OPEN

Scaling up and down of 3-D floating-point data in quantum computation

Meiyu Xu¹, Dayong Lu^{2✉} & Xiaoyun Sun³

In the past few decades, quantum computation has become increasingly attractive due to its remarkable performance. Quantum image scaling is considered a common geometric transformation in quantum image processing, however, the quantum floating-point data version of which does not exist. Is there a corresponding scaling for 2-D and 3-D floating-point data? The answer is yes. In this paper, we present a quantum scaling up and down scheme for floating-point data by using trilinear interpolation method in 3-D space. This scheme offers better performance (in terms of the precision of floating-point numbers) for realizing the quantum floating-point algorithms than previously classical approaches. The Converter module we proposed can solve the conversion of fixed-point numbers to floating-point numbers of arbitrary size data with $p + q$ qubits based on IEEE-754 format, instead of 32-bit single-precision, 64-bit double-precision and 128-bit extended-precision. Usually, we use nearest-neighbor interpolation and bilinear interpolation to achieve quantum image scaling algorithms, which are not applicable in high-dimensional space. This paper proposes trilinear interpolation of floating-point data in 3-D space to achieve quantum algorithms of scaling up and down for 3-D floating-point data. Finally, the quantum scaling circuits of 3-D floating-point data are designed.

Quantum computation is a theoretical computation system that performs operations on data by using quantum-mechanical phenomena (such as superposition and entanglement). In 1982, Feynman¹ proposed the concept of quantum computer at first. After that, the integer factoring problem² and database search algorithm³ were essential evidences supporting the power of quantum computers. Researches in quantum image processing field started with proposals on quantum image representations such as Qubit Lattice⁴, Real Ket⁵ and Entangled Image⁶. After 2011, some more practical quantum image models using entangled states (FRQI⁷) and superposition states (NEQR⁸, INEQR⁹, GQIR¹⁰, QRDS¹¹, GFPRQS¹², QR2-DD¹³, etc.) have been proposed. With the proposal of NEQR representation, Chetia et al.¹⁴ introduced a quantum image edge detection algorithm based on NEQR. Chakraborty et al.¹⁵ provided a brief survey of the current status of research in the quantum image processing paradigm in 2018. In addition, Chakraborty et al.¹⁶ proposed a combination of three different approaches for representing color quantum images in ternary quantum system. In 2018, Chakraborty et al.¹⁷ proposed and designed a circuit level implementation of the quantum multilevel threshold based color image segmentation technique.

At present, quantum image processing research has been paid more attention to the transform domain. Fijany and Williams¹⁸ studied fast algorithms and complete circuits for quantum wavelet transforms (QWT). Caraiman and Manta¹⁹ introduced quantum image filtering in the frequency domain. Quantum arithmetic operations were given by Ruiz-Perez and Garcia-Escartin²⁰ based on quantum Fourier transform (QFT). Li and Sun²¹ presented quantum color image filtering in the frequency domain. There are many research results in the transform domain. Asaka et al.²² discussed an implementation of the fast Fourier transform (FFT) as a quantum circuit. The quantum version of the FFT (QFFT) is defined as a transformation of a tensor product of quantum states. Chakraborty et al.^{23,24} proposed the image denoising schemes based on QWT. Chang and Vasilakos²⁵ presented core concepts of QFT and inverse QFT and given the reason of why QFT and inverse QFT are able to give exponential speed-up for FFT. Grigoryan and Aghaian²⁶ proposed a new quantum representation of images: Fourier transform representation. Indeed, the QFT is a key ingredient of many important quantum algorithms, including Shor's factoring algorithm and the quantum phase estimation algorithm to estimate the eigenvalues of a unitary operator. Therefore, we propose the design scheme of quantum algorithm for 3-D floating-point data based on QFT in this paper.

¹School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang 471000, China. ²School of Mathematics and Statistics, Henan University, Kaifeng 475001, China. ³College of Information and Management Science, Henan Agricultural University, Zhengzhou 450000, China. ✉email: dayonglu@163.com

In recent years, quantum image scaling as a kind of image geometric transformation has been widely studied and applied in quantum image processing. Jiang and Wang⁹ proposed quantum algorithms and circuits to realize the quantum image scaling based on the INEQR model for quantum images using the nearest neighbor interpolation. Jiang et al.¹⁰ given an improved quantum image representation GQIR and proposed a quantum algorithm to scale up quantum images based on nearest-neighbor interpolation with integer scaling ratio. Sang et al.²⁷ constructed the quantum circuits of the nearest-neighbor interpolation method for FRQI and NEQR. Zhou et al.²⁸ proposed the bilinear interpolation method for NEQR and given the corresponding quantum realization circuits. Li and Liu²⁹ designed quantum image scaling using bilinear interpolation method based on QFT. Zhou et al.³⁰ given quantum image scaling based on bilinear interpolation with arbitrary scaling ratio. However, as far as we know, three-dimensional (3-D) quantum floating-point data versions of which do not exist.

The problem of classical 3-D image (or data) reconstruction is currently a hot topic, such as the reconstruction technology of 2-D to 3-D data. We given a method to convert a 3-D data into 2-D data based on QR2-DD¹³. The dimensionality reduction from 3-D to 2-D data can reduce the number of qubits, and consequently, resources are saved at location information. As far as we know, the inverse problem, that is, 2-D to 3-D reconstruction problem, such as computed tomography (CT) imaging, has important practical significance. However, the quantum floating point data are not discussed. Inspired by this, we conducted preliminary explorations on the reconstruction of 2-D to 3-D quantum floating-point data. Therefore, the further research of our work can provide research ideas for the reconstruction of 2-D to 3-D quantum data.

Floating-point arithmetic precision is limited in length, such as the IEEE single (double, extended) precision format is 32-bit (64-bit, 128-bit) long. However, some problems require a longer floating-point format because of round-off errors. Such problems are usually solved in arbitrary precision (p+q)¹³.

Compared with fixed-point numbers, floating-point numbers offer great savings in the number of qubits when the required range of values or relative precision is large. It is very meaningful to consider the quantum scaling of floating-point data in 3-D space. Li and Liu²⁹ designed quantum image scaling using bilinear interpolation method based on QFT, in which two core operations (i.e., addition and multiplication) are implemented using QFT. Therefore, we consider using trilinear interpolation method to study the quantum scaling of floating-point data in 3-D space. In this paper, we present a quantum scaling up and down scheme of floating-point data by using trilinear interpolation method based on QFT in 3-D space. It has three main contributions:

- The trilinear interpolation method for quantum scaling up and down of 3-D floating-point data is proposed for the first time.
- The generalized floating-point quantum representation of 3-D data in this paper can represent the arbitrary precision (p+q).
- A Converter module for converting fixed-point numbers to floating-point numbers is proposed.

In addition, based on QFT, we design the addition and multiplication (Q-Adder and Q-Multiplier modules) of 3-D floating-point data. Combining some basic modules in this paper, we propose the design scheme of the quantum scaling up and down of 3-D floating-point data using trilinear interpolation method based on QFT. Finally, we give the quantum scaling circuits of 3-D floating-point data.

The remainder of this paper is organized as follows: “Preliminaries” section gives a brief introduction to quantum representation of 3-D floating-point numbers, classical interpolation methods and QFT. After the description of the basic modules in “Some modules” section, the addition and multiplication of the floating-point based on QFT (Q-Adder and Q-Multiplier) and Converter for converting fixed-point numbers to floating-point numbers are designed in “Floating-point addition and multiplication based on QFT” section. “Quantum scaling-up circuit for 3-D floating-point data” and “Quantum scaling-down circuit for 3-D floating-point data” sections introduce the design scheme of quantum scaling up and down for 3-D floating-point data using trilinear interpolation method. “Complexity analysis” section analyzes the computational complexity of the scaling circuits based on the elementary gates. Finally, conclusions and future research works are described in “Conclusions” section.

Preliminaries

In this section, we briefly explain an overview of the foundation of the proposed methodology, including quantum representation of 3-D floating-point numbers, classical interpolation methods and QFT.

A quantum gate is simply an operator that acts on qubits. Such operators can be represented by unitary matrices. Some of the basic gates and their corresponding matrices are shown in (1). The identity gate (I), Hadamard gate (H), NOT gate (X), controlled-NOT gate ($CNOT$) and controlled-phase gate (CR_k) are well-known³¹, here,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, CR_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix}. \quad (1)$$

Format of floating-point numbers in IEEE-754 (Institute of Electrical and Electronics Engineers, IEEE)¹³ is shown in Table 1.

Generalized floating-point quantum representation of 3-D data. Suppose binary sequence $|s_{YXZ}^0 s_{YXZ}^1 \dots s_{YXZ}^{p-1} s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1}\rangle$ encodes the floating-point number $|S_{YXZ}\rangle_F$ of 3-D data corresponding to

s	e	f
signbit	exponent	fractional(mantissa)

Table 1. Format of floating-point numbers in IEEE-754. **Figure 1.** 3-D data.

l	p	$q - 1$
s	e	f
s_{YXZ}^0	$s_{YXZ}^1 s_{YXZ}^2 \dots s_{YXZ}^p$	$s_{YXZ}^{p+1} s_{YXZ}^{p+2} \dots s_{YXZ}^{p+q-1}$
S_{YXZ}^s	S_{YXZ}^e	S_{YXZ}^f

Table 2. Floating point format of $|S_{YXZ}\rangle_F$.

the location of $|YXZ\rangle$. This proposal integrates information about a $H \times W \times Z$ 3-D data in Fig. 1 into a quantum state having it formula in (2):

$$|D_3\rangle = \frac{1}{2^{\frac{h+w+l}{2}}} \left(\sum_{Y=0}^{H-1} \sum_{X=0}^{W-1} \sum_{Z=0}^{L-1} |S_{YXZ}\rangle_F \otimes |YXZ\rangle \right), \quad (2)$$

where

$$\begin{aligned} |YXZ\rangle &= |Y\rangle |X\rangle |Z\rangle = |y_0 y_1 \dots y_{h-1}\rangle |x_0 x_1 \dots x_{w-1}\rangle |z_0 z_1 \dots z_{l-1}\rangle, \quad y_i, x_i, z_i \in \{0, 1\}, \\ |S_{YXZ}\rangle_F &= |s_{YXZ}^0 s_{YXZ}^1 \dots s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1}\rangle, \quad s_{YXZ}^i \in \{0, 1\}, \\ h &= \begin{cases} \lceil \log_2 H \rceil, & H > 1; \\ 1, & H = 1, \end{cases} \quad w = \begin{cases} \lceil \log_2 W \rceil, & W > 1; \\ 1, & W = 1, \end{cases} \quad l = \begin{cases} \lceil \log_2 L \rceil, & L > 1; \\ 1, & L = 1. \end{cases} \end{aligned} \quad (3)$$

Here, $|YXZ\rangle$ is the location information, $|S_{YXZ}\rangle_F = |s_{YXZ}^0 s_{YXZ}^1 \dots s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1}\rangle$ can store a floating-point number of 3-D data, and $\lceil \cdot \rceil$ denotes rounding up. Let $S_{YXZ}^s = s_{YXZ}^0$ is a sign qubit, an encoded exponent field of p qubits denoted by $S_{YXZ}^e = s_{YXZ}^1 s_{YXZ}^2 \dots s_{YXZ}^p$ and a normalized binary fractional having $q - 1$ qubits to the right of the radix point denoted by $S_{YXZ}^f = s_{YXZ}^{p+1} s_{YXZ}^{p+2} \dots s_{YXZ}^{p+q-1}$. That is, floating-point format of $|S_{YXZ}\rangle_F$ is shown in Table 2, and each format's parameters are shown in Table 3.

Classical interpolation methods. 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, linear and cubic interpolation in Fig. 2:

As shown in Fig. 2, the value of the destination pixel (x, c) can be estimated by linear interpolation method, i.e.,

Parameter	Format			
	Single	Single extended	Double	Double extended
p	8	≥ 11	11	≥ 15
q	24	≥ 32	53	≥ 64
Format width in bits	32	≥ 43	64	≥ 79

Table 3. Summary of format parameters.

Figure 2. The principle of interpolation in one direction.

Figure 3. Coordinate map relationship.

$$c = \frac{(x_1 - x)c_0 + (x - x_0)c_1}{x_1 - x_0}. \tag{4}$$

The bilinear interpolation method plays an important role in classical image scaling. For a $W \times H$ (width and height) image, the size of the corresponding interpolated image is $W' \times H'$, which can be described in two steps.
Step 1: Coordinate map.

The coordinate (Y', X') of the interpolated image is restored from the positions (Y, X) , $(Y + 1, X)$, $(Y, X + 1)$ and $(Y + 1, X + 1)$ in the original image. The corresponding relationship is shown in Fig. 3. Here,

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

where $\lfloor \cdot \rfloor$ denotes rounding down.
Step 2: Calculating pixel value.

The pixel value in position (Y', X') of the interpolated image using bilinear interpolation method can be calculated as follows:

Figure 4. Quantum circuits that implement QFT. Not shown are swap gates at the end of the circuit which reverse the order of the qubits³¹.

$$\begin{aligned} \begin{cases} f(Y', X')_1 = (1-w)f(Y, X) + wf(Y, X+1) \\ f(Y', X')_2 = (1-w)f(Y+1, X) + wf(Y+1, X+1) \end{cases} \\ \Rightarrow f(Y', X') = (1-h)f(Y', X')_1 + hf(Y', X')_2 \\ = (1-w)(1-h)f(Y, X) + w(1-h)f(Y, X+1) \\ + (1-w)hf(Y+1, X) + whf(Y+1, X+1). \end{aligned} \quad (6)$$

Hence, the bilinear interpolation method is a single linear interpolation method in the y direction and two single linear interpolation methods in the x direction.

Quantum Fourier transform. The Quantum Fourier Transform, QFT, is an application of classical discrete Fourier transform to the quantum states. The QFT on an orthonormal basis $|0\rangle, |1\rangle, \dots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states:

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}} |k\rangle, \quad (7)$$

where i is an imaginary unit. The specific quantum circuits are shown in Fig. 4, where $|x\rangle = |x_0x_1\dots x_{n-1}\rangle$. For simplicity, in this figure we have omitted the sequence of SWAP gates needed to invert the order of the output qubits.

We can equally define an Inverse Quantum Fourier Transform operator IQFT so that

$$IQFT|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-\frac{2\pi ijk}{N}} |j\rangle. \quad (8)$$

With the direct and the inverse Fourier transforms, we can move back and forth between the computational basis and the phase representation. In our notation, this conversion from the phase encoding to the computational basis $|x\rangle = |x_0x_1\dots x_{n-1}\rangle$ is written as

$$IQFT \cdot QFT|x\rangle = |x\rangle.$$

By employing 1-qubit Hadamard gates H and 2-qubit controlled-phase gates CR_k , the QFT and IQFT can be efficiently implemented.

Quantum scaling up and down

In the following subsections, we first introduced some basic modules: *adding one*, *rotation*, *adder*, *multiplier*, *special subtractor* and *divided by 2* modules, and then we designed the addition and multiplication of the floating-point based on QFT (*Q-Adder* and *Q-Multiplier*) and *Converter* for converting fixed-point numbers to floating-point numbers. Finally, we proposed the design scheme of quantum scaling up and down for 3-D floating-point data.

Some modules. In the next subsection, a series of quantum modules are used. So we introduce their circuits in this subsection.

1. Adding one module based on integer

In this paper, we use the *adding one* module based on integer³², and its quantum circuit is shown in Fig. 5, where $|x\rangle = |x_0x_1\dots x_{n-1}\rangle$, n is a positive integer, $n \geq 1$, $x_0, x_1, \dots, x_{n-1} \in \{0, 1\}$.

2. Rotation module

Figure 5. Adding one module³².

Figure 6. Rotation module³¹.

The operator of the *rotation* module can be expressed as

$$|x\rangle|y\rangle \xrightarrow{\text{Rotation}} e^{i\frac{2\pi xy}{2^{n+1}}} |x\rangle|y\rangle, \quad (9)$$

and the corresponding quantum circuit is shown in Fig. 6, in other words, its effect is to introduce a phase shift in frequency domain, where $|x\rangle = |x_0x_1 \dots x_{n-1}\rangle$.

3. Adder and Multiplier modules

Li²⁹ proposed a new method for the design of two core modules (i.e., addition and multiplication) based on QFT. It is clear that the subtraction operation could be implemented by a bit of modification in CR_k . If we replace the element $e^{2\pi i/2^k}$ in CR_k matrix with $e^{-2\pi i/2^k}$, then the quantum *adder* circuit would act as a quantum *subtractor* whose output will become $|x\rangle|x-y\rangle$. We don't give the detail quantum circuit of the quantum *subtractor* module. The quantum circuits of quantum *adder* and *subtractor* and *multiplier* modules are shown in Figs 7, 8 and 9, respectively.

4. Special Subtractor module

The function of this module is to implement the subtraction of two m -qubit numbers, i.e. $2^m - x$, $0 \leq x \leq 2^m - 1$, and the *special subtractor* quantum circuit is shown in Fig. 10.

5. Divided by 2 module

Zhang et al.¹² design the *divided by 2* module in 2020. The *divided by 2* module is to make a floating-point number $|s\rangle_F$ to $|\frac{1}{2}s\rangle_F$, where $|s\rangle_F = |s^0s^1 \dots s^ps^{p+1} \dots s^{p+q-1}\rangle_F$. The circuit for floating-point *divided by 2* module is depicted in Fig. 11.

Figure 7. Adder module²⁹.

Figure 8. Subtractor module²⁹.

Figure 9. Multiplier module²⁹.

Floating-point addition and multiplication based on QFT. In the next subsection, the addition and multiplication of the floating-point based on QFT (*Q-Adder* and *Q-Multiplier*) and *Converter* module are designed. The QFT offers an interesting way to perform arithmetic operations on a quantum computer. Nielsen³¹ given the quantum circuit of the QFT. Adder and Multiplier modules³³ based on floating-point numbers are given. We will benefit from the circuits^{29,33} in our quantum addition and multiplication circuits based on QFT. In this paper, we design the addition and multiplication based on QFT (*Q-Adder* and *Q-Multiplier*) operations, and the quantum circuits are depicted in Figs. 12 and 13. In order to have the same number of bits of the two floating-point numbers that are multiplied, we designed the *Converter* module for converting fixed-point numbers to floating-point numbers in the design scheme of quantum scaling for 3-D floating-point data, and the quantum circuit is depicted in Fig. 14. For convenience, other unremarked qubits are the garbage outputs.

1. *Q-Adder module*

Figure 12 shows the addition of two floating-point numbers $|s\rangle_F$ and $|t\rangle_F$, where $|s\rangle_F = |s^s s^e s^f\rangle_F = |s^s s^1 s^2 \dots s^{p1} s^{p1+1} \dots s^{p1+q1-1}\rangle_F$, $|t\rangle_F = |t^s t^e t^f\rangle_F = |t^s t^1 t^2 \dots t^{p2} t^{p2+1} \dots t^{p2+q2-1}\rangle_F$.

Figure 10. Special Subtractor module²⁹.

Figure 11. Divided by 2 module¹².

Figure 12. Q-Adder module.

Figure 13. Q-Multiplier module.

Figure 14. Converter module.

Step ①. Determine result exponent $\Delta E = s^e - t^e$.

The comparator module based on integer (CMP) is used to compare two exponents information s^e and t^e of $|s\rangle_F$ and $|t\rangle_F$. If $s^e < t^e$, swap the two floating-point numbers (comparison followed by controlled swaps). Align the two results according to the difference in exponents $\Delta E = s^e - t^e$. (only if $\Delta E < q_2 - 1$, else the adder will have no effect, $m = \{1, 2, \dots, q_2 - 1\}$ in ②).

Step ②. Add mantissas in two's complement.

Compute two's complement from sign bits and mantissas (including the implicit leading 1). The second mantissa $|t^f\rangle$ is shifted by the difference of the exponents s^e and t^e . Add mantissas in two's complement.

Step ③. Renormalize the intermediate result.

The final RN gate renormalizes the intermediate result using the first-ones circuit followed by shifting the mantissa by the output of the first-ones circuit (i.e., if adding the two mantissas in ② caused an overflow, right-shift the result by 1 and increment the exponent) and copies out the resulting floating-point representation.

2. Q-Multiplier module

Figure 13 shows the multiplication of two floating-point numbers $|s\rangle_F$ and $|t\rangle_F$, where $|s\rangle_F = |s^s s^1 s^2 \dots s^{p_1} s^{p_1+1} \dots s^{p_1+q_1-1}\rangle_F$, $|t\rangle_F = |t^s t^e t^f\rangle_F = |t^s t^1 t^2 \dots t^{p_2} t^{p_2+1} \dots t^{p_2+q_2-1}\rangle_F$. There is only one renormalization step involved. In summary, it requires the following steps:

- (1). Determine result exponent.
- (2). Multiply mantissas (including the implicit leading 1) into a $(q_1 + q_2 - 2)$ -bit register.
- (3). If there was an overflow, right-shift by one and increment the result exponent.
- (4). The final step denoted by c COPY consists of conditionally copying out of the resulting exponent, mantissa and sign bit.

3. Converter module

In Fig. 14, Converter module converts fixed-point numbers to floating-point numbers, where $|x\rangle = |x_0 x_1 \dots x_{n-1}\rangle$, $|x\rangle_F = |x_0' x_1' \dots x_p' x_{p+1}' \dots x_{p+q-1}'\rangle_F$. Before explaining the circuit, we first need to normalize $|x\rangle$.

Step ①. Calculate the bias value. That is, $+2^{p-1} - 1$ module can be realized by adding one module $2^{p-1} - 1$ times.

Step ②. Calculate the value of the exponent code using the formula $\text{exponent} = \text{bias value} + \text{exponent truth value}$, where the truth value of the exponent code is the value of exponent code after $|x\rangle$ normalization, $+n-1, +n-2, \dots$, similar to Step ①.

Step ③. Calculate the value of the mantissa. If $x_0 = 1, x_1 = 1$, the value of the mantissa can be realized using the Toffoli gate and zero padding at the end of $|x_1 x_2 \dots x_{n-1}\rangle$. Therefore, the remaining cases (example $x_0 = 0, x_1 = 1$ and $x_0 = 0, x_1 = 0, x_2 = 1$) are carried out in this way in sequence. That is to say, determine the position of the first 1.

Quantum scaling-up circuit for 3-D floating-point data. In this subsection, the quantum scaling-up circuit of 3-D floating-point data based on QFT using trilinear interpolation is designed. The key idea of the proposed circuit is mathematically explained in (13).

Without loss of generality, the quantum representation for an array of $2^h \times 2^w \times 2^l$ 3-D floating-point data in (2) ($H = 2^h, W = 2^w, L = 2^l$) can be expressed as follows:

$$|D_3\rangle = \frac{1}{2^{\frac{h+w+l}{2}}} \left(\sum_{Y=0}^{2^h-1} \sum_{X=0}^{2^w-1} \sum_{Z=0}^{2^l-1} |S_{YXZ}\rangle_F \otimes |YXZ\rangle \right), \quad (10)$$

where

$$|YXZ\rangle = |Y\rangle |X\rangle |Z\rangle = |y_0 y_1 \dots y_{h-1}\rangle |x_0 x_1 \dots x_{w-1}\rangle |z_0 z_1 \dots z_{l-1}\rangle, \quad y_i, x_i, z_i \in \{0, 1\},$$

$$|S_{YXZ}\rangle_F = |s_{YXZ}^0 s_{YXZ}^1 \dots s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1}\rangle, \quad s_{YXZ}^i \in \{0, 1\}.$$

The quantum representation uses Y (h qubits), X (w qubits), Z (l qubits), $|S_{YXZ}\rangle_F$ ($p + q$ qubits), respectively, to denote the Y -coordinate, X -coordinate, Z -coordinate and a floating-point number of 3-D data.

Assume that there is an array of $2^h \times 2^w \times 2^l$ 3-D floating-point data and the size of the resulting data after being scaled is $2^{h+h_1} \times 2^{w+w_1} \times 2^{l+l_1}$, i.e., $r_y = 2^{h_1}, r_x = 2^{w_1}$ and $r_z = 2^{l_1}$ (where r_y, r_x and r_z represent the scaling in y -coordinate, x -coordinate, z -coordinate axis directions, respectively). The trilinear interpolation method for 3-D floating-point data can be described within the following two steps in detail.

Step 1: Build coordinate map relationship.

The coordinate position (Y', X, Z') of the interpolated data can build a map relationship with the four positions $(Y + 1, X, Z + 1)$, $(Y + 1, X, Z)$, $(Y, X, Z + 1)$ and (Y, X, Z) of the original data when X is fixed. The coordinate position $(Y', X + 1, Z')$ of the interpolated data can build a map relationship with the four positions $(Y + 1, X + 1, Z + 1)$, $(Y + 1, X + 1, Z)$, $(Y, X + 1, Z + 1)$ and $(Y, X + 1, Z)$ of the original data when $X + 1$

Figure 15. Coordinate map relationship for 3-D.

is fixed. The coordinate position (Y', X', Z') of the interpolated data can build a map relationship with the two positions (Y', X, Z') , $(Y', X+1, Z')$ of the original data when yoz plane is fixed. The corresponding relationship is shown in Fig. 15. Therein,

$$Y = \left\lfloor \frac{Y'}{2^{h_1}} \right\rfloor, X = \left\lfloor \frac{X'}{2^{w_1}} \right\rfloor, Z = \left\lfloor \frac{Z'}{2^{l_1}} \right\rfloor, r = \frac{Y'}{2^{h_1}} - Y, s = \frac{X'}{2^{w_1}} - X, t = \frac{Z'}{2^{l_1}} - Z, \quad (11)$$

where $\lfloor \cdot \rfloor$ represents the rounding down operation, $|Y'\rangle = |Y_{h+h_1-1} Y_{h+h_1-2} \dots Y_1 Y_0\rangle$, $|X'\rangle = |X_{w+w_1-1} X_{w+w_1-2} \dots X_1 X_0\rangle$, $|Z'\rangle = |Z_{l+l_1-1} Z_{l+l_1-2} \dots Z_1 Z_0\rangle$. To build the mapping relationship described in Fig. 15, the multiply Control-Not operations and adding one operation +1 are chosen as the unitary operators. The function of the multiply Control-Not operators is to utilize h Control-Not gates to copy the h qubits $|Y_{h+h_1-1} Y_{h+h_1-2} \dots Y_{h_1}\rangle$ into the h ancillary qubits $|0\rangle^{\otimes h}$. Through these two unitary operators, the interpolation mapping relationship between the position of original data and the interpolated data has been established. The details are described in Figs. 17 and 18.

Step 2: Calculating value for 3-D floating-point data.

According to (6), the value for 3-D floating-point data in position (Y', X', Z') of the interpolated data using trilinear interpolation can be calculated as follows:

$$\begin{aligned} |S_{Y',X,Z'}\rangle &= (1-t)|S_{Y',X,Z'}\rangle_1 + t|S_{Y',X,Z'}\rangle_2 \\ &= (1-r)(1-t)|S_{Y+1,X,Z+1}\rangle + r(1-t)|S_{Y,X,Z+1}\rangle \\ &\quad + (1-r)t|S_{Y+1,X,Z}\rangle + rt|S_{Y,X,Z}\rangle. \\ |S_{Y',X+1,Z'}\rangle &= (1-t)|S_{Y',X+1,Z'}\rangle_1 + t|S_{Y',X+1,Z'}\rangle_2 \\ &= (1-r)(1-t)|S_{Y+1,X+1,Z+1}\rangle + r(1-t)|S_{Y,X+1,Z+1}\rangle \\ &\quad + (1-r)t|S_{Y+1,X+1,Z}\rangle + rt|S_{Y,X+1,Z}\rangle. \\ \Rightarrow |S_{Y',X',Z'}\rangle &= (1-s)|S_{Y',X,Z'}\rangle + s|S_{Y',X+1,Z'}\rangle \\ &= (1-r)(1-t)(1-s)|S_{Y+1,X,Z+1}\rangle + r(1-t)(1-s)|S_{Y,X,Z+1}\rangle \\ &\quad + (1-r)t(1-s)|S_{Y+1,X,Z}\rangle + rt(1-s)|S_{Y,X,Z}\rangle \\ &\quad + (1-r)(1-t)s|S_{Y+1,X+1,Z+1}\rangle + r(1-t)s|S_{Y,X+1,Z+1}\rangle \\ &\quad + (1-r)ts|S_{Y+1,X+1,Z}\rangle + rts|S_{Y,X+1,Z}\rangle. \end{aligned} \quad (12)$$

Therefore,

$$\begin{aligned}
|S_{Y',X',Z'}\rangle &= \left[1 - \left(\frac{Y'}{2^{h_1}} - Y\right)\right] \left[1 - \left(\frac{Z'}{2^{l_1}} - Z\right)\right] \left[1 - \left(\frac{X'}{2^{w_1}} - X\right)\right] |S_{Y+1,X,Z+1}\rangle \\
&+ \left(\frac{Y'}{2^{h_1}} - Y\right) \left[1 - \left(\frac{Z'}{2^{l_1}} - Z\right)\right] \left[1 - \left(\frac{X'}{2^{w_1}} - X\right)\right] |S_{Y,X,Z+1}\rangle \\
&+ \left[1 - \left(\frac{Y'}{2^{h_1}} - Y\right)\right] \left(\frac{Z'}{2^{l_1}} - Z\right) \left[1 - \left(\frac{X'}{2^{w_1}} - X\right)\right] |S_{Y+1,X,Z}\rangle \\
&+ \left(\frac{Y'}{2^{h_1}} - Y\right) \left(\frac{Z'}{2^{l_1}} - Z\right) \left[1 - \left(\frac{X'}{2^{w_1}} - X\right)\right] |S_{Y,X,Z}\rangle \\
&+ \left[1 - \left(\frac{Y'}{2^{h_1}} - Y\right)\right] \left[1 - \left(\frac{Z'}{2^{l_1}} - Z\right)\right] \left(\frac{X'}{2^{w_1}} - X\right) |S_{Y+1,X+1,Z+1}\rangle \\
&+ \left(\frac{Y'}{2^{h_1}} - Y\right) \left[1 - \left(\frac{Z'}{2^{l_1}} - Z\right)\right] \left(\frac{X'}{2^{w_1}} - X\right) |S_{Y,X+1,Z+1}\rangle \\
&+ \left[1 - \left(\frac{Y'}{2^{h_1}} - Y\right)\right] \left(\frac{Z'}{2^{l_1}} - Z\right) \left(\frac{X'}{2^{w_1}} - X\right) |S_{Y+1,X+1,Z}\rangle \\
&+ \left(\frac{Y'}{2^{h_1}} - Y\right) \left(\frac{Z'}{2^{l_1}} - Z\right) \left(\frac{X'}{2^{w_1}} - X\right) |S_{Y,X+1,Z}\rangle. \\
&= \left\{ [2^{h_1} - (Y' - 2^{h_1}Y)][2^{l_1} - (Z' - 2^{l_1}Z)][2^{w_1} - (X' - 2^{w_1}X)] |S_{Y+1,X,Z+1}\rangle \right. \\
&+ (Y' - 2^{h_1}Y)[2^{l_1} - (Z' - 2^{l_1}Z)][2^{w_1} - (X' - 2^{w_1}X)] |S_{Y,X,Z+1}\rangle \\
&+ [2^{h_1} - (Y' - 2^{h_1}Y)](Z' - 2^{l_1}Z)[2^{w_1} - (X' - 2^{w_1}X)] |S_{Y+1,X,Z}\rangle \\
&+ (Y' - 2^{h_1}Y)(Z' - 2^{l_1}Z)[2^{w_1} - (X' - 2^{w_1}X)] |S_{Y,X,Z}\rangle \\
&+ [2^{h_1} - (Y' - 2^{h_1}Y)][2^{l_1} - (Z' - 2^{l_1}Z)](X' - 2^{w_1}X) |S_{Y+1,X+1,Z+1}\rangle \\
&+ (Y' - 2^{h_1}Y)[2^{l_1} - (Z' - 2^{l_1}Z)](X' - 2^{w_1}X) |S_{Y,X+1,Z+1}\rangle \\
&+ [2^{h_1} - (Y' - 2^{h_1}Y)](Z' - 2^{l_1}Z)(X' - 2^{w_1}X) |S_{Y+1,X+1,Z}\rangle \\
&\left. + (Y' - 2^{h_1}Y)(Z' - 2^{l_1}Z)(X' - 2^{w_1}X) |S_{Y,X+1,Z}\rangle \right\} \div 2^{h_1} \div 2^{w_1} \div 2^{l_1}.
\end{aligned} \tag{13}$$

According to (11), $(Y' - 2^{h_1}Y)$ and $(X' - 2^{w_1}X)$ and $(Z' - 2^{l_1}Z)$ in (13) are the remainder of $Y'/2^{h_1}$ and $X'/2^{w_1}$ and $Z'/2^{l_1}$, respectively. From (13), in order to prepare the floating-point data $|S_{Y',X',Z'}\rangle$ in position (Y', X', Z') of the resulting data, the floating-point data $|S_{Y+1,X,Z+1}\rangle, |S_{Y+1,X,Z}\rangle, |S_{Y,X,Z+1}\rangle, |S_{Y,X,Z}\rangle, |S_{Y+1,X+1,Z+1}\rangle, |S_{Y,X+1,Z+1}\rangle$ and $|S_{Y,X+1,Z}\rangle$ in positions $(Y+1, X, Z+1), (Y+1, X, Z), (Y, X, Z+1), (Y, X, Z), (Y+1, X+1, Z+1), (Y+1, X+1, Z), (Y, X+1, Z+1)$ and $(Y, X+1, Z)$ of the original data need to be prepared first. Trilinear interpolation method utilizes these eight different positions of the original position (Y, X, Z) to map into one position (Y', X', Z') of the resulting data as shown in Fig. 15. The size of an array of given original data is known. At the same time, the size of the resulting data is known under a certain scaling ratio. Therefore, (Y', X', Z') can be considered as the input state when designing quantum circuit. Figs. 17 and 18 provide scaling-up circuit of 3-D floating-point data based on QFT using trilinear interpolation method.

Next, we explain the workflow of this circuit.

The inputs of this circuit are eight identical original data (denoted by $|S_{Y+1,X,Z+1}\rangle, |S_{Y+1,X,Z}\rangle, |S_{Y,X,Z+1}\rangle, |S_{Y,X,Z}\rangle, |S_{Y+1,X+1,Z+1}\rangle, |S_{Y+1,X+1,Z}\rangle, |S_{Y,X+1,Z+1}\rangle$ and $|S_{Y,X+1,Z}\rangle$) and the data positions in the scaled-up data denoted by $(Y+1, X, Z+1), (Y+1, X, Z), (Y, X, Z+1), (Y, X, Z), (Y+1, X+1, Z+1), (Y+1, X+1, Z), (Y, X+1, Z+1)$ and $(Y, X+1, Z)$, where the subscript indicates the number of qubits. The output of this circuit is the scaled-up data denoted by $|S_{Y',X',Z'}\rangle$.

Firstly, eight quantum oracle operators $\Omega_{Y+1,X,Z+1}, \Omega_{Y+1,X,Z}, \Omega_{Y,X,Z+1}, \Omega_{Y,X,Z}, \Omega_{Y+1,X+1,Z+1}, \Omega_{Y+1,X+1,Z}, \Omega_{Y,X+1,Z+1}$ and $\Omega_{Y,X+1,Z}$ are used to compute the original data values of $|S_{Y+1,X,Z+1}\rangle, |S_{Y+1,X,Z}\rangle, |S_{Y,X,Z+1}\rangle, |S_{Y,X,Z}\rangle, |S_{Y+1,X+1,Z+1}\rangle, |S_{Y+1,X+1,Z}\rangle, |S_{Y,X+1,Z+1}\rangle$ and $|S_{Y,X+1,Z}\rangle$, respectively. A quantum oracle operator $\Omega_{Y,X,Z}$ can realize the aim of assigning floating-point data $|S_{Y,X,Z}\rangle$ to the ancillary qubits $|0\rangle^{\otimes(p+q)}$, which can be expressed by (14)

$$\Omega_{Y,X,Z} = \bigotimes_{i=0}^{p+q-1} \Omega_{Y,X,Z}^i, \tag{14}$$

the function of $\Omega_{Y,X,Z}^i$ is setting the value of the i th qubit of the data in (Y, X, Z) :

$$\Omega_{Y,X,Z}^i : |0\rangle \rightarrow |0 \oplus s_{Y,X,Z}^i\rangle, \tag{15}$$

where $\oplus$ is the XOR operation, $s_{Y,X,Z}^i \in \{0, 1\}$. $\Omega_{Y,X,Z}$ are used to perform $(p+q)$ XOR gate on the $(p+q)|0\rangle$ to obtain the data, as shown in following equation:

Figure 16. Schematic representation of the scaling-up circuit, where $r_y = 4, r_x = 2, r_z = 2$.

$$\Omega_{Y,X,Z}|0\rangle^{\otimes(p+q)} = \bigotimes_{i=0}^{p+q-1} (\Omega_{Y,X,Z}^i|0\rangle) = \bigotimes_{i=0}^{p+q-1} |0 \oplus s_{Y,X,Z}^i\rangle = \bigotimes_{i=0}^{p+q-1} |s_{Y,X,Z}^i\rangle = |S_{Y,X,Z}\rangle_F. \quad (16)$$

Then, three *Special Subtraction* modules are used to obtain $2^{w_1} - x', 2^{h_1} - y', 2^{l_1} - z'$, where $x' = X' - 2^{w_1}X, y' = Y' - 2^{h_1}Y, z' = Z' - 2^{l_1}Z$, respectively. Further, eight groups of intermediate results $(2^{h_1} - y')(2^{l_1} - z')(2^{w_1} - x')|S_{Y+1,X,Z+1}\rangle, y'(2^{h_1} - z')(2^{w_1} - x')|S_{Y,X,Z+1}\rangle, z'(2^{h_1} - y')(2^{w_1} - x')|S_{Y+1,X,Z}\rangle, y'z'(2^{w_1} - x')|S_{Y,X,Z}\rangle, x'(2^{h_1} - y')(2^{l_1} - z')|S_{Y+1,X+1,Z+1}\rangle, y'x'(2^{h_1} - z')|S_{Y,X+1,Z+1}\rangle, z'x'(2^{h_1} - y')|S_{Y+1,X+1,Z}\rangle, y'z'x'|S_{Y,X+1,Z}\rangle$ are obtained. Here, we need to point out that the multiplication between the first three items in each group is the fixed-point multiplication based on QFT (*Multiplier* module, given by Fig. 9). Since the *Multiplier* module requires the qubits to be the same, we obtain the final number of qubits by seeking the maximum value. The number of qubits after multiplying the first three items in each group is $n_1 = \max\{h_1, w_1 + 1, l_1\}, n_2 = \max\{h_1 + 1, w_1 + 1, l_1\}, n_3 = \max\{h_1, w_1 + 1, l_1 + 1\}, n_4 = \max\{h_1 + 1, w_1 + 1, l_1 + 1\}, n_5 = \max\{h_1, w_1, l_1\}, n_6 = \max\{h_1 + 1, w_1, l_1\}, n_7 = \max\{h_1, w_1, l_1 + 1\}, n_8 = \max\{h_1 + 1, w_1, l_1 + 1\}$, respectively. Before multiplying with the fourth item floating-point data, the result of the previous fixed-point multiplication needs to be converted into the floating-point data of the same qubit through the *Converter* module, given by Fig. 14. Finally, the multiplication here uses the floating-point multiplier based on QFT (*Q-Multiplier* module, given by Fig. 13). For convenience, we omit the process of seeking the maximum value in the *Multiplier* module and the *Converter* module before the *Q-Multiplier* module, refer to Figs. 17 and 18.

Finally, 3-D floating-point data $|S_{Y',X',Z'}\rangle$ of the scaled-up is obtained by using seven *Q-Adder* modules (given by Fig. 12) and $h_1 + w_1 + l_1$ *Divided by 2* modules (given by Fig. 11). So far, we have completed the image scaling-up operation based on trilinear interpolation method. The scaling-up circuit of 3-D floating-point data is given by Figs. 17 and 18. For convenience, other unremarked qubits are the garbage outputs. Therefore, Figs. 17 and 18 are simplified scaling-up circuit, where ancillary and garbage outputs are omitted.

For convenience, we show the schematic representation of the scaling-up circuit as Fig. 16, where the small cube represents a voxel point of the 3-D floating-point data, the right side is the scaled-up data representation, $r_y = 4, r_x = 2, r_z = 2$.

Quantum scaling-down circuit for 3-D floating-point data. Assume that there is an array of $2^{h+h_1} \times 2^{w+w_1} \times 2^{l+l_1}$ 3-D floating-point data and the size of the resulting data after being scaled is $2^h \times 2^w \times 2^l$, i.e., $r_y = 2^{-h_1}, r_x = 2^{-w_1}$ and $r_z = 2^{-l_1}$ (where r_y, r_x and r_z represent the scaling in y-coordinate, x-coordinate, z-coordinate axis directions, respectively). Therefore, the scaling-down value for 3-D floating-point data in position (Y', X', Z') of the interpolated data using trilinear interpolation can be calculated as follows:

Figure 17. Scaling-up circuit(1).

Figure 18. Scaling-up circuit(2).

Figure 19. Scaling-down circuit(1).

Figure 20. Scaling-down circuit(2).

$$\begin{aligned}
|S_{Y',X',Z'}\rangle = & \left\{ [2^{h_1} - (Y - 2^{h_1} Y')][2^{l_1} - (Z - 2^{l_1} Z')][2^{w_1} - (X - 2^{w_1} X')]|S_{Y+1,X,Z+1}\rangle \right. \\
& + (Y - 2^{h_1} Y')[2^{l_1} - (Z - 2^{l_1} Z')][2^{w_1} - (X - 2^{w_1} X')]|S_{Y,X,Z+1}\rangle \\
& + [2^{h_1} - (Y - 2^{h_1} Y')(Z - 2^{l_1} Z')][2^{w_1} - (X - 2^{w_1} X')]|S_{Y+1,X,Z}\rangle \\
& + (Y - 2^{h_1} Y')(Z - 2^{l_1} Z')[2^{w_1} - (X - 2^{w_1} X')]|S_{Y,X,Z}\rangle \\
& + [2^{h_1} - (Y - 2^{h_1} Y')][2^{l_1} - (Z - 2^{l_1} Z')(X - 2^{w_1} X')]|S_{Y+1,X+1,Z+1}\rangle \\
& + (Y - 2^{h_1} Y')[2^{l_1} - (Z - 2^{l_1} Z')(X - 2^{w_1} X')]|S_{Y,X+1,Z+1}\rangle \\
& + [2^{h_1} - (Y - 2^{h_1} Y')(Z - 2^{l_1} Z')(X - 2^{w_1} X')]|S_{Y+1,X+1,Z}\rangle \\
& \left. + (Y - 2^{h_1} Y')(Z - 2^{l_1} Z')(X - 2^{w_1} X')|S_{Y,X+1,Z}\rangle \right\} \div 2^{h_1} \div 2^{w_1} \div 2^{l_1}.
\end{aligned} \quad (17)$$

Figures 19 and 20 provide scaling-down circuit of 3-D floating-point data based on QFT using trilinear interpolation method.

Next, we explain the workflow of this circuit.

Firstly, eight quantum oracle operators $\Omega_{Y+1,X,Z+1}$, $\Omega_{Y+1,X,Z}$, $\Omega_{Y,X,Z+1}$, $\Omega_{Y,X,Z}$, $\Omega_{Y+1,X+1,Z+1}$, $\Omega_{Y+1,X+1,Z}$, $\Omega_{Y,X+1,Z+1}$ and $\Omega_{Y,X+1,Z}$ are used to compute the original data values of $|S_{Y+1,X,Z+1}\rangle$, $|S_{Y+1,X,Z}\rangle$, $|S_{Y,X,Z+1}\rangle$, $|S_{Y,X,Z}\rangle$, $|S_{Y+1,X+1,Z+1}\rangle$, $|S_{Y+1,X+1,Z}\rangle$, $|S_{Y,X+1,Z+1}\rangle$ and $|S_{Y,X+1,Z}\rangle$, respectively.

Then, *adding one*, *Special Subtractor*, *Multiplier*, *Converter* and *Q-Multiplier* modules are used to design the scaling-down circuit of 3-D floating-point data.

Finally, 3-D floating-point data $|S_{Y',X',Z'}\rangle$ of the scaled-down is obtained using seven *Q-Adder* modules and $(h_1 + w_1 + l_1)$ *Divided by 2* modules. So far, we have completed the image scaling-down operation using trilinear interpolation method. For convenience, other unremarked qubits are the garbage outputs. Therefore, Figs. 19 and 20 are simplified scaling-down circuit, where ancillary and garbage outputs are omitted.

Complexity analysis. The circuit network complexity depends on the number of elementary gate in quantum image processing (QIMP). The complexity of the basic quantum gate is considered to be one, including NOT gate, Control-Not gate and any 2×2 unitary operator³¹.

The network complexities of *adding one*, *QFT*, *IQFT*, *rotation*, *adder* and *subtractor* modules are all $O(n^2)^{29,31}$. The *Multiplier* module consists of 1 *QFT*, $(n-1)$ *Multiply by 2*, n *Rotation*, and 1 *IQFT*, and so, the complexity of *Multiplier* module is $O(n^3)^{29}$. For the *Special Subtractor* module, Fig. 10 shows that it consists of $(m+1)$ CNOT gates and 1 *adding one* module, and so, the complexity of *Special Subtractor* module is $O(m^2)$. An m -controlled NOT gate in the *divided by 2* module can be decomposed into $2(m-1)$ Toffoli gates and 1 CNOT gate and the Toffoli gate can be approximately simulated by 6 CNOT gates³¹, so the complexity of it is $O(12m-11)$. Hence, the complexity of preparing the *divided by 2* module is $O(p^2)$. In (16), if $s_{Y,X,Z}^i = 1$, $\Omega_{Y,X,Z}^i$ is a $(h+w+l)$ -controlled NOT qubit gate. Otherwise, it is a quantum identity gate. That is to say, every oracle operator $\Omega_{Y,X,Z}^i$, $i = 0, \dots, p+q-1$, is at most a $(h+w+l)$ -controlled NOT qubit gate. For other oracle operators $\Omega_{Y+1,X,Z+1}$, $\Omega_{Y+1,X,Z}$, $\Omega_{Y,X,Z+1}$, $\Omega_{Y,X,Z}$, $\Omega_{Y+1,X+1,Z+1}$, $\Omega_{Y+1,X+1,Z}$, $\Omega_{Y,X+1,Z+1}$ and $\Omega_{Y,X+1,Z}$, the principle is also same as $\Omega_{Y,X,Z}$. The complexity of oracle operator $\Omega_{Y,X,Z}$ is $O((p+q)(h+w+l))$. The scaling-up circuit of 3-D floating-point data includes eight oracle operators of $\Omega_{Y+1,X,Z+1}$, $\Omega_{Y+1,X,Z}$, $\Omega_{Y,X,Z+1}$, $\Omega_{Y,X,Z}$, $\Omega_{Y+1,X+1,Z+1}$, $\Omega_{Y+1,X+1,Z}$, $\Omega_{Y,X+1,Z+1}$ and $\Omega_{Y,X+1,Z}$. Here, the total quantum cost in eight oracle operators is $O(8(p+q)(h+w+l))$. The network complexities of *Q-Adder* and *Q-Multiplier* modules are $O(p^2 + q^2)$ and $O(p^3 + q^2)$ based on^{31,33,34}, respectively. According to Fig. 14, the *Converter* module can be decomposed into $(2^{p-1} - 1)$ *adding one*, $(n-k)$ k -Control-adding-one, $k = 1, 2, \dots, n-1$, $(n-1)$ Toffoli gates, $(n-m+1)$ m -controlled NOT gates, $m = 3, 4, \dots, n$. The quantum cost of $(n-m+1)$ m -controlled NOT gates ($m = 3, 4, \dots, n$) is

$$\begin{aligned}
& (n-3+1)(12 \times 3 - 11) + (n-4+1)(12 \times 4 - 11) + \dots + (n-n+1)(12 \times n - 11) \\
& = 2n^3 + 0.5n^2 - 4.5n - 9.
\end{aligned}$$

Only $(4k-8)$ 2-Control-Unitary gates were needed to construct 1 k -Control-Unitary gate, as well as some assistant qubits³⁴. Therefore, 1 k -Control-adding-one can be constructed by $(4k-8)$ 2-Control-adding-one. The quantum cost of $(n-k)$ k -Control-adding-one ($k = 1, 2, \dots, n-1$) is

$$\begin{aligned}
& (n-1)(4 \times 1 - 8)p^2 + (n-2)(4 \times 2 - 8)p^2 + \dots + [n - (n-1)][4 \times (n-1) - 8]p^2 \\
& = p^2 \left(\frac{2}{3}n^3 - 4n^2 + \frac{10}{3}n \right).
\end{aligned}$$

To sum up, the complexity of the *Converter* module is $O((2^{p-1}-1)p^2 + 2n^3 + 0.5n^2 - 4.5n - 9 + 6(n-1)p^2(\frac{2}{3}n^3 - 4n^2 + \frac{10}{3}n)) \approx O(2^{p-1}p^2 + n^3p^2)$.

Case 1: The complexity of the scaling-up circuit.

We analyze the complexity of the scaling-up circuit of 3-D floating-point data. The quantum circuit consists of 12 *Special Subtractor*, 8 *adding one*, 16 *Multiplier*, 8 *Q-Multiplier*, 7 *Q-Adder*, $8(h+w+l+h_1+w_1+l_1)$ CNOT gates, 8 oracle operators, 8 *Converter*, and $(h_1 + w_1 + l_1)$ *Divided by 2*.

Therefore, the complexity of the scaling-up circuit of 3-D floating-point data can be calculated as follows:

$$O \left(\begin{array}{c} 4(h_1^2 + w_1^2 + l_1^2) \\ 4h^2 + w^2 + 3l^2 \\ 2(n_1^3 + n_2^3 + n_3^3 + n_4^3 + n_5^3 + n_6^3 + n_7^3 + n_8^3) \\ 8(p^3 + q^3) \\ 7(p^2 + q^2) \\ 8(h + w + l + h_1 + w_1 + l_1) \\ 8(p + q)(h + w + l) \\ 8p^2 2^{p-1} + (n_1^3 + n_2^3 + n_3^3 + n_4^3 + n_5^3 + n_6^3 + n_7^3 + n_8^3)p^2 \\ p^2(h_1 + w_1 + l_1) \end{array} \right) \approx O(p^2 \alpha^3 + p^3 + \beta^2 + q^2),$$

where $\alpha = \max(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8)$, $\beta = \max(h, w, l)$.

Case 2: The complexity of the scaling-down circuit.

Similar to the complexity of the scaling-up circuit, we analyze the complexity of the scaling-down circuit of 3-D floating-point data. The quantum circuit consists of 12 *Special Subtractor*, 12 *adding one*, 16 *Multiplier*, 8 *Q-Multiplier*, 7 *Q-Adder*, $8(h + w + l + h_1 + w_1 + l_1)$ CNOT gates, 8 oracle operators, 8 *Converter*, and $(h_1 + w_1 + l_1)$ *Divided by 2*.

Therefore, the complexity of the scaling-down circuit of 3-D floating-point data can be calculated as follows:

$$O \left(\begin{array}{c} 4(h_1^2 + w_1^2 + l_1^2) \\ 4(h^2 + w^2 + l^2) \\ 2(n_1^3 + n_2^3 + n_3^3 + n_4^3 + n_5^3 + n_6^3 + n_7^3 + n_8^3) \\ 8(p^3 + q^3) \\ 7(p^2 + q^2) \\ 8(h + w + l + h_1 + w_1 + l_1) \\ 8(p + q)(h + w + l) \\ 8p^2 2^{p-1} + (n_1^3 + n_2^3 + n_3^3 + n_4^3 + n_5^3 + n_6^3 + n_7^3 + n_8^3)p^2 \\ p^2(h_1 + w_1 + l_1) \end{array} \right) \approx O(p^2 \alpha^3 + p^3 + \beta^2 + q^2),$$

where $\alpha = \max(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8)$, $\beta = \max(h, w, l)$.

Conclusions

Quantum computation has become a novel and important tool in the field of image/data processing. In this paper, the trilinear interpolation method for quantum scaling up and down of 3-D floating-point data is proposed for the first time. 3-D data have a wide range of applications in many diverse fields such as artificial intelligence, aeronautics, architecture, biological science, medicine, etc. Floating-point numbers offer great savings in the number of qubits when the required range of values and/or relative precision is large. Therefore, based on QFT, we have designed the addition and multiplication (Q-Adder and Q-Multiplier modules) of 3-D floating-point data. And then, we have proposed a Converter module for converting fixed-point numbers to floating-point numbers. Combining some basic modules in this paper, we can achieve the quantum scaling up and down for 3-D floating-point data. Finally, we have proposed the design scheme of quantum scaling up and down for 3-D floating-point data using trilinear interpolation method based on QFT.

In future work, the design scheme of quantum scaling up and down is extremely helpful for quantum reconstruction theory, such as the reconstruction technology of 2-D to 3-D data. And we can use the reconstruction technology to perform quantum data processing such as medical data reconstruction in future research.

Received: 28 July 2021; Accepted: 4 February 2022

Published online: 17 February 2022

References

1. Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys.* **21**(6–7), 467–488 (1982).
2. Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science* 124–134 (1994).
3. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing* 212–219 (1996).
4. Venegas-Andraca, S. E. & Bose, S. Storing, processing and retrieving an image using quantum mechanics. In *Proceeding of SPIE Conference of Quantum Information and Computation* 5105 137–147 (2003).
5. Latorre, J. I. Image compression and entanglement. *Computer. Science* 1–4 (2005).
6. Venegas-Andraca, S. E. & Ball, J. L. Processing images in entangled quantum systems. *Quantum Inf. Process.* **9**(1), 1–11 (2010).
7. Le, P. Q., Dong, F. & Hirota, K. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Inf. Process.* **10**(1), 63–84 (2011).
8. Zhang, Y., Lu, K., Gao, Y. & Wang, M. NEQR: a novel enhanced quantum representation of digital images. *Quantum Inf. Process.* **12**(8), 2833–2860 (2013).
9. Jiang, N. & Wang, L. Quantum image scaling using nearest neighbor interpolation. *Quantum Inf. Process.* **14**(5), 1559–1571 (2015).
10. Jiang, N., Wang, J. & Mu, Y. Quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio. *Quantum Inf. Process.* **14**(11), 4001–4026 (2015).
11. Li, H., Fan, P., Xia, H., Peng, H. & Song, S. Quantum implementation circuits of quantum signal representation and type conversion. *IEEE Trans. Circuits Syst. I Regul. Pap.* **66**(1), 341–354 (2019).
12. Zhang, R., Lu, D. & Yin, H. A generalized floating-point representation and manipulation of quantum signals based on IEEE-754. *Int. J. Theor. Phys.* **59**, 936–952 (2020).
13. Zhang, R., Xu, M. & Lu, D. A generalized floating-point quantum representation of 2-D data and their applications. *Quantum Inf. Process.* **19**(11), 390 (2020).

14. Chetia, R., Boruah, S. M. B. & Sahu, P. P. Quantum image edge detection using improved Sobel mask based on NEQR. *Quantum Inf. Process.* **20**(1), 21 (2021).
15. Chakraborty, S., Mandal, S. B. & Shaikh, S. H. Quantum image processing: challenges and future research issues. *Int. J. Inf. Technol.* 1–15 (2018).
16. S. Chakraborty, S. B. Mandal, S. H. Shaikh, and L. Dey. Ternary quantum circuit for color image representation. In *Advanced Computing and Systems for Security* 95–108 (2017).
17. Chakraborty, S., Mandal, S. B. & Shaikh, S. H. Design and implementation of a multivalued quantum circuit for threshold based color image segmentation. *Intell. Decis. Technol.* **12**(2), 251–264 (2018).
18. Fijany, A. & Williams, C. P. Quantum wavelet transforms: fast algorithms and complete circuits. In *NASA international conference on quantum computing and quantum communications* 10–33 (1998).
19. Caraiman, S. & Manta, V. Quantum image filtering in the frequency domain. *Adv. Electr. Comput. Eng.* **13**(3), 77–84 (2013).
20. Ruiz-Perez, L. & Garcia-Escartin, J. C. Quantum arithmetic with the quantum Fourier transform. *Quantum Inf. Process.* **16**(6), 152 (2017).
21. Li, P. & Sun, H. Quantum color image filtering in the frequency domain. *J. Electron. Inf. Technol.* **40**(3), 594–601 (2018).
22. Asaka, R., Sakai, K. & Yahagi, R. Quantum circuit for the fast Fourier transform. *Quantum Inf. Process.* **19**(8), 277 (2020).
23. Chakraborty, S., Shaikh, S. H., Chakrabarti, A. & Ghosh, R. An image denoising technique using quantum wavelet transform. *Int. J. Theor. Phys.* **59**(11), 3348–3371 (2020).
24. Chakraborty, S., Shaikh, S. H., Chakrabarti, A. & Ghosh, R. A study of scrambled noisy quantum image formation with geometric transformation and its denoising using QWT. In *High Performance Vision Intelligence: Recent Advances* 137–150 (2020).
25. Chang, W. L. & Vasilakos, A. V. *Fundamentals of Quantum Programming in IBM's Quantum Computers* (Springer, Berlin, 2021).
26. Grigoryan, A. M. & Agaian, S. S. New look on quantum representation of images: Fourier transform representation. *Quantum Inf. Process.* **19**(5), 148 (2020).
27. Sang, J., Wang, S. & Niu, X. Quantum realization of the nearest-neighbor interpolation method for FRQI and NEQR. *Quantum Inf. Process.* **15**(1), 37–64 (2016).
28. Zhou, R., Hu, W., Fan, P. & Ian, H. Quantum realization of the bilinear interpolation method for NEQR. *Sci. Rep.* **7**(1), 2511 (2017).
29. Li, P. & Liu, X. Bilinear interpolation method for quantum images based on quantum Fourier transform. *Int. J. Quantum Inf.* **16**(4), 1850031 (2018).
30. Zhou, R., Cheng, Y. & Liu, D. Quantum image scaling based on bilinear interpolation with arbitrary scaling ratio. *Quantum Inf. Process.* **18**, 267 (2019).
31. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge University Press, New York, 2000).
32. Zhang, Y., Lu, K., Xu, K., Gao, Y. & Wilson, R. Local feature point extraction for quantum images. *Quantum Inf. Process.* **14**(5), 1573–1588 (2015).
33. Haener, T., Soeken, M., Roetteler, M. & Svore, K. M. Quantum circuits for floating-point arithmetic. *Lect. Notes Comput. Sci.* 162–174 (2018).
34. Barenco, A. *et al.* Elementary gates for quantum computation. *Phys. Rev. A* **52**(5), 3457–3467 (1995).

Acknowledgements

The authors express their gratitude to the anonymous referees for their kind suggestions and useful comments on the original manuscript, which resulted in this final version. This work is supported by the National Natural Science Foundation of China (Grant No. 41771375).

Author contributions

D.Y. L. and M.Y. X. give the core theme of this paper and main guideline on the paper writing stage. M.Y. X. is the main writer of the manuscript paper. X.Y. S. is the main reviser of the manuscript, making a lot of changes to the manuscript and language correcting works. In addition, many figures are drawn by M.Y. X. and D.Y. L. Finally, M.Y. X., D.Y. L. and X.Y. S. reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to D.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2022