Scaling Up and Down of 3-D Floating-point Data in Quantum Computation

Meiyu Xu

School of Mathematics and Statistics, Henan University of Science and Technology

Dayong Lu (✉ dayonglu@163.com)

School of Mathematics and Statistics, Henan University

Xiaoyun Sun

College of Information and Management Science, Henan Agricultural University

Research Article

Keywords: Quantum computation, Floating-point data, Quantum Fourier transform, Trilinear interpolation method, Quantum circuits

Posted Date: August 4th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-757209/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at Scientific Reports on February 17th, 2022. See the published version at <https://doi.org/10.1038/s41598-022-06756-w>.

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

Meiyu Xu^a, Dayong Lu^b, Xiaoyun Sun^c

^a School of Mathematics and Statistics, Henan University of Science and Technology, Luoyang 471000, China

^b School of Mathematics and Statistics, Henan University, Kaifeng 475001, China

^c College of Information and Management Science, Henan Agricultural University, Zhengzhou 450000, China

Abstract 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 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 compared to 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 or 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 numbers in 3-D space to achieve quantum algorithms of scaling up and down for 3-D floating-point data. Finally, the circuits of quantum scaling up and down for 3-D floating-point data are designed.

Key words : Quantum computation · Floating-point data · Quantum Fourier transform · Trilinear interpolation method · Quantum circuits

1 Introduction

Quantum computation is a theoretical computation system that performs operations on data by using quantum-mechanical phenomena (such as superposition and entanglement). In 1982, Feynman [1] proposed the concept of quantum computer at first. After that the integer factoring problem [2] and database search algorithm [3] were essential evidences supporting the power of quantum computer. Researches in quantum image processing field started with proposals on quantum image representations such as Qubit Lattice [4], Real Ket [5] and Entangled Image [6]. After 2011, some more practical quantum image models using entangled states and superposition states have been proposed including FRQI [7], NEQR [8], INEQR [9], GQIR [10], QRDS [11], GFPRQS [12], QR2-DD [13], etc.

At present, quantum image processing research has been paid more and more attention in transform domain. In [14], Fijany and Williams studied fast algorithms and complete circuits for quantum wavelet transforms. In [15], Caraiman and Manta introduced quantum image filtering in the frequency domain. In [16], quantum arithmetic operations were given by Ruiz-Perez and Garcia-Escartin based on quantum Fourier transform. In [17], Li and Sun presented quantum color image filtering in the frequency domain. There are many research results in transform domain, please refer to [18–21].

In recent years, quantum image scaling as a kind of image geometric transformation has been widely studied and applied in quantum image processing. In [9, 10], Jiang et al. proposed quantum image scaling using nearest-neighbor interpolation based on INEQR and GQIR, respectively. In [22], Sang et al. constructed the quantum circuits of the nearest-neighbor interpolation method for FRQI and NEQR. In [23], Zhou et al. proposed the bilinear interpolation method for NEQR and given the corresponding quantum realization circuits. In [24], Li and Liu designed quantum image scaling

using bilinear interpolation method based on QFT. In [25], Zhou et al. given quantum image scaling based on bilinear interpolation with arbitrary scaling ratio. However, 3-dimensional (3-D) quantum floating-point data version of which do not exist.

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), refer to [13].

Compared to fixed-point numbers, floating-point numbers offer great savings in number of qubits when the required range of values and/or relative precision is large. Therefore, finding good 3-D floating-point data scaling operations could be of tremendous use in many applications of quantum computing. In this paper, we proposed scaling up and down of 3-D floating-point data in quantum computation. In order to achieve this scaling of 3-D floating-point data, 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. Finally, we introduced the design scheme of quantum scaling up and down for 3-D floating-point data.

The remainder of this paper is organized as follows : Section 2 gives a brief introduction to quantum representation of 3-D floating-point numbers, classical interpolation methods and QFT. After the described of the basic modules in subsection 3.1, 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 Subsection 3.2. Subsections 3.3 and 3.4 introduce the design scheme of quantum scaling up and down for 3-D floating-point data using trilinear interpolation method. Finally, conclusions and future research works are described in Section 4.

2 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 (2.1). The identity gate (I), Hadamard gate (H), NOT gate (X), controlled-NOT gate ($CNOT$) and controlled-phase gate (CR_k) are well-known and can be found in the basic reference [26], here,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, CR_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix}. \quad (2.1)$$

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

s	e	f
<i>sign bit</i>	<i>exponent</i>	<i>fractional(mantissa)</i>

Table 1. *Format of floating-point numbers in IEEE-754*

2.1 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 the location of $|YXZ\rangle$. This proposal integrates information

about a $H \times W \times Z$ 3-D data in Figure 1 into a quantum state having it formula in (2.2) :

$$|D_3\rangle = \frac{1}{2^{\frac{h+w+l}{2}}} \left(\sum_{Y=0}^{H-1} \sum_{X=0}^{W-1} \sum_{Z=0}^{L-1} |S_{YXZ}\rangle_F \otimes |YXZ\rangle \right), \quad (2.2)$$

where

$$|YXZ\rangle = |Y\rangle|X\rangle|Z\rangle = |y_0y_1 \dots y_{h-1}\rangle |x_0x_1 \dots x_{w-1}\rangle |z_0z_1 \dots z_{l-1}\rangle, \quad y_i, x_i, z_i \in \{0, 1\},$$

$$|S_{YXZ}\rangle_F = \left| s_{YXZ}^0 s_{YXZ}^1 s_{YXZ}^2 \dots s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1} \right\rangle, \quad s_{YXZ}^i \in \{0, 1\},$$

$$h = \begin{cases} \lceil \log_2 H \rceil, & H > 1; \\ 1, & H = 1, \end{cases} \quad w = \begin{cases} \lceil \log_2 W \rceil, & W > 1; \\ 1, & W = 1, \end{cases} \quad l = \begin{cases} \lceil \log_2 L \rceil, & L > 1; \\ 1, & L = 1. \end{cases} \quad (2.3)$$

Here, $|YXZ\rangle$ is the location information, $|S_{YXZ}\rangle_F = |s_{YXZ}^0 s_{YXZ}^1 s_{YXZ}^2 \dots s_{YXZ}^p s_{YXZ}^{p+1} \dots s_{YXZ}^{p+q-1}\rangle$ 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.

Figure 1. 3-D data

1	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 $|S_{YXZ}\rangle_F$

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

2.2 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 Figure 2 :

Figure 2. The principle of interpolation in one direction

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

$$c = \frac{(x_1 - x)c_0 + (x - x_0)c_1}{x_1 - x_0}. \quad (2.4)$$

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 Figure 3. Here,

$$Y = \left\lfloor Y' \times \frac{H}{H'} \right\rfloor, \quad X = \left\lfloor X' \times \frac{W}{W'} \right\rfloor, \quad h = \frac{H}{H'}Y' - Y, \quad w = \frac{W}{W'}X' - X, \quad (2.5)$$

where $\lfloor \cdot \rfloor$ denotes rounding down.

Figure 3. Coordinate map relationship

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 :

$$\begin{aligned} \begin{cases} f(Y', X')_1 = (1 - w)f(Y, X) + wf(Y, X + 1) \\ f(Y', X')_2 = (1 - w)f(Y + 1, X) + wf(Y + 1, X + 1) \end{cases} \\ \Rightarrow f(Y', X') = (1 - h)f(Y', X')_1 + hf(Y', X')_2 \\ = (1 - w)(1 - h)f(Y, X) + w(1 - h)f(Y, X + 1) \\ + (1 - w)hf(Y + 1, X) + whf(Y + 1, X + 1). \end{aligned} \quad (2.6)$$

Hance, bilinear interpolation method is three times of single linear interpolation method in the x and y directions.

2.3 Quantum Fourier transform

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

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle, \quad (2.7)$$

where i is an imaginary unit. The specific quantum circuits are shown in Figure 4, where $|x\rangle = |x_0 x_1 \dots x_{n-1}\rangle$. For simplicity, in this figure we have omitted the sequence of SWAP gates needed to invert the order of the output qubits.

Figure 4. Quantum circuits that implement QFT. Not shown are swap gates at the end of the circuit which reverse the order of the qubits (refer to [26])

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

$$IQFT|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-\frac{2\pi i j k}{N}} |j\rangle. \quad (2.8)$$

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 $|x\rangle = |x_0 x_1 \dots x_{n-1}\rangle$ 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.

3 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.

3.1 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 integer in [27], and its quantum circuit is shown in Figure 5, where $|x\rangle = |x_0x_1\dots x_{n-1}\rangle$, n is a positive integer, $n \geq 1$, $x_0, x_1, \dots, x_{n-1} \in \{0, 1\}$.

Figure 5. *Adding one module (refer to [27])*

2. Rotation module

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

$$|x\rangle|y\rangle \xrightarrow{\text{Rotation}} e^{i\frac{2\pi xy}{2^{n+1}}} |x\rangle|y\rangle, \quad (3.9)$$

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

Figure 6. *Rotation module (refer to [26])*

3. Adder and Multiplier modules

In [24], Li 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 little 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 Figures 7, 8 and 9, respectively.

Figure 7. Adder module (refer to [24])

Figure 8. Subtractor module (refer to [24])

Figure 9. Multiplier module (refer to [24])

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 Figure 10.

Figure 10. Special Subtractor module (refer to [24])

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 $|s\rangle_F = |s^0 s^1 \dots s^p s^{p+1} \dots s^{p+q-1}\rangle_F$. The circuit for floating-point *divided by 2* module is depicted in Figure 11.

Figure 11. *Divided by 2* module (refer to [12])

3.2 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. In [26], Nielsen given the quantum circuit of the QFT. Adder and Multiplier modules based on floating-point number are given in [28]. We will benefit from the circuits given by [24, 28] 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 combine [24] with [28], and the quantum circuits are depicted in Figures 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 Figure 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 $|s\rangle_F = |s^s s^e s^f\rangle_F = |s^s s^1 s^2 \dots s^{p_1} s^{p_1+1} \dots s^{p_1+q_1-1}\rangle_F$, $|t\rangle_F = |t^s t^e t^f\rangle_F = |t^s t^1 t^2 \dots t^{p_2} t^{p_2+1} \dots t^{p_2+q_2-1}\rangle_F$.

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

The comparator module based on integer (CMP) is used to compares 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.

Figure 12. *Q-Adder module*

2. *Q-Multiplier module*

Figure 13 shows the multiplication of two floating-point numbers $|s\rangle_F$ and $|t\rangle_F$, where $|s\rangle_F = |s^s s^e s^f\rangle_F = |s^s s^1 s^2 \dots s^{p_1} s^{p_1+1} \dots s^{p_1+q_1-1}\rangle_F$, $|t\rangle_F = |t^s t^e t^f\rangle_F = |t^s t^1 t^2 \dots t^{p_2} t^{p_2+1} \dots t^{p_2+q_2-1}\rangle_F$. 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 overflow, right-shift by 1 and increment the result exponent.
- (4). The final step denoted by ${}^c\text{COPY}$ consists of conditionally copying out of the resulting exponent, mantissa and sign bit.

Figure 13. *Q-Multiplier module*

3. *Converter module*

In Figure 14, *Converter* module converts fixed-point numbers to floating-point numbers, where $|x\rangle = |x_0 x_1 \dots x_{n-1}\rangle$, $|x\rangle_F = |x_0' x_1' \dots x_{p+1}' \dots x_{p+q-1}'\rangle_F$. 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.

Figure 14. *Converter module*

3.3 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 (3.13).

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 (2.2) ($H = 2^h, W = 2^w, L = 2^l$) can be expressed as follows :

$$|D_3\rangle = \frac{1}{2^{\frac{h+w+l}{2}}} \left(\sum_{Y=0}^{2^h-1} \sum_{X=0}^{2^w-1} \sum_{Z=0}^{2^l-1} |S_{YXZ}\rangle_F \otimes |YXZ\rangle \right), \quad (3.10)$$

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). Trilinear interpolation method for 3-D floating-point data can be described within following two steps in detail.

Step 1 : Build the 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 $X + 1$ 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 yz plane is fixed. The corresponding relationship is shown in Figure 15. Therein,

$$Y = \left\lfloor \frac{Y'}{2^{h_1}} \right\rfloor, X = \left\lfloor \frac{X'}{2^{w_1}} \right\rfloor, Z = \left\lfloor \frac{Z'}{2^{l_1}} \right\rfloor, r = \frac{Y'}{2^{h_1}} - Y, s = \frac{X'}{2^{w_1}} - X, t = \frac{Z'}{2^{l_1}} - Z, \quad (3.11)$$

where $\lfloor \cdot \rfloor$ represents the rounding down operation, $|Y'\rangle = |Y_{h+h_1-1}Y_{h+h_1-2} \dots Y_1Y_0\rangle$, $|X'\rangle = |X_{w+w_1-1}X_{w+w_1-2} \dots X_1X_0\rangle$, $|Z'\rangle = |Z_{l+l_1-1}Z_{l+l_1-2} \dots Z_1Z_0\rangle$. To build the mapping relationship described in Figure 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 Figures 17 and 18.

Figure 15. Coordinate map relationship for 3-D

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

According to (2.6), 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 :

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

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

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

$$+ (1-r)ts|S_{Y+1,X+1,Z}\rangle + rts|S_{Y,X+1,Z}\rangle. \quad (3.12)$$

Therefore,

$$\begin{aligned} |S_{Y',X',Z'}\rangle &= [1 - (\frac{Y'}{2^{h_1}} - Y)][1 - (\frac{Z'}{2^{l_1}} - Z)][1 - (\frac{X'}{2^{w_1}} - X)]|S_{Y+1,X,Z+1}\rangle \\ &\quad + (\frac{Y'}{2^{h_1}} - Y)[1 - (\frac{Z'}{2^{l_1}} - Z)][1 - (\frac{X'}{2^{w_1}} - X)]|S_{Y,X,Z+1}\rangle \\ &\quad + [1 - (\frac{Y'}{2^{h_1}} - Y)](\frac{Z'}{2^{l_1}} - Z)[1 - (\frac{X'}{2^{w_1}} - X)]|S_{Y+1,X,Z}\rangle \\ &\quad + (\frac{Y'}{2^{h_1}} - Y)(\frac{Z'}{2^{l_1}} - Z)[1 - (\frac{X'}{2^{w_1}} - X)]|S_{Y,X,Z}\rangle \\ &\quad + [1 - (\frac{Y'}{2^{h_1}} - Y)][1 - (\frac{Z'}{2^{l_1}} - Z)](\frac{X'}{2^{w_1}} - X)|S_{Y+1,X+1,Z+1}\rangle \\ &\quad + (\frac{Y'}{2^{h_1}} - Y)[1 - (\frac{Z'}{2^{l_1}} - Z)](\frac{X'}{2^{w_1}} - X)|S_{Y,X+1,Z+1}\rangle \\ &\quad + [1 - (\frac{Y'}{2^{h_1}} - Y)](\frac{Z'}{2^{l_1}} - Z)(\frac{X'}{2^{w_1}} - X)|S_{Y+1,X+1,Z}\rangle \\ &\quad + (\frac{Y'}{2^{h_1}} - Y)(\frac{Z'}{2^{l_1}} - Z)(\frac{X'}{2^{w_1}} - X)|S_{Y,X+1,Z}\rangle. \\ &= \left\{ [2^{h_1} - (Y' - 2^{h_1}Y)][2^{l_1} - (Z' - 2^{l_1}Z)][2^{w_1} - (X' - 2^{w_1}X)]|S_{Y+1,X,Z+1}\rangle \right. \\ &\quad + (Y' - 2^{h_1}Y)[2^{l_1} - (Z' - 2^{l_1}Z)][2^{w_1} - (X' - 2^{w_1}X)]|S_{Y,X,Z+1}\rangle \\ &\quad + [2^{h_1} - (Y' - 2^{h_1}Y)](Z' - 2^{l_1}Z)[2^{w_1} - (X' - 2^{w_1}X)]|S_{Y+1,X,Z}\rangle \\ &\quad + (Y' - 2^{h_1}Y)(Z' - 2^{l_1}Z)[2^{w_1} - (X' - 2^{w_1}X)]|S_{Y,X,Z}\rangle \\ &\quad + [2^{h_1} - (Y' - 2^{h_1}Y)][2^{l_1} - (Z' - 2^{l_1}Z)](X' - 2^{w_1}X)|S_{Y+1,X+1,Z+1}\rangle \\ &\quad + (Y' - 2^{h_1}Y)[2^{l_1} - (Z' - 2^{l_1}Z)](X' - 2^{w_1}X)|S_{Y,X+1,Z+1}\rangle \\ &\quad + [2^{h_1} - (Y' - 2^{h_1}Y)](Z' - 2^{l_1}Z)(X' - 2^{w_1}X)|S_{Y+1,X+1,Z}\rangle \\ &\quad \left. + (Y' - 2^{h_1}Y)(Z' - 2^{l_1}Z)(X' - 2^{w_1}X)|S_{Y,X+1,Z}\rangle \right\} \div 2^{h_1} \div 2^{w_1} \div 2^{l_1}. \quad (3.13) \end{aligned}$$

According to (3.11), $(Y' - 2^{h_1}Y)$ and $(X' - 2^{w_1}X)$ and $(Z' - 2^{l_1}Z)$ in (3.13) are the remainder of $Y'/2^{h_1}$ and $X'/2^{w_1}$ and $Z'/2^{l_1}$, respectively. From (3.13), 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 $|S_{Y+1,X,Z+1}\rangle$, $|S_{Y+1,X,Z}\rangle$, $|S_{Y,X,Z+1}\rangle$, $|S_{Y,X,Z}\rangle$, $|S_{Y+1,X+1,Z+1}\rangle$, $|S_{Y+1,X+1,Z}\rangle$, $|S_{Y,X+1,Z+1}\rangle$ 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 Figure 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. Figures 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 (3.14)

$$\Omega_{Y,X,Z} = \bigotimes_{i=0}^{p+q-1} \Omega_{Y,X,Z}^i, \quad (3.14)$$

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

$$\Omega_{Y,X,Z}^i : |0\rangle \rightarrow |0 \oplus s_{Y,X,Z}^i\rangle, \quad (3.15)$$

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 :

$$\Omega_{Y,X,Z} |0\rangle^{\otimes(p+q)} = \bigotimes_{i=0}^{p+q-1} (\Omega_{Y,X,Z}^i |0\rangle) = \bigotimes_{i=0}^{p+q-1} |0 \oplus s_{Y,X,Z}^i\rangle = \bigotimes_{i=0}^{p+q-1} |s_{Y,X,Z}^i\rangle = |S_{Y,X,Z}\rangle_F. \quad (3.16)$$

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 $(2^{h_1} - y')(2^{l_1} - z')(2^{w_1} - x')|S_{Y+1,X,Z+1}\rangle$, $y'(2^{l_1} - z')(2^{w_1} - x')|S_{Y,X,Z+1}\rangle$, $z'(2^{h_1} - y')(2^{w_1} - x')|S_{Y+1,X,Z}\rangle$, $y'z'(2^{w_1} - x')|S_{Y,X,Z}\rangle$, $x'(2^{h_1} - y')(2^{l_1} - z')|S_{Y+1,X+1,Z+1}\rangle$, $y'x'(2^{l_1} - z')|S_{Y,X+1,Z+1}\rangle$, $z'x'(2^{h_1} - y')|S_{Y+1,X+1,Z}\rangle$, $y'z'x'|S_{Y,X+1,Z}\rangle$ 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 Figure 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 Figure 14. Finally, the multiplication here uses the floating-point multiplier based on QFT (*Q-Multiplier* module, given by Figure 13). For convenience, we omit the process of the seeking the maximum value in the *Multiplier* module and the *Converter* module before the *Q-Multiplier* module, refer to Figures 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 Figure 12) and $h_1 + w_1 + l_1$ *Divided by 2* modules (given by Figure 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 Figures 17 and 18.

For convenience, we show the schematic representation of the scaling-up circuit as Figure 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, $r_y = 4$, $r_x = 2$, $r_z = 2$.

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

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

4 Conclusions

Quantum computation has become a novel and important tool in the field of image/data processing. In this paper, 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 application in many diverse fields such as artificial intelligence, aeronautics, architecture, biological science, medicine, etc. Floating-point numbers offer great savings in 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 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 the 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 the future research.

Acknowledgments

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).

Figure 17. *Scaling-up circuit(1)*

Figure 18. *Scaling-up circuit(2)*

Figure 19. *Scaling-down circuit(1)*

References

- [1] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, **21**(6-7), 467–488(1982).
- [2] P. W. Shor. Algorithms for quantum computation : Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134(1994).
- [3] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 212–219(1996).
- [4] S. E. Venegas-Andraca and S. Bose. Storing, processing and retrieving an image using quantum mechanics. In *Proceeding of SPIE Conference of Quantum Information and Computation 5105*, pp. 137–147(2003).
- [5] J. I. Latorre. Image compression and entanglement. *Computer Science*, pp. 1–4(2005).
- [6] S. E. Venegas-Andraca and J. L. Ball. Processing images in entangled quantum systems. *Quantum Information Processing*, **9**(1), 1–11(2010).
- [7] P. Q. Le, F. Dong, and K. Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, **10**(1), 63–84(2011).
- [8] Y. Zhang, K. Lu, Y. Gao, and M. Wang. NEQR : a novel enhanced quantum representation of digital images. *Quantum Information Processing*, **12**(8), 2833–2860(2013).
- [9] N. Jiang and L. Wang. Quantum image scaling using nearest neighbor interpolation. *Quantum Information Processing*, **14**(5), 1559–1571(2015).
- [10] N. Jiang, J. Wang, and Y. Mu. Quantum image scaling up based on nearest-neighbor interpolation with integer scaling ratio. *Quantum Information Processing*, **14**(11), 4001–4026(2015).
- [11] H. Li, P. Fan, H. Xia, H. Peng, and S. Song. Quantum implementation circuits of quantum signal representation and type conversion. *IEEE Transactions on Circuits and Systems I : Regular Papers*, **66**(1), 341–354(2019).
- [12] R. Zhang, D. Lu, and H. Yin. A generalized floating-point representation and manipulation of quantum signals based on IEEE-754. *International Journal of Theoretical Physics*, **59**, 936–952(2020).
- [13] R. Zhang, M. Xu, and D. Lu. A generalized floating-point quantum representation of 2-D data and their applications. *Quantum Information Processing*, **19**(11), 390(2020).
- [14] A. Fijany and C. P. Williams. Quantum wavelet transforms : fast algorithms and complete circuits. In *NASA international conference on quantum computing and quantum communications*, pp. 10–33(1998).
- [15] S. Caraiman and V. Manta. Quantum image filtering in the frequency domain. *Advances in Electrical and Computer Engineering*, **13**(3), 77–84(2013).
- [16] L. Ruiz-Perez and J. C. Garcia-Escartin. Quantum arithmetic with the quantum Fourier transform. *Quantum Information Processing*, **16**(6), 152(2017).
- [17] P. Li and H. Sun. Quantum color image filtering in the frequency domain. *Journal of Electronics and Information Technology*, **40**(3), 594–601(2018).

- [18] R. Asaka, K. Sakai, and R. Yahagi. Quantum circuit for the fast Fourier transform. *Quantum Information Processing*, **19**(8), 277(2020).
- [19] S. Chakraborty, S. H. Shaikh, A. Chakrabarti, and R. Ghosh. An image denoising technique using quantum wavelet transform. *International Journal of Theoretical Physics*, **59**(11), 3348–3371(2020).
- [20] W. L. Chang and A. V. Vasilakos. *Fundamentals of Quantum Programming in IBM's Quantum Computers*. Springer, 2021.
- [21] A. M. Grigoryan and S. S. Agaian. New look on quantum representation of images : Fourier transform representation. *Quantum Information Processing*, **19**(5), 148(2020).
- [22] J. Sang, S. Wang, and X. Niu. Quantum realization of the nearest-neighbor interpolation method for FRQI and NEQR. *Quantum Information Processing*, **15**(1), 37–64(2016).
- [23] R. Zhou, W. Hu, P. Fan, and H. Ian. Quantum realization of the bilinear interpolation method for NEQR. *Scientific Reports*, **7**(1), 2511(2017).
- [24] P. Li and X. Liu. Bilinear interpolation method for quantum images based on quantum Fourier transform. *International Journal of Quantum Information*, **16**(4), 1850031(2018).
- [25] R. Zhou, Y. Cheng, and D. Liu. Quantum image scaling based on bilinear interpolation with arbitrary scaling ratio. *Quantum Information Processing*, **18**, 267(2019).
- [26] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, New York, 2000.
- [27] Y. Zhang, K. Lu, K. Xu, Y. Gao, and R. Wilson. Local feature point extraction for quantum images. *Quantum Information Processing*, **14**(5), 1573–1588(2015).
- [28] T. Haener, M. Soeken, M. Roetteler, and K. M. Svore. Quantum circuits for floating-point arithmetic. *Lecture Notes in Computer Science*, pp. 162–174(2018).