolation methods, we first scale down the original image of size 128×128 by a factor of two and then scale them up by the same factor. Figures 24 and 25 both show the comparison between the original image and the interpolated image, and Table 2 lists their PSNR values. The results demonstrate that our proposed quantum cubic convolution interpolation method outperforms the nearest-neighbor interpolation in terms of both visual quality and PSNR. Moreover, our scheme achieves higher PSNR than quantum bilinear interpolation, although the visual difference is subtle.



Fig. 24. Lena and images interpolated by using different quantum image processing methods: (a) original, (b) nearest, (c) bilinear, (d) bicubic.



Fig. 25. Peppers and images interpolated by using different quantum image processing methods: (a) original, (b) nearest, (c) bilinear, (d) bicubic.

Table 2. PSNR (dB) results for interpolated images produced by using nearest-neighbor bilinear method and bicubic interpolation method. Labels (b), (c), and (d) are interpolated images in Figs. 24 and 25.

Original image	PSNR(dB)		
	(b)	(c)	(d)
Fig. 24(a) Lena	21.3759	25.5749	25.7742
Fig. 25(a) Peppers	21.0604	25.4882	25.9290

7. Conclusions and prospects

In this work, a bicubic interpolation algorithm based on NEQR is proposed for the first time. Moreover, a circuit based on quantum matrix multiplier is designed to calculate the pixel values of the interpolated image by implementing the matrix multiplication operation of eight weights and sixteen neighborhood information. We analyzed and simulated our scheme and showed that it has cubic complexity, achieving an exponential speedup over classical image scaling. The simulation results also demonstrate better performance than those from the nearest-neighbor interpolation method and bilinear interpolation method proposed by previous researchers.

This work also has some limitations, which are the directions for our future work:

(i) Due to the characteristics of NEQR, our method only supports integer scaling ratios for image enlargement and reduction. We plan to generalize our algorithm to other quantum representations such as GQIR and FRQI in the future.

(ii) Since the convolution kernel used will cause most of the weights to be zero when shrinking by an integral multiple,

the shrinking circuit is equivalent to the nearest-neighbor interpolation, and the final result is not satisfactory. Therefore, designing a quantum circuit suitable for integer multiple reduction by using a proper convolution kernel is also our next goal.

(iii) Too many auxiliary qubits are used by our algorithm, especially in the weight calculation module and matrix operation module. This problem is currently unavoidable and limited by the development of quantum computing modules. Our circuit has been optimized as much as possible and will be further improved in the future.

Acknowledgements

Project supported by the Scientific Research Fund of Hunan Provincial Education Department, China (Grant No. 21A0470), the Natural Science Foundation of Hunan Province, China (Grant No. 2023JJ50268), the National Natural Science Foundation of China (Grant Nos. 62172268 and 62302289), and the Shanghai Science and Technology Project, China (Grant Nos. 21JC1402800 and 23YF1416200).

References

[1] Feynman R P 1982 *Int. J. Theor. Phys.* **21** 467
[2] Shor P W 1994 *Proceedings of 35th Annual Symposium on Foundations of Computer Science* pp. 124–134
[3] Grover L K 1996 *Proceedings of the 28th Annual ACM symposium on the Theory of Computing* pp. 212–219
[4] Venegas-Andraca S and Bose S 2003 *Proc. SPIE* 5105, *Quantum Information and Computation August* **4** 2003
[5] Latorre J I 2005 arXiv: quant-ph/0510031 [hep-ph]
[6] Le P Q, Dong F and Hirota K 2011 *Quantum Inf. Process.* **10** 63
[7] Zhang Y, Lu K, Gao Y and Wang M 2013 *Quantum Inf. Process.* **12** 2833
[8] Jiang N and Wang L 2015 *Quantum Inf. Process.* **14** 1559
[9] Jiang N, Wang J and Mu Y 2015 *Quantum Inf. Process.* **14** 4001
[10] Fan P, Zhou R G and Jing N 2016 *Inf. Sci.* **340** 191
[11] Sang J, Wang S and Niu X 2016 *Quantum Inf. Process.* **15** 37
[12] Zhou R G, Tan C and Fan P 2017 *Mod. Phys. Lett. B* **31** 1750184
[13] Zhou R G, Hu W, Fan P and Ian H 2017 *Sci. Rep.* **7** 2511
[14] Zhou R G, Liu X and Luo J 2017 *Int. J. Theor. Phys.* **56** 2966
[15] Li P and Liu X 2018 *Int. J. Quantum Inf.* **16** 1850031
[16] Yan F, Zhao S, Venegas-Andraca S E and Hirota K 2021 *Digital Signal Processing* **117** 103149
[17] Gao C, Zhou R G, Li X and Li Y C 2022 *Quantum Inf. Process.* **21** 270
[18] Gao C, Zhou R G and LI X 2023 *Chin. Phys. B* **32** 050303
[19] Yao X W, Wang H, Liao Z, Chen M C, Pan J, Li J, Zhang K, Lin X, Wang Z, Luo Z, Zheng W, Li J, Zhao M, Peng X and Suter D 2017 *Phys. Rev. X* **7** 031041
[20] Ali A E, Abdel-Galil H and Mohamed S 2020 *Quantum Inf. Process.* **19** 238
[21] Luo J, Zhou R G, Hu W W, Li Y and Luo G F 2022 *Chin. Phys. B* **31** 040302
[22] Zhang Y, Lu K, Xu K, Gao Y and Wilson R 2015 *Quantum Inf. Process.* **14** 1573
[23] Huo F, Sun X and Ren W 2020 *Multimed Tools Appl.* **79** 2447
[24] Keys R 1981 *IEEE Trans. Acoust., Speech, Signal Process.* **29** 1153
[25] Le P Q, Iliyasu A M, Dong F and Hirota K 2011 *Theoretical Computer Science* **412** 1406
[26] Ruiz-Perez L and Garcia-Escartin J C 2017 *Quantum Inf. Process.* **16** 152
[27] Thapliyal H and Ranganathan N 2009 *Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Tampa* pp. 229–234
[28] Thapliyal H and Ranganathan N 2011 2011 *11th IEEE Conference on Nanotechnology (IEEE-NANO), IEEE* pp. 1430–1435
[29] Munoz-Coreas E and Thapliyal H 2019 *IEEE Trans. Comput.* **68** 729
[30] Li P, Liu X and Xiao H 2017 *Int. J. Theor. Phys.* **56** 3690
[31] Li H, Jiang N, Wang Z, Wang J and Zhou R 2021 *Int. J. Theor. Phys.* **60** 2037
[32] Wang D, Liu Z H, Zhu W M and Li S Z 2012 *Comput. Sci.* **39** 302
[33] Lu X, Jiang N, Hu H and Ji Z 2018 *Int. J. Theor. Phys.* **57** 2575
[34] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[35] Kotiyal S, Thapliyal H and Ranganathan N 2014 2014 *27th International Conference on VLSI Design and* 2014 *13th International Conference on Embedded Systems* 2014 *27th International Conference on VLSI Design* pp. 545–350