Quantum Fast Corner Detection Algorithm

Suzhen Yuan, Wenping Lin, Bo Hang, Hongying Meng

Abstract

With the increase in image data, the task of image recognition is also increasing. Detection of local features is the basis of image recognition, and corners are important local features. However, the size and number of images seriously affect the speed of corner detection. Therefore, this paper designs a quantum fast corner detection algorithm, which takes full advantage of quantum parallelism. The algorithm is implemented in three steps: the first step selects the neighborhood of the nucleus, the second step selects two thresholds k_1 and k_2, and the third step judges the corners. The first and second steps are the same with the selection of the range of parameters in the classical corner detection algorithm. The third step is divided into two stages. The first stage calculates the differences between the grayscale values of the nucleus and pixels in the neighborhood, followed by the comparison of those differences with threshold k_1, then performs quantum measurement on the comparison results, and finally organizes the measured result into an array M. The second stage uses the array M for counting, followed by the comparison of those counted results with threshold k_2, and then judges the corners. It is worth noting that through the quantum-classical-quantum mode, quantum resources can be saved greatly. The analysis of the proposed quantum fast corner detection algorithm shows that the time complexity and quantum delay of the algorithm do not increase with the increase in the size and number of images, and its time complexity is exponentially lower than that of the classical fast corner detection algorithm.

Keywords: Quantum fast corner detection, Quantum full subtractor, Quantum parallelism, Time complexity

1. Introduction

Quantum computers can store and process information in parallel [1]. Larger quantum algorithms are believed to be capable of solving certain computational problems, e.g., the Shor's prime factorization algorithm [2] and Grover's quantum search algorithm [3] demonstrated the superiority and capability of the quantum algorithms. In recent years, with the rapid development of the computer hardware, the quantum technology is gradually moving from theoretical research to practical applications.

Image recognition plays an important role in various fields. Its basis is the detection of local features. One of the most intuitive types of feature points is the corner. Corners are image points that show a great change in intensity as there are two dominant and different edge directions in the local neighbourhood of a corner. Corner algorithms are wildly used for matching graphics [4], detecting moving objects [5], [6] and tracking objects [7]. In the last decade, different corner algorithms were developed [8, 9, 10, 11]. Among those corner detectors, the fast corner detector is more faster [12, 13]. However, no quantum fast corner detection algorithm has yet been developed.

In this paper, a quantum fast corner detection algorithm is proposed. The main contributions made in this paper are as follows. Firstly, a quantum full subtractor with fewer quantum gates is designed for reducing the time complexity and quantum delay of the whole algorithm. Secondly, a quantum fast corner detection algorithm with high parallelism is proposed. Its time complexity is exponentially lower than that of its corresponding classical algorithm. Moreover, it is not affected by the size and number of images to be processed. Thirdly, we adopt a new quantum-classical-quantum mode for processing quantum images, which can help in saving quantum resources.

The remainder of this paper is organised as follows. Section 2 describes the quantum image storage models and various operations on quantum images. Section 3 elaborates the proposed quantum fast corner detection algorithm. Section 4 calculates the cost of quantum resources. In Section 5, the simulation of the proposed quantum fast corner detection algorithm is implemented and corner maps are obtained. Finally, Section 6 concludes this work.

2. Preparatory work

In the quantum image processing, images need to be stored first in the quantum state. For this, three models are widely used, namely the Qubit Lattice representation model [14], FRQI (Flexible Representation of Quantum Images) model [15], and NEQR (Novel Enhanced Quantum Representation of Digital Images) model [16]. The grayscale values are stored in the probability amplitude of the quantum state in the Qubit Lattice and FRQI models, and in the quantum ground state in the NEQR model. Therefore, the NEQR model can operate with the grayscale values more accurately. On the other hand, the NEQR model can accurately restore the quantum image information to the classical image information through finite quantum measurements. Therefore, in this paper, the NEQR model

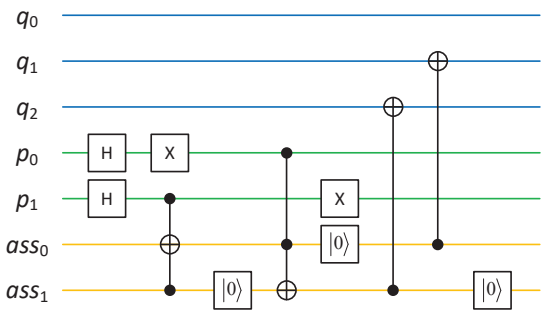


Figure 1: Quantum circuit for preparing pixel with position of 01 and grayscale value of 011

is chosen for storing quantum images. We introduce the NEQR model and some quantum operating modules below which will be used in the present work.

2.1. NEQR model for image representation

We optimized quantum circuits of the NEQR model in our previous work [17], where the number of the auxiliary qubits became 2 (constant) due to the use of the reset operation. It solved the problem that auxiliary qubits increase with the increase in the image size. In this paper, we adopt the method proposed in [17] for preparing quantum images. Next, we briefly introduce the concept of the image preparation stage proposed in [17].

For a quantum image of size of $2^n \times 2^n$ and scope of grayscale value of [0, 2^q-1], the NEQR model can be written in the form of a quantum superposition state as expressed by Eq. (1)[16].

$$|I\rangle = \frac{1}{2^n} \sum_{Y=0}^{2^n-1} \sum_{X=0}^{2^n-1} \bigotimes_{i=0}^{q-1} \left|C_{YX}^i\right\rangle |YX\rangle \tag{1}$$

where X and Y represent the position information along x and y axes, and $\bigotimes_{i=0}^{q-1} \left|C_{YX}^i\right\rangle$ represents the grayscale information.

Taking a 2×2 image with the scope of the grayscale values of [0, 7] as an example, the grayscale values can be expressed by Eq.(2).

$$f_{2\times 2} = \begin{bmatrix} 7 & 3 \\ 5 & 6 \end{bmatrix} \tag{2}$$

Here, 2 qubits are needed to store the position information, and 3 qubits are needed to store the grayscale information. Figure 1 shows a quantum circuit for preparing a pixel with position of '01' and grayscale value of '011'. The reset operation is represented by $|0\rangle$. The position information is encoded in qubits p_0 and p_1, and the grayscale information is encoded in qubits q_0, q_1 and q_2. Besides, other 2 auxiliary qubits ass_0 and ass_1 are used. The advantage of this quantum circuit is that the number of auxiliary qubits does not increase with the size of the image.

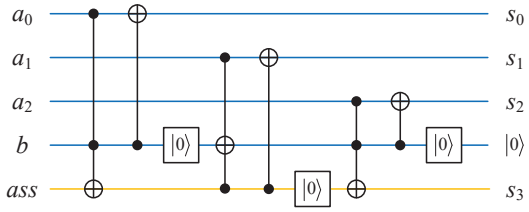


Figure 2: Quantum circuit of quantum adder

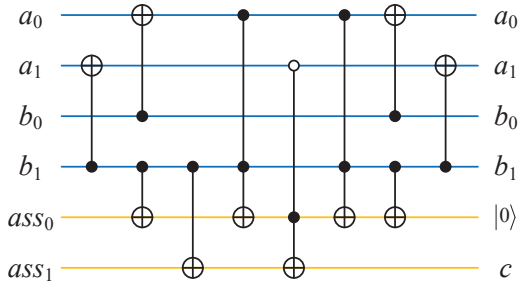


Figure 3: Quantum circuit of two-bit quantum comparator

2.2. Quantum adder

Figure 2 shows the specific circuit of 3-bit and 1-bit quantum adders [18]. Among them, $|a_2a_1a_0\rangle$ is set as a 3-bit addend, and $|b_0\rangle$ is set as a 1-bit addend, and $|ass\rangle$ is set as an auxiliary qubit. The final result after the addition is stored in $|s_3s_2s_1s_0\rangle$ as the output.

2.3. Quantum comparator

In order to save qubits, the quantum comparator designed by Xia and Li et al [19] is adopted in this paper. Taking the comparison of two 2-digit quantum comparators as an example, the quantum circuit diagram is shown in Fig. 3, where $|a\rangle = |a_1a_0\rangle$ and $|b\rangle = |b_1b_0\rangle$ are two numbers to be compared, and $|ass_0\rangle$ and $|ass_1\rangle$ are 2 auxiliary qubits. The comparative results are stored in the output qubit $|c\rangle$, which can be described as follows: $c = 0$ when $a \geq b$, and $c = 1$ when $a < b$.

2.4. Quantum full subtractor

In order to reduce the consumption of quantum resources, we design an efficient quantum full subtractor. Figure 4 shows the proposed quantum circuit of a 1-bit quantum full subtractor, where $|a\rangle$ is the minuend, $|b\rangle$ is the subtrahend, and $|c\rangle$ is the borrow information from the previous step. The last qubit $|ass\rangle$ is an auxiliary qubit which is initialized to $|0\rangle$. The subtracted result is stored in $|s\rangle$, and the borrow from the high-order bit is stored in $|c\rangle$.

Figure 5 shows the quantum circuit of a 3-bit quantum full subtractor, where $|a\rangle = |a_2a_1a_0\rangle$ is the minuend and $|b\rangle = |b_2b_1b_0\rangle$ is the subtrahend. Further, $|ass_0\rangle$ and $|ass_1\rangle$ are two auxiliary qubits, which are used to store the borrow information and can be reused by a reset operation. The subtracted result is stored in

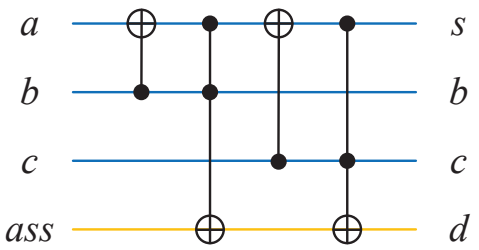


Figure 4: 1-bit quantum full subtractor

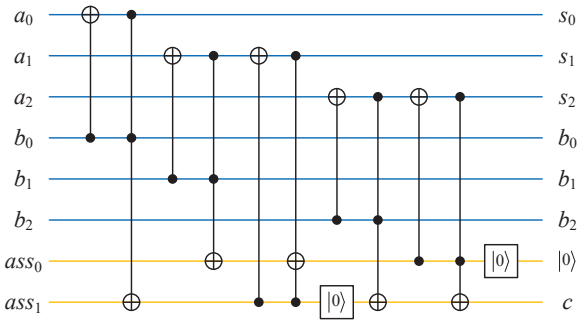


Figure 5: 3-bit quantum full subtractor

$|s\rangle = |s_2 s_1 s_0\rangle$, and the borrow from the high-order bit is stored in $|c\rangle$.

Generalized to n bits, an n-bit quantum full subtractor is implemented in the same way, where the number of auxiliary qubits do not increase with the minuend and subtrahend. For the n-bit quantum full subtractor, U_{SUB} can be expressed as:

$$U_{SUB}\,|a\rangle\,|b\rangle\,|ass_0\rangle\,|ass_1\rangle = |s\rangle\,|b\rangle\,|0\rangle\,|c\rangle \tag{3}$$

where $|a\rangle$, $|b\rangle$ and $|s\rangle$ are, respectively, the n-bit minuend, subtrahend and result; $|ass_0\rangle$ and $|ass_1\rangle$ are two auxiliary qubits; and $|c\rangle$ is the borrow from the high-order bit.

3. Quantum fast corner detection algorithm

In this section, the proposed quantum fast corner detection algorithm is presented. Firstly, its steps are given, and then those are introduced in detail and the corresponding quantum circuits are designed.

3.1. Steps of quantum fast corner detection algorithm

Firstly, the concepts related to the corner detection are introduced. The center pixel of a circle is called the 'nucleus' [20]. A compact region in the circle is similar to the nucleus in the grayscale value, which is referred to as USAN (Univalue Segment Assimilating Nucleus) [20]. Figure 6 shows three representative shapes of USAN, where the nucleus of Fig. 6(a) is inside USAN, the nucleus of Fig. 6(b) is an edge point, and that of Fig. 6(c) is a corner. The task of the fast corner detection algorithms is to detect corners through image processing methods.

(a) Nucleus is inside the USAN (b) Nucleus is an edge point (c) Nucleus is a corner

Figure 6: Three representative shapes of USAN

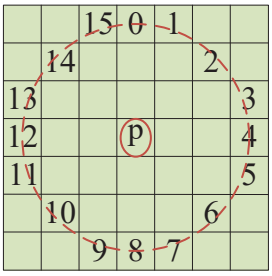


Figure 7: Neighborhood for corner detection

Next, an efficient method for judging corners in the classical fast corner detection algorithm is briefly introduced [21]. Its first step selects the neighborhood of the nucleus, whose pixels participate in the subsequent operations. The second step selects various thresholds. The judgment for a corner needs to select two thresholds k_1 and k_2. The first threshold k_1 is determined by the contrast of an image, which decreases as the contrast of the image decreases. The second threshold k_2 is determined by the accuracy of the corners, which decreases as the accuracy of the corners decreases. The third step is to determine the corners. Among all the neighborhood pixels, if the difference between the grayscale values of the pixels of consecutive N points and the grayscale value of the nucleus p is greater than or equal to the threshold k_1, and N is greater than or equal to k_2, then the nucleus p is a corner. The first two steps of the quantum fast corner detection algorithm designed in this paper are the same with those of the classical fast corner detection algorithm. The third step is implemented in two stages.

The implementation steps of the proposed quantum fast corner detection algorithm are introduced below.

Step 1: The neighborhood of the nucleus is selected, which is shown in Fig. 7. If point p is selected as the nucleus, the selected neighborhood will be the pixels on a circle with p as the center and 3 pixels as the radius [21]. The neighborhood of the nucleus is marked as $0 \rightarrow 15$ by the red dashed line.

Step 2: Two thresholds are selected. Following the recommended range of $[5, 30]$ [20], k_1 in this paper is set to be 20 according to the contrast of the image. The recommended range of k_2 is $[9, 12]$ [8], and accordingly we choose its value to be 11.

Step 3: In the first stage, the difference between each nucleus and its neighboring pixels is calculated and compared with threshold k_1, and the compared all such results are measured and sorted into an array M. In the second stage, array M is used for counting, then the counted results are compared with

threshold k_2, and finally the corner is determined. The following content of this paper mainly focuses on the two stages of this third step. The first stage is described in Section 3.2, and the second stage is discussed in Section 3.3.

3.2. Stage 1: Calculation and comparison of differences

3.2.1. Extraction of nucleus image and neighborhood information

In order to facilitate the introduction, in this paper an image of size of 14×14 and grayscale range of [0, 255] is selected. The image and its grayscale values are shown in Figs. 8(a) and 8(b), respectively. As shown in the red box in Fig. 8(a), the image formed by its internal pixels is called the nucleus image. Each nucleus pixel has 16 neighborhoods. In order to increase the parallelism of the overall algorithm, we use 16 auxiliary images each of size of 8×8 to represent all the neighborhood information. The first auxiliary image is shown in the green dashed box in Fig. 8(a), which stores the neighborhood information of all the pixels in the nucleus image as marked as '0'. By analogy, the second to sixteenth auxiliary images store the neighborhood information of all the pixels as marked as $1 \rightarrow 15$. Taking the third auxiliary image as shown in the yellow box as an example, the corresponding relationship between the auxiliary image and neighborhood information can be established as follows: if the position of the pixel of the auxiliary image is the same as that of the nucleus image, then the grayscale value of the auxiliary image is the neighborhood marked '2' of the nucleus image. For example, when both of the nucleus image and the third auxiliary image are at position 'a' as shown in Fig. 8(a), the grayscale value of 240 in the auxiliary image is the neighborhood marked '2' of the grayscale value 70 in the nucleus image. When both of them are at position 'b', the grayscale value 250 in the auxiliary image is the neighborhood marked '2' of the grayscale value 255 in the nuclear image, and so on.

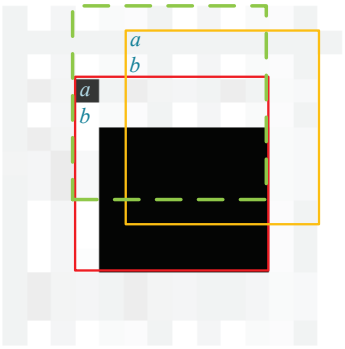
A total of 16 auxiliary images and 1 nucleus image are extracted, which lay the foundation for parallel processing of the subsequent steps.

3.2.2. Preparation of nucleus and auxiliary images based on NEQR model

Both nucleus and auxiliary images are prepared into a quantum superposition state. The grayscale information about the nucleus image is encoded into 8 qubits $|b_7b_6b_5b_4b_3b_2b_1b_0\rangle$. The grayscale information about the 16 auxiliary images shares 8 qubits $|a_7a_6a_5a_4a_3a_2a_1a_0\rangle$. Four qubits $|c_3c_2c_1c_0\rangle$ are used to distinguish these 16 auxiliary images, which are called as the marker qubits. These nucleus and auxiliary images share the position information, which is encoded into 6 qubits $|d_5d_4d_3d_2d_1d_0\rangle$. In addition, two auxiliary qubits $|ass_0\rangle$ and $|ass_1\rangle$ are required.

Figure 9 shows the preparation of a quantum circuit for a nucleus pixel and one of its neighborhood pixel. The position of this nucleus pixel is '000000' and its grayscale value is '01000110'. The marker qubit of the neighborbood pixel is '1111' and its grayscale value is '11110100'.

The advantage of the method for preparation of the nucleus and auxiliary images is that we use 4-bit marker qubits to encode the grayscale values of 16 auxiliary images into 8 qubits.



(a) Image to be detected (the first auxiliary image is in the green dashed box, the nucleus image is in the red solid box, and the third auxiliary image is in the red solid box)

240	255	244	255	247	250	240	255	244	255	247	250	240	255
240	240	240	240	240	240	240	240	240	240	240	240	240	240
240	255	244	255	247	250	240	255	244	255	247	250	240	255
240	236	241	70	240	236	241	244	240	236	240	247	246	255
240	255	244	255	247	250	240	255	244	255	247	250	240	255
240	236	241	254	4	4	5	5	5	6	7	247	246	255
241	244	234	246	3	6	6	5	7	7	7	247	246	255
240	244	234	246	5	5	2	15	4	4	4	247	246	255
240	255	244	255	3	3	4	5	6	7	8	250	240	255
240	255	244	255	2	3	4	5	6	7	8	250	240	255
240	255	244	251	3	4	5	6	7	7	8	250	240	255
240	236	241	244	240	236	241	244	240	240	244	247	246	255
240	236	241	244	240	236	241	244	240	240	244	247	246	255
240	255	244	255	247	250	240	255	244	255	247	250	240	255

(b) Grayscale value of image to be detected, nucleus image and third auxiliary image

Figure 8: Grayscale values of the image to be detected

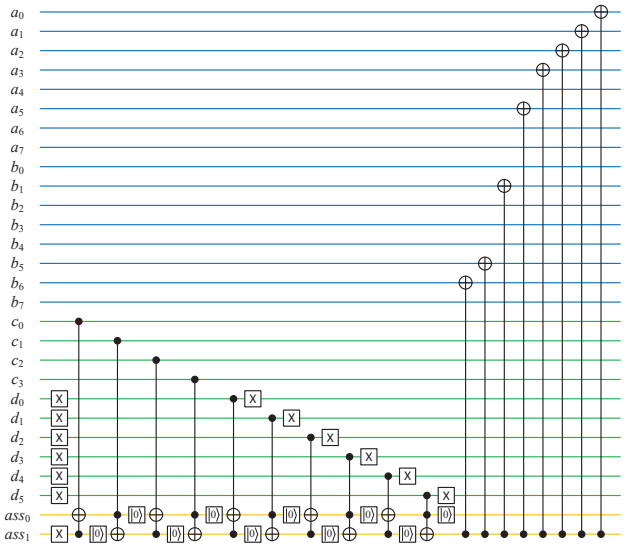


Figure 9: Quantum circuits of a nucleus and one of its neighborhood pixel

Compared with the way that each auxiliary image uses 8 qubits to store its grayscale values, this method greatly saves the consumption of qubits and computational resources.

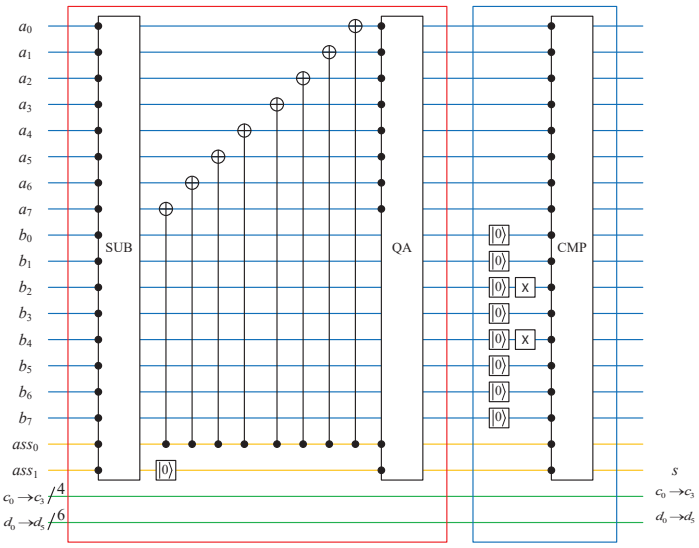


Figure 10: Quantum circuits of the first stage of quantum fast corner detection

3.2.3. *Calculation of difference D and its comparison with threshold k_1*

The red block in Fig. 10 shows the quantum circuit that calculates the differences between the grayscale values of the nucleus and neighbohood auxiliary images, where $|a_7a_6a_5a_4a_3a_2a_1a_0\rangle$ is the minuend, $|b_7b_6b_5b_4b_3b_2b_1b_0\rangle$ is the subtracted, SUB represents the quantum full subtractor, and QA is the quantum adder. We use an 8-bit quantum full subtractor for calculating the difference between the grayscale values of the auxiliary images and nucleus. It is worth noting that the subtracted result is displayed in a 2's complement form. $|a_7a_6a_5a_4a_3a_2a_1a_0\rangle$ is the value of the 2's complement form, and the last borrow bit $|ass_0\rangle$ is its sign bit. However, we should convert the 2's complement form into the true form as the value of the true form needs to be used in the subsequent comparison. Finally, the absolute values of the differences between the grayscale values of the nucleus and auxiliary images are obtained and stored in the quantum register $|a_7a_6a_5a_4a_3a_2a_1a_0\rangle$. In the following part, we use D to represent the absolute value of the difference.

The blue block in Fig. 10 shows the comparison of $|a\rangle$ with the threshold k_1, where CMP is the quantum comparator. Firstly, we use the reset operation to clean the previously generated garbage output bits $|b_7b_6b_5b_4b_3b_2b_1b_0\rangle$ and $|ass_1ass_0\rangle$. Then, the threshold value k_1 (the value of k_1 is '00010100') is stored in $|b_7b_6b_5b_4b_3b_2b_1b_0\rangle$. D is compared with the threshold k_1 using a quantum comparator, and the comparative result is stored in s. The obtained results are as follows: s is 0 when D is less than k_1, and s is 1 when D is greater than or equal to k_1.

It is worth noting that because of the quantum parallelism, when the calculation and comparison of a nucleus and its neighbors are complete, the same operations for all the pixels are also completed at the same time.

3.2.4. *Quantum measurement and data restructuring*

Suppose that Fig. 8 shows an image to be detected. After the above calculation, a probability histogram can be obtained, which would an array P with a total of 1024 elements. Each element is a binary number containing 1-bit comparative result, 6-bit positional information and 4-bit marker information. For the convenience of displaying, we extract one out of every 16 elements to form the data as shown in Fig. 11. The abscissa of the histogram represents 64 elements obtained by quantum measurement, and its ordinate represents the probability. Take the first element "000000010000" for example. The first bit '0' from left to right is the compared result which indicates that $|a\rangle$ is less than k_1, the second to seventh bits '000001' are the positional information of the second pixel in the nucleus image, and the eighth to eleventh bits '0000' are the marker qubits which point to the first auxiliary image.

After the 1024 elements of array P are sorted in ascending order using the positional information, the array is divided into a total of 64 groups by taking every 16 adjacent elements as a group, where the $2 \to 7$ bits in each group are identical. The first bit of every element in each group and one element from the $2 \to 7$ bits are combined to form a new binary number. That is, the 1024 elements of P will be combined into 64 elements, which is denoted by M. Take the ith element "00000001100000000000001" in array M as an example: "0000000110000000" from left to right is the comparative results, which can be denoted as $M_0^i - M_{15}^i$. Here, "000001" represents the position of the nucleus pixel, which can be denoted as $M_{16}^i - M_{21}^i$. Figure 12 shows the positional relationship between the individual qubit of each element of M and the image pixel, where the '0' marked in red is the starting position of the neighborhood pixel, and the '0' highlighted in yellow is its ending positing.

3.3. *Corner detection*

Based on the comparative result M, we use the count module to obtain the result. If the result is greater than or equal to the threshold k_2, the nucleus pixel point becomes a corner. Next, we introduce the counting module.

3.3.1. *Counting module*

Here, T is used to represent the result and it is initialized to 0. One number participates in the counting each time, which is either "0" or "1". T is incremented by 1 when the number participating in the counting is "1", and all the previous numbers participating in the counting are made "1". When the number participating in the counting is "0", the result T remains unchanged regardless of whether the subsequent number involved in the counting is "0" or "1". Next, take '1111100011' as an example to show the counting method. T is initialized to 0, the first five numbers participating in the count are all "1", and the sixth number is "0". Therefore, T remains 5, no matter what values appear after the sixth number.

3.3.2. *Preparation of array M*

In order to accomplish the task of counting, the array M should be encoded into a quantum superposition state. Firstly, the binary information about the last 6 bits $M_{16} - M_{21}$ of the array M is stored in the 6 qubits $|d_5d_4d_3d_2d_1d_0\rangle$. Then, the neighborhood comparison results $M_0 - M_{15}$ for each element should

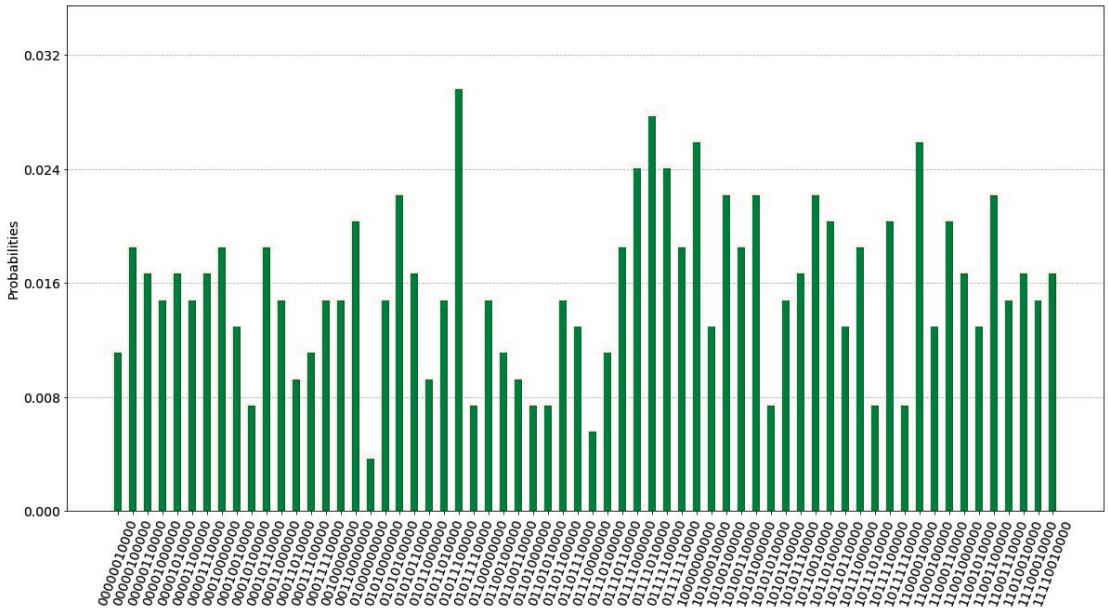


Figure 11: Probability histogram of 64 data in array P

		0	0	0		
	0				0	
0						0
0			000 001			0
0						0
	0				0	
		0	1	1		

Figure 12: Comparative results of the neighborhood at position 000001 in the nucleus image

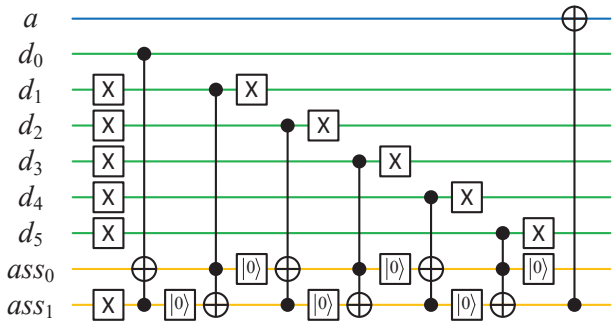


Figure 13: Preparation quantum circuit for an element of array M

be encoded. Although there are a total of 16 neighborhood comparison results in each element, in order to save qubits, we select only one comparison result at a time for preparation. After completing the counting of M_0, prepare M_1 for counting. Thus, $M_{16} - M_{21}$ are prepared in the qubit $|a\rangle$. Further, two additional auxiliary qubits $|ass_1 ass_0\rangle$ are required. Figure 13 shows the preparation the quantum circuits of M_7 and $M_{16} - M_{21}$ for the second element "00000001100000000000001" of the array M.

3.3.3. Counting and judgement of corners

In this paper, the threshold k_2 is taken to be 11. Figure 14 shows an example of counting, where p is the nucleus, 16 surrounding neighborhood pixels store the comparative results $M_0 - M_{15}$ in the array M, and 0 to 15 are their indices.

The whole process of counting and judging is divided into 3 parts: first round of counting, second round of counting, and the third part is for the corner judgment. Taking Fig. 14 as an example, these three parts are elaborated below.

Firstly, the first round of counting:

Step 1: From M_0 to M_{11}, K_2 numbers are selected in the clockwise direction, for which the specific counting sequence

		1_{15}	1_{0}	1_{1}		
	1_{14}				1_{2}	
1_{13}						1_{3}
1_{12}			p			1_{4}
1_{11}						1_{5}
	0_{10}				1_{6}	
		0_{9}	1_{8}	0_{7}		

Figure 14: An example for counting

becomes "11111110100". Using the counting module to count,

the number of consecutive '1's from the beginning of the sequence is found to be 7. So, the result of T_1 is 7. Step 2: From M_{15} to M_5, K_2 numbers are selected in the counterclockwise direction, for which the specific counting sequence becomes '11111001011'. The number of consecutive '1's from the beginning of the sequence is found to be 5. So, the result of the first round is $T_1 = 12$ (7+5=12).

Secondly, the second round of counting:

Step 1: From M_7 to M_{13}, K_2 numbers are selected in the counterclockwise direction, for which the specific counting sequence becomes '01111111111'. The number of consecutive '1's from the beginning of the sequence is found to be 0. So, the result of T_2 is 0. Step 2: From M_8 to M_2, K_2 numbers are selected in the clockwise direction, for which the specific counting sequence becomes '10011111111'. The number of consecutive '1's from the beginning of the sequence is found to be 1. So, the result of T_2 is 1 (0+1).

Thirdly, judging corners:

The counting results T_1 and T_2 are compared with k_2. If both T_1 and T_2 are less than the threshold 11, p will not be a corner. Otherwise, p is a corner. In this example, T_1 is greater than the threshold 11. So, p is the corner.

Note that independent pixels need to be eliminated in the process of judging corners. If all the 16 comparative results $M_0 - M_{15}$ are 1 only, then the nucleus is an independent pixel point and it does not belong to any corner. So, it needs to be screened out. In this algorithm, 4 qubits are chosen to store the final result. Its maximum value is 1111, which is 15 when converted in to decimal value. If all the 16 comparative results are 1 only, then the adder will overflow and the result will be 0. Therefore, no special operation is required to exclude independent pixels.

Algorithm 1 shows the process of the first round of counting. The second round of counting is similar to the first round of counting.

As shown in Fig. 15, the quantum circuit for corner detection is designed, where registers $|b_3b_2b_1b_0\rangle$ are used to store the counting results T_1 and T_2, which are initialized to $|0000\rangle$. It is necessary to judge the status of both current and previous bits during the counting. So, add a register $|c\rangle$ to save the status of the previous bit. *QA* represents the quantum adder. The quantum circuit in the red rectangle shows the first count of the first round. The quantum circuit in the first purple rectangle is the comparison operation, which compares T_1 with k_2, and stores the result in register $|e\rangle$. The quantum circuit in the green rectangle shows the last count of the second round. The quantum circuit in the second purple rectangle is the comparison operation, which compares T_2 with k_2, and stores the result in register $|ass_1\rangle$. Finally, the quantum circuit in the yellow rectangle makes the final judgement, and stores the result in register $|a\rangle$. $a = 0$ means that the corresponding pixel is a corner, and $a = 1$ means that the corresponding pixel is not a corner.

algorithm 1 First round of counting

Input Threshold value $k_2 \in (9, 10, 11, 12)$
Output The comparative result of the first round T_1

Prepare a qubit to store the comparative result a of array M
Prepare 4 qubits to store the result of the addition $b_3b_2b_1b_0$
$b_3b_2b_1b_0 \leftarrow 0000$
$c \leftarrow 1$
for $number \leftarrow 0$ to k_2 **do**
    $a \leftarrow p_{number+1}$
    if $number \geq 7$ **then**
        if $c == 1$ and $a == 1$ **then**
            $c \leftarrow a\&c$
            $b_3b_2b_1b_0 \leftarrow b_3b_2b_1b_0 + c$
        end if
    else if $number \geq 3$ and $number < 7$ **then**
        if $c == 1$ and $a == 1$ **then**
            $c \leftarrow a\&c$
            $b_2b_1b_0 \leftarrow b_2b_1b_0 + c$
        end if
    else if $number \geq 1$ and $number < 3$ **then**
        if $c == 1$ and $a == 1$ **then**
            $c \leftarrow a\&c$
            $b_1b_0 \leftarrow b_1b_0 + c$
        end if
    else if $number == 0$ **then**
        if $c == 1$ and $a == 1$ **then**
            $c \leftarrow a\&c$
            $b_0 \leftarrow b_0 + c$
        end if
    end if
end for
$c \leftarrow 1$
for $number \leftarrow 0$ to k_2 **do**
    $number_2 \leftarrow 16 - number$
    $a \leftarrow p_{number_2}$
    if $c == 1$ and $a == 1$ **then**
        $c \leftarrow a\&c$
        $b_3b_2b_1b_0 \leftarrow b_3b_2b_1b_0 + c$
    end if
end for
if $b_3b_2b_1b_0 \geq k_2$ **then**
    $T_1 \leftarrow 0$
else
    $T_1 \leftarrow 1$
end if
return T_1

4. Calculation of cost of quantum resources

The cost in this work is calculated from the aspects of qubit cost, time complexity and quantum delay. The qubit cost is the total number of qubits required to design a quantum circuit. The time complexity is calculated as follows: the time complexity of each of CNOT and NOT gates is 1, and that of Toffoli gate is 5 [22]. The quantum delay is calculated by combining the time complexity and "depth" as follows: the quantum delay of each of CNOT and NOT gates is 1, and that of Toffoli gate is 5. The quantum delay at the same depth takes the maximum time complexity of the depth [23].

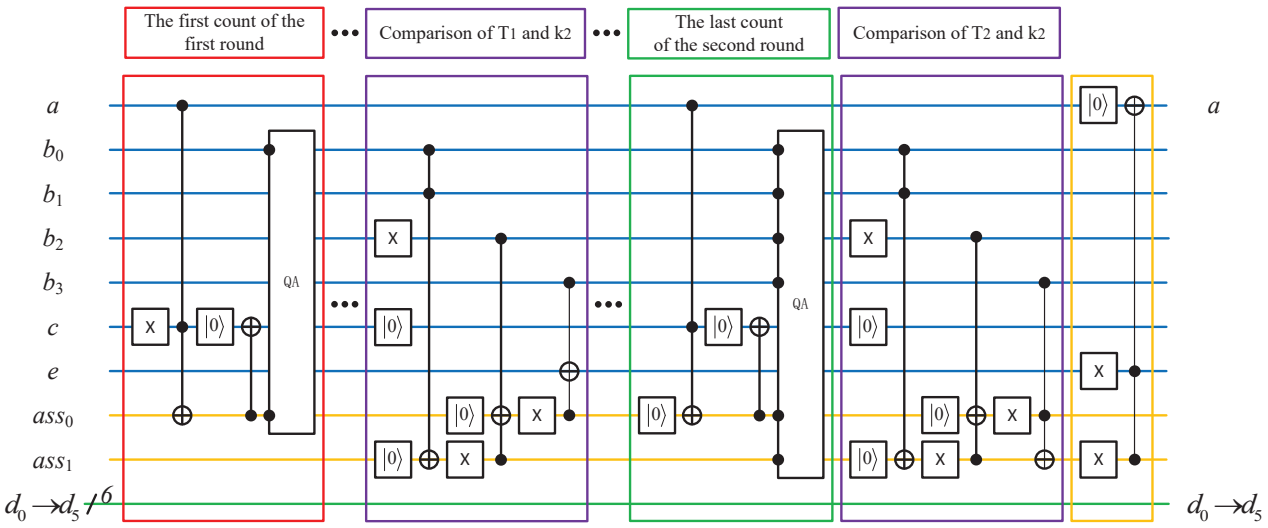


Figure 15: Quantum circuits for the second stage of quantum fast corner detection algorithm

Table 1: Time complexity and quantum delay of quantum full subtractors

	Time complexity	Quantum delay
Cheng [24]	$16n$	$16n$
Yuan [25]	$16n$	$14n$
Proposed	$12n$	$12n$

4.1. Cost of quantum full subtractor

We presented the design of an efficient quantum full subtractor in subsection 2.4, the cost of which is calculated as follows:

For each additional bit, the quantum full subtractor adds 2 CNOT gates and 2 Toffoli gates, which increases the time complexity leading to the quantum delay of 12.

It can be seen in Table 1 that the quantum full subtractor proposed in this paper outperforms the those designed in [24] and [25] in terms of both time complexity and quantum delay.

4.2. Cost in this work

Assume that the size of the image to be detected is $2^n \times 2^n$, where the range of the grayscale values is $[0, 2^m - 1]$, and the second threshold is k_2.

Qubit cost: The whole algorithm is divided into two stages. In the first stage, the image preparation consumes $2m + 2n + 6$ qubits. Due to the use of the quantum reset operation, the consumption of qubits does not increase in the following operations. Hence, the qubit cost in the first stage becomes $2m+2n+6$. After the quantum measurement, all the qubits are released. In the second stage, the preparation step consumes $2n+3$ qubits, where 4 qubits are used to store the counting result. At the same time, 2 auxiliary qubits are needed. The second stage eventually consumes $2n + 9$ qubits. Therefore, the largest number of qubits consumed in the two stages is the number of the qubits consumed by the entire quantum corner detection algorithm, which is $2m + 2n + 6$.

Time complexity: The first stage uses 1 quantum full subtractor, 1 quantum adder, 1 quantum comparator, 2 NOT gates and m CNOT gates. The time complexities of individual subtratcor, adder and comparator are $12m$, $6m$ and $14m-6$, respectively. The total time complexity of the first stage is $12m + m + 6m + 2 + 14m - 6 = 33m - 4$. The second stage consists of $4k_2$ counts, 2 comparisons and 1 OR operation. The $4k_2$ counts include k_2 Toffoli gates, k_2 cnot gates, and $4k_2$ adders. Among them, the $4k_2$ adders include 2 1-bit adders, 4 2-bit adders, 8 3-bit adders, and $4k_2 - 14$ 4-bit adders. The time complexity of $4k_2$ counts is $102k_2 - 132$. Two comparisons involve 6 NOT gates and 6 Toffoli gates. So, its time complexity is 36. The OR operation includes 2 NOT gates and 1 Toffoli gate. So, its time complexity is 7. The time complexity of the second stage is $102k_2 - 132 + 36 + 7 = 102k_2 - 89$. The total time complexity is $33m + 102k_2 - 93$, obtaining by summing up the individual time complexities of the two stages.

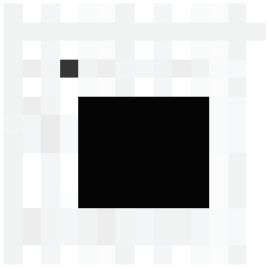
Quantum delay: In the first stage, the quantum delay of 1 m-bit comparator is $4m - 5$ less than the time complexity. The first stage also has 2 NOT gates at the same depth. Hence, the quantum delay of the first stage is $29m$. Similarly, the quantum delay of the second stage is $102k_2 - 88$. The consumed total quantum delay is $29m + 102k_2 - 88$.

Since any existing quantum image corner detection algorithm could not be found, we compare the time complexity of the proposed algorithm with that of the classical image corner detection algorithm.

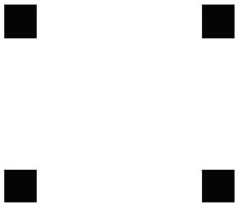
Table 2 shows the comparison between the time complexities of the quantum fast corner detection and classical fast corner detection algorithms. As can be seen in table 2, the proposed quantum fast corner detection algorithm could reduce the time complexity at an exponential rate compared with the corresponding classical algorithm.

Table 2: Comparison of time complexity

Fast corner detection algorithm	Time complexity
Trajković [20]	$O(2^{2n})$
Rosten [21]	$O(2^{2n})$
Present proposal	$O(m + k_2)$



(a) Schematic of the image to be detected (14×14)



(b) Schematic diagram of corner map (8×8)

Figure 16: Schematic of the image to be detected and corner map obtained by quantum corner detection algorithm

5. Simulation of quantum fast corner detection algorithm

The proposed quantum algorithm is simulated in the IBM Q Experience platform. It is an online platform with several quantum devices and quantum simulators, which can be accessed using the Qiskit framework [26]. As the quantum simulator has 5000 qubits, we adopt it to perform our simulation.

The image to be detected is shown in Fig. 16(a). The size of the image is 14×14 and the range of its grayscale values is [0, 255]. A point on the upper left corner of this image is considered as an independent point. The middle area of this image is a square with 4 corners. The corner map can be obtained using the proposed quantum fast corner detection algorithm, which is shown in Fig. 16(b). The size of the corner map is 8×8, the grayscale value of each of its four corners is 0, and the grayscale value of other pixels is 1.

Figure 17 shows the probability histogram of the corner map in the quantum fast corner detection algorithm. Each element of the abscissa contains 7 qubits. For example, the first element is '0010111', where the first '0' indicates that it is a corner, and the remaining six qubits '010111' determine the position of the corner. The ordinate is the probability of its occurrence during the quantum measurement.

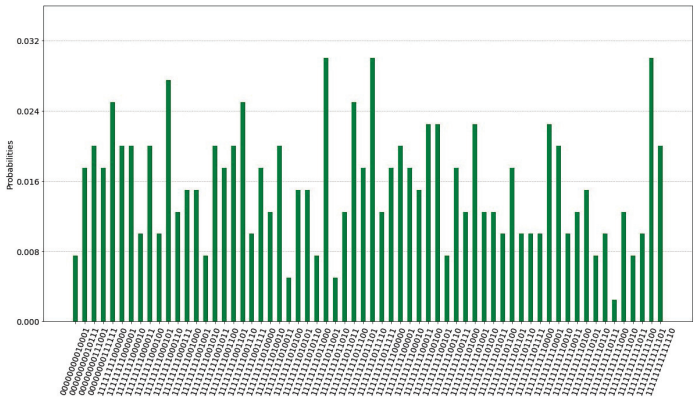


Figure 17: Probability histogram of corner map

6. Conclusions

We have proposed a new quantum algorithm, called as the quantum fast corner detection algorithm. Based on the reset operation, 16 auxiliary images and 4 marker qubits take full advantage of the quantum parallelism. Through the quantum-classical-quantum mode, quantum resources could be saved greatly. The proposed approach has been simulated using a quantum simulator through the Qiskit framework, and compared with the classical fast corner detection algorithm. The contributions made in this paper are summarized below:

1) An efficient quantum full subtractor is designed.

2) When preparing 16 auxiliary images, we make them to share grayscale information by using 4 auxiliary qubits, which can save a large number of qubits.

3) It is assumed that the size of the image is $2^n \times 2^n$, the range of grayscale values is $[0, 2^m - 1]$, and the second threshold is k_2. Under these considerations, it is found that the qubit cost of the quantum fast corner detection algorithm is $2m + 2n + 6$, time complexity is $33m + 102k_2 - 93$, and quantum delay is $29m + 102k_2 - 88$. Therefore, the time complexity and quantum delay are irrelevant to the size and number of images.

4) The time complexity of the quantum fast corner detection algorithm is exponentially lower than that of the classical fast corner detection algorithm.

In conclusion, the quantum fast corner detection algorithm proposed in this paper fills the gap in the design of quantum fast corner detection algorithm, which is the foundation of a quantum feature point extraction algorithm. Furthermore, the idea discussed in this paper will provide efficient solutions for many problems in the area of big data analysis. More interesting practical applications involving quantum image processing and machine learning can be explored based on the proposed algorithm.

References

[1] L. Gyongyosi, S. Imre, A survey on quantum computing technology, Computer Science Review 31 (2019) 51–71.

[2] Peter, W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM review 41 (2) (1999) 303–332.

[3] Grover, L. K, A framework for fast quantum mechanical algorithms, in: Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp. 53–62.
[4] E. Vincent, R. Laganière, Detecting and matching feature points, Journal of Visual Communication and Image Representation 16 (1) (2005) 38–54.
[5] W. Yang, L. Dou, J. Zhang, J. Lu, Automatic moving object detection and tracking in video sequences, in: SPIE Fifth International Symposium on Multispectral Image Processing and Pattern Recognition, Vol. 14, 2007, pp. 676–712.
[6] H. Yue, W. Chen, X. Wu, J. Liu, Fast 3d modeling in complex environments using a single kinect sensor, Optics and Lasers in Engineering 53 (2014) 104–111.
[7] X. Li, B. Yan, H. Wang, X. Luo, Q. Yang, W. Yan, Corner detection based target tracking and recognition for uav-based patrolling system, in: 2016 IEEE International Conference on Information and Automation (ICIA), IEEE, 2016, pp. 282–286.
[8] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: European conference on computer vision, Springer, 2006, pp. 430–443.
[9] C. Harris, M. Stephens, et al., A combined corner and edge detector, in: Alvey vision conference, Vol. 15, Citeseer, 1988, pp. 10–5244.
[10] S. W. Teng, R. M. N. Sadat, G. Lu, Effective and efficient contour-based corner detectors, Pattern Recognition 48 (7) (2015) 2185–2197.
[11] C. Wan, J. Cao, X. Wei, J. Huang, Z. Chen, D. Xu, F. Qiu, Ipcs: An improved corner detector with intensity, pattern, curvature, and scale, The Visual Computer (2022) 1–15.
[12] T. Lindeberg, Scale invariant feature transform, Scholarpedia 7 (5) (2012) 10491.
[13] J. Shi, et al., Good features to track, in: 1994 Proceedings of IEEE conference on computer vision and pattern recognition, IEEE, 1994, pp. 593–600.
[14] S. E. Venegas-Andraca, S. Bose, Storing, processing, and retrieving an image using quantum mechanics, in: Quantum Information and Computation, Vol. 5105, SPIE, 2003, pp. 137–147.
[15] P. Q. Le, F. Dong, K. Hirota, A flexible representation of quantum images for polynomial preparation, image compression, and processing operations, Quantum Information Processing 10 (1) (2011) 63–84.
[16] Y. Zhang, K. Lu, Y. Gao, M. Wang, Neqr: a novel enhanced quantum representation of digital images, Quantum information processing 12 (8) (2013) 2833–2860.
[17] S. Yuan, C. Wen, B. Hang, Y. Gong, The dual-threshold quantum image segmentation algorithm and its simulation, Quantum Information Processing 19 (12) (2020) 1–21.
[18] Barbosa, A. Geraldo, Quantum half-adder, Physical Review A 73 (5) (2006) 052321.
[19] H. Xia, H. Li, H. Zhang, Y. Liang, J. Xin, Novel multi-bit quantum comparators and their application in image binarization, Quantum Information Processing 18 (7) (2019) 1–17.
[20] M. Trajković, M. Hedley, Fast corner detection, Image and vision computing 16 (2) (1998) 75–87.
[21] E. Rosten, T. Drummond, Fusing points and lines for high performance tracking, in: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Vol. 2, Ieee, 2005, pp. 1508–1515.
[22] J. A. Smolin, D. P. DiVincenzo, Five two-bit quantum gates are sufficient to implement the quantum fredkin gate, Physical Review A 53 (4) (1996) 2855.
[23] M. Mohammadi, M. Eshghi, On figures of merit in reversible and quantum logic designs, Quantum Information Processing 8 (4) (2009) 297–318.
[24] K. W. Cheng, C. C. Tseng, Quantum full adder and subtractor, Electronics Letters 38 (22) (2002) 1343–1344.
[25] S. Yuan, S. Gao, C. Wen, Y. Wang, H. Qu, Y. Wang, A novel fault-tolerant quantum divider and its simulation, Quantum Information Processing 21 (5) (2022) 1–15.
[26] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C.-F. Chen, et al., Qiskit: An open-source framework for quantum computing, Accessed on: Mar 16 (2019).