Hadamard-gate Quantum Image Recognition: Theory and Experiment

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science in
Physics from the College of William and Mary in Virginia,

by

Yuqiao (Cordelia) Li

Prof. Chi-Kwong Li

Prof. Konstantinos Orginos

Prof. David Armstrong

Williamsburg, Virginia
May 16 2022

Contents

Abstract

Quantum computing has gained much attention in recent years due to its strong computational power. Specifically, quantum computing utilizes the ensemble property of quantum states to accelerate computations. In quantum computers, information is encoded in quantum bits (qubits). To perform operations on qubits, we design quantum circuits using different quantum channels. Previous studies have been done on image processing using quantum computing, like the Flexible Representation of Quantum Images. However, the image recognition algorithm in quantum computing is still an unsolved problem. In this work, we describe our quantum image recognition algorithm based on Quantum Hadamard Edge Detection and Image Recognition. Our algorithm can identify the similarity of two properly aligned images, and we tested our image recognition algorithm on IBM quantum computers with the Qiskit python package. In our experiment, our algorithm successfully identify the difference of two images. We also discuss the advantages, disadvantages, and possible improvements of our algorithm in this paper.

Chapter 1

Introduction

1.1 Background

In recent years, quantum computing has gained much attention due to its strong computational power for certain classes of problems. A quantum computer utilizes the ensemble property of quantum states and entanglement to accelerate computations. For example, currently, classical computers do not have enough power to factorize an arbitrarily large integer into prime numbers within polynomial time. However, as described in [1], quantum computers can factorize an integer by Shor's algorithm within polynomial time. One encryption method, called the RSA encryption, is based on factorization of large prime numbers. Thus, if a strong enough quantum computer could be built in the future, the RSA encryption method would no longer be secure to encrypt messages.

Even though quantum computing offers high computational power, it is currently hard to conduct many classical computational tasks in quantum computers. It is difficult to entangle a large number of particles, and errors and noise may significantly affect the measurement results and thus the fidelity of the calculations. In addition, designing quantum algorithms and building quantum circuits is hard, and different from classical computing. Therefore, more in-depth studies are needed in this field.

1.2 Basics of Quantum Computation

In classical computing, the binary bits (0 and 1) are used as the basic computational unit; in quantum computing, quantum bits, or *qubits*, are used, in analogy to binary bits. Specifically, a qubit is a unit vector in the vector space $\mathbb{C}^2$ represented under the basis vectors

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Also, a qubit can be in a *superposition* state:

$$|\psi\rangle = a|0\rangle + b|1\rangle,$$

where $a, b \in \mathbb{C}, |a|^2 + |b|^2 = 1$. The superposition state means that if we measure $|\psi\rangle = a|0\rangle + b|1\rangle$, the probability of getting state $|0\rangle$ is $|a|^2$, and the probability of getting state $|1\rangle$ is $|b|^2$.

As introduced in [1], quantum computation is a collection of a set of registers, a unitary matrix U, and measurements. When measurements have not been made, the time evolution of a state follows the Schrödinger equation

$$i\hbar \frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle,$$

where the time evolution is given by

$$|\Psi(t)\rangle = \hat{U}(t)|\Psi(0)\rangle.$$

Here, the unitary matrix U is a time-evolution operator: $UU^\dagger = U^\dagger U = I$. The set of all unitary matrices form a group, which is closed under multiplication and the taking of inverses. Therefore, we can decompose any unitary matrix U into multiple unitary matrices: $U = U_1 U_2 ... U_k$. Unitary matrices preserve length, which implies conservation of energy in a closed system.

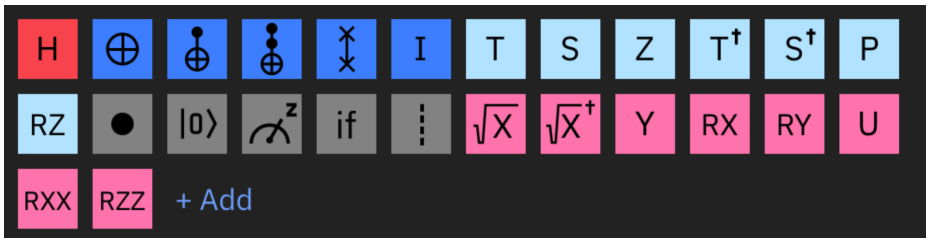

Figure 1.1: Qiskit gate library shown in quantum-computing.ibm.com.

As discussed in [1], in quantum circuits, quantum logic gates are unitary matrices. Therefore, we can achieve any desired unitary gate by decomposing them into basic unitary gates. For example, given a desired unitary operator U, we can decompose it as $U = U_1U_2...U_k$, where U_i's are the basic unitary gates. With these basic unitary gates, we can implement them in, for example, the IBM quantum computers. We will explain that in the next section.

Since the group of unitary matrices is closed under taking inverses, quantum logic gates are reversible. Given registers and quantum logic gates, measurements can be made to extract information. When measurements are performed, the quantum states collapse, which means we can no longer do any additional operation on the qubits. The result of measurement of each qubit will be either $|0\rangle$ or $|1\rangle$, with corresponding probability. Therefore, when we design a quantum circuit, we need to do all operations before we measure the circuit.

1.3 Basic Unitary Gates

In this section, we present the basic unitary gates for later reference. In our discussion, we will use the standard gate library from IBM Qiskit shown in Fig. 1.1.

1.3.1 Single Qubit Gates

Let $\mathbb{C}^n$ be the complex vector field with n rows. Since each single qubit is a vector in $\mathbb{C}^2$, the single qubit gates are 2-by-2 unitary matrices.

First, we present the Pauli X, Y, Z-gates.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Let us define $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Then, the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

changes $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$.

Additionally, we have the P, S, T-gates:

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}, S = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix}, T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}.$$

Also, these are all special cases of the U-gate:

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

where the θ, ϕ, λ are parameters that we can specify.

1.3.2 Entanglement of Qubits

When two pure state qubits are not entangled, their state vector can be expressed as *tensor product*. For two vector $v_1 = \begin{bmatrix} a \\ b \end{bmatrix}$ and $v_2 = \begin{bmatrix} c \\ d \end{bmatrix}$, the tensor product of v_1, v_2 is

$$v_1 \otimes v_2 = \begin{bmatrix} a \cdot \begin{bmatrix} c \\ d \end{bmatrix} \\ b \cdot \begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}.$$

For example, if we entangle $|0\rangle$ and $|1\rangle$, we obtain a tensor product

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

and we abbreviate this tensor product as $|0\rangle \otimes |1\rangle = |01\rangle$.

If two qubits are entangled, their state vector (a vector in $\mathbb{C}^4$) cannot be expressed in tensor product of two vectors in $\mathbb{C}^2$. To understand entangled qubits, we look at their density matrix:

$$\rho = \sum_i p_i \left|\Psi_i\right\rangle \left\langle\Psi_i\right|.$$

This density matrix ρ means that the probability of getting $|\Psi_i\rangle$ is $|p_i|^2$. When measurement is made, the qubits are no longer entangled. The probability for getting measurement result $|\Phi\rangle$ is $p(|\Phi\rangle) = \mathrm{Tr}(|\Phi\rangle\langle\Phi|\rho)$.

Since the state vector of 2 qubits is in $\mathbb{C}^4$, a 4-by-4 unitary matrix would be used as a 2-qubit gate. All gates involving more than 2 qubits can be decomposed to a series of unitary gates involving less than or equal to 2 qubits.

1.3.3 Multiple Qubit Gates

One of the most important 2-qubit gates is the CNOT-gate, which uses first qubit (called the control bit) as a control to decide whether to flip the second qubit (called the target bit) or not. As shown in Fig. 1.2, the first qubit q_0 is the control bit, and the second qubit q_1 is the target bit. If q_0 is $|0\rangle$, then this gate does nothing on q_1; if q_0 is $|1\rangle$, then q_1 is flipped as an X-gate is applied on q_1.

A CNOT-gate can be represented as the two following matrices, depending on which bit is the control bit:

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{or } \mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

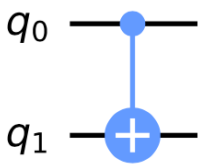

Figure 1.2: CNOT-gate

With the CNOT-gate and various single qubit gates, we can generate any kind of unitary gates, and we are ready to move on.

Chapter 2

Quantum Image Encoding

2.1 Overview of Image Recognition

Image processing and image recognition play important role in classical computer vision theory. In classical computing theory, we store the image in computers, apply machine learning algorithms, and let the computer recognize the image.

In the first step, images are stored as pixel values in classical computers, and high computational powers are needed to analyze these values. However, with quantum computers, we need many fewer qubits to store the same information. For example, given a 2^n-by-2^n square image, 2^{2n} classical bits are needed to store the information in classical computers. In Flexible Representation for Quantum Images described in the following, only $2n+1$ qubits are needed to store the image in quantum computers.

After storing the image, quantum algorithms are applied to classify the images. Namely, the tested image is compared to the sample image in multiple ways. After a series of comparisons, boundary values are set for the machines to classify the tested image. The details of classifications are further explained in Chapter 3.

In this chapter, we present two well-studied quantum image encoding methods. We make improvement on implementation of the first encoding method, and we will use the second encoding method for later chapters.

2.2 Flexible Representation for Quantum Images (FRQI)

2.2.1 Theory

As introduced in [2], *Flexible Representation for Quantum Images* (FRQI) offers a way to encode quantum images. In classical computing, images are stored as pixel representations, and FRQI is a similar method to pixel representations. Specifically, in FRQI, two types of information are stored: 1) the position of every pixel, and 2) the color of every pixel.

We would like to indicate that we will be using grayscale as colors in our later description. However, our θ is a continuous value in FRQI, so we can also map all colors to this continuous value $\theta \in [0, \pi/2]$ according to different colors' wavelengths. In that way, we can also encode all colors.

Given a 2^n-by-2^n image, we integrate the information of this image into a quantum state $|I\rangle$ as the following:

$$|I\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} |c_i\rangle \otimes |i\rangle,$$

where

$$|c_i\rangle = \cos\theta_i|0\rangle + \sin\theta_i|1\rangle.$$

Here, $|i\rangle$ encodes the position of each pixel; $\theta = (\theta_0, \theta_1, ..., \theta_{2^{2n}-1}), \theta_i \in [0, \frac{\pi}{2}]$ is a vector that encodes the color of each pixel; $|0\rangle$ and $|1\rangle$ are 2-dimensional basis states as introduced in Section 1.1. We show an example of a 2-by-2 image with its FRQI encoding state in Fig. 2.1.

2.2.2 Circuit Implementation and Improvement

The Qiskit Textbook [3] written by the IBM Qiskit team provides a sample quantum circuit to encode the picture in Fig. 2.1. Their implementation scheme is shown in

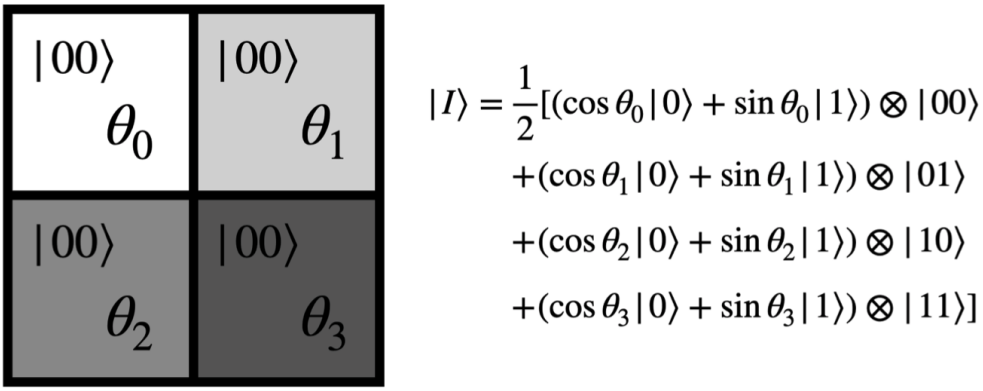

Figure 2.1: An example of FRQI encoding of a 2-by-2 image.

Fig. 2.2. In Fig. 2.2, we can see the X-gates, H-gates, CNOT-gates, and the R_y-gate, which is the following:

$$R_y = e^{(-i\frac{\theta}{2}Y)},$$

where each θ is individually specified in $\{\theta_0, \theta_1, \theta_2, \theta_3\}$ as in the picture. After the gates, the black boxes with arrows are measurements.

As mentioned in Section 1.1, errors and noises may largely affect the measurement results. Among all basic gates provided by IBM `Qiskit` [3], the *controlled NOT* gate, or CNOT gate, introduce the most noise. In [4], a detailed analysis of the level of noise of the CNOT gate using stochastic differential equations is provided.

In current quantum computers, due to technology deficiency, errors in the measurement results largely come from gate noise. Therefore, to let the quantum computers carry out the tasks with less error and get an accurate experimental result, we use as few control gates as possible in quantum algorithms.

In quantum circuits, control gates are represented as solid dots. In IBM's implementation scheme [3], to encode a 2-by-2 image, 20 control gates are used as depicted in Fig. 2.2. Here, we offer a better way to encode a 2-by-2 FRQI image as shown in

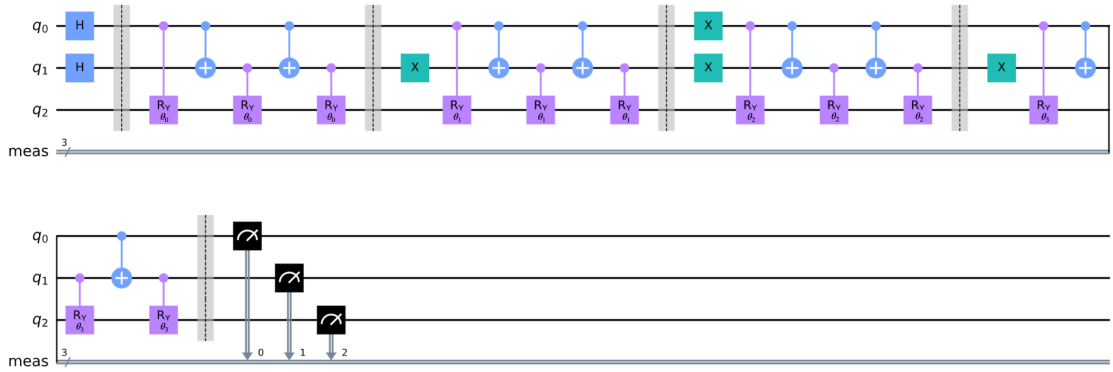

Figure 2.2: FRQI implementation scheme provided by [3]

Fig. 2.3.

In the IBM Qiskit library, the Hadamard gate (denoted H) has the following operation:

$$|0\rangle \rightarrow \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}$$
$$|1\rangle \rightarrow \frac{(|0\rangle - |1\rangle)}{\sqrt{2}};$$

the rotation gate $R_y(\theta)$ is a rotation along the y-axis:

$$R_y = e^{(-i\frac{\theta}{2}Y)},$$

where Y is the Pauli matrix,

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

In our circuit shown in Fig. 2.3, $R_{y,1} = R_y(\theta_0) = e^{(-i\frac{\theta_0}{2}Y)}$, $R_{y,2} = R_y(\theta_1 - \theta_0) = e^{(-i\frac{\theta_1-\theta_0}{2}Y)}$, $R_{y,3} = R_y(\theta_2 - \theta_0) = e^{(-i\frac{\theta_2-\theta_0}{2}Y)}$, $R_{y,4} = R_y(\theta_3 - \theta_1) = e^{(-i\frac{\theta_3-\theta_1}{2}Y)}$.

In our method, only 2 single control gates ($R_{y,2}$ and $R_{y,3}$) and 1 double control gate ($R_{y,4}$) are used. As shown in [5], a double control gate can be decomposed into basic gates with 6 single control gates. Therefore, our encoding scheme is made up of 8 control gates and introduces less noises to the system.

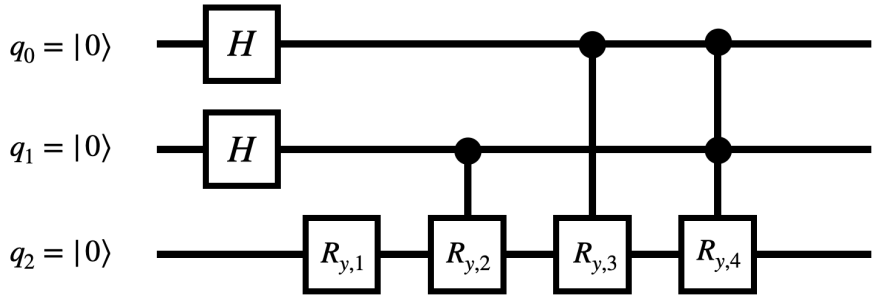

Figure 2.3: Our improved FRQI implementation scheme with fewer control gates [3]

2.3 Quantum Probability Image Encoding (QPIE)

Besides FRQI, [6] and [3] introduce another way of quantum image encoding, namely, the *Quantum Probability Image Encoding* (QPIE). In particular, instead of separately storing the colors and positions of the pixels, QPIE utilizes the probability amplitudes of a quantum state to store the pixel values of a classical image [3].

As described in [3], for an image with N pixels, we need

$$m = \lceil \log_2 N \rceil$$

number of qubits to represent this image. Given an n-by-n picture, we can view the image in terms of the pixel intensities I :

$$I = (I_{yx})_{n \times n},$$

where I_{yx} is the pixel intensity of position (x, y). Then, we normalize this vector I and get a normalization parameter c_i for all pixels:

$$c_i = \frac{I_{yx}}{\sqrt{\sum_{I_{yx}^2}}}.$$

For easier notation, let N be the number of qubits we need in this implementation.

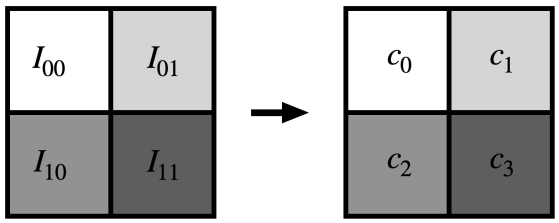

Figure 2.4: An example of a 2-by-2 image under QPIE representation

Then, the state representing this image, $|\text{Img}\rangle$, is represented as

$$|\text{Img}\rangle = \sum_{i=0}^{N} c_i |i\rangle,$$

where c_i is the normalization parameter described above, and $|i\rangle$ represents the position of each pixel.

We provide an example in Fig. 2.4 to illustrate a 2-by-2 image under QPIR representation. We start with image intensity $I = (I_{00}, I_{01}, I_{10}, I_{11})$, and c_i for $i = \{0, 1, 2, 3\}$ are the normalization parameters. The resulting encoded state of this image is

$$|\text{Img}\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle.$$

In the next section, we will further utilize the QPIR representation to conduct image recognition under a simplified setting.

Chapter 3

Quantum Image Recognition

3.1 Quantum Hadamard Edge Detection (QHED) and Image Recognition

In [6], an edge detection algorithm is provided to identify the boundary points in an image. We make use of this edge detection algorithm and develop an algorithm to determine if two images are similar. First, we introduce the edge detection algorithm: *Quantum Hadamard Edge Detection (QHED).*

In classical computing, the worst case of edge detection requires the computer to consider each individual pixel, and the complexity for most of them is $\mathcal{O}(2^n)$. However, in quantum computers, edge detection can be much easier. More specifically, the algorithm utilizes the action of H-gate, and it has complexity $\mathcal{O}(1)$.

In QHED, a more sophisticated way (using an auxiliary qubit) to implement the Hadamard gate algorithm is provided; details can be found in [6]. In the following, we describe a simplified QHED algorithm without the auxiliary qubit.

Given an N-pixel image, we first encode our picture by the QPIE representation described in Section 2.3, and we get an encoded state of $|\,\mathrm{Img}\rangle = \sum_{i=0}^{N-1} c_i|i\rangle$. Let $I_{2^{n-1}}$ be an identity matrix with size 2^{n-1}. Then, we apply a Hadamard gate to the

last qubit, which has the following unitary matrix:

$$I_{2^{n-1}} \otimes H_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix}.$$

After applying the Hadamard gate, we obtain the following vector:

$$(I_{2^{n-1}} \otimes H_0) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} \rightarrow \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{bmatrix}.$$

Note that the entries $(c_i - c_{i+1})$ is the difference between two consecutive pixels. Also, all entries of the form $(c_i - c_{i+1})$ are located at the even entries. Therefore, after measurement of all qubits, we only need to look at cases where the last qubit is 1 (i.e. only look at cases in form of $|* ... * 1\rangle$). If the difference is large, then we admit that there is an edge between the two consecutive pixels.

In [6], the author presents a computer simulation result of the edge detection of a picture in Fig. 3.1.

3.2 Applications to Image Recognition

We then describe how can this simplified QHED be applied to detect whether two images are similar. Given a sample image and a tested image, we cut the 2-D image into horizontal and vertical 1-D stripes. Then, we stack the stripes together, with stripes from the tested image and sample images interlacing with each other.

Then, we use QHED to detect the difference between horizontal stripes and vertical stripes. An illustration is included in Fig. 3.2. Here, $\{a, b\}$ represents the rows

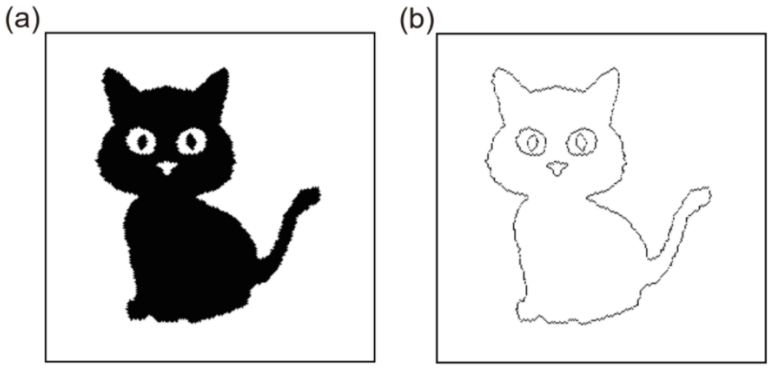

Figure 3.1: An example of edge detection algorithm

of the sample image, and $\{c, d, e, f\}$ represents the columns of the sample image. Similarly, $\{1, 2\}$ are the rows of the tested image, and $\{3, 4, 5, 6\}$ are the columns of the tested image.

After stacking the rows and columns vertically and horizontally, we conduct QHED on both cases. The arrows between stripes indicate the consecutive pixels for which comparisons are made. Therefore, we can know pixel-by-pixel what positions are different between the two images.

We can also define a threshold value to determine if two images are similar. If the differences are large for several positions, then we may claim that the two images are not similar.

Furthermore, rotations may be used on images with the rotation matrix:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Specifically, for a pixel at the (x, y) position of the image, we may conduct rotation by multiplying this rotation matrix on the left of the vector $\mathbf{v} = \begin{bmatrix} x \\ y \end{bmatrix}$:

$$R\mathbf{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}.$$

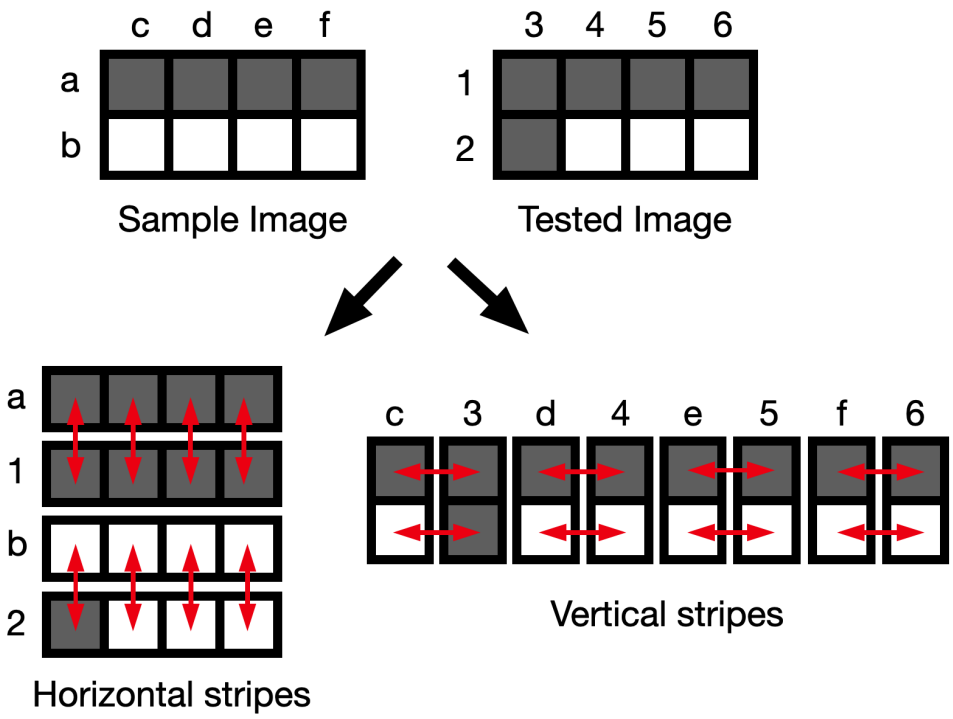

Figure 3.2: Illustration of horizontal and vertical stripe cutting

Chapter 4

Experiments

In this chapter, we present our experimental result by using the IBM Qiskit Quantum Computer. IBM Qiskit is a python package that enables people to conduct experiments on IBM quantum computers from everywhere. To first encode the image, we can easily make the initial state to the QPIE state by the initialization function in IBM Qiskit [3].

4.1 Theory of Superconducting Quantum Computer

Superconducting quantum computer uses superconducting electronic circuit to implement quantum computing. In a superconductor, the basic charge carriers are pairs of electrons, *Cooper pairs*, instead of single electrons. Cooper pairs are bosons, with integer total spin, and they occupy a single quantum energy level. At a very low temperature, all bosons occupy the lowest quantum state, and this effect is known as the *Bose-Einstein condensate*.

In a superconducting quantum computer, there are tree types of qubit archetypes: 1) phase; 2) charge; 3) flux. Here, the two states of qubit, $|0\rangle$ and $|1\rangle$ are mapped to different states of the physical systems, like the quantized energy levels, or the quantum superpositions. For example, the charge qubit archetype uses different energy level of the system, which is an integer number of Cooper pairs on a superconducting

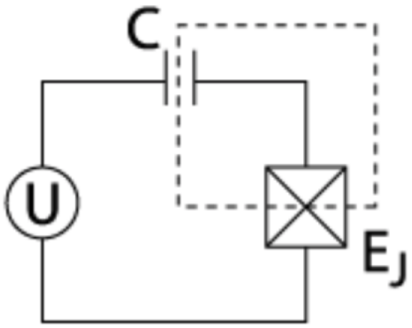

Figure 4.1: A charge qubit archetype

island. Fig. 4.1 shows a charge qubit, with the dashed area as the superconducting island.

Every single qubit gate is a rotation, which can be done by microwave pulses. When two qubits are coupled, they are connected by an intermediate electrical coupling circuit, and this can be done by capacitors.

4.2 Experiment with IBM Qiskit

Before we start to explain our experiment result, we would like to remark that IBM Qiskit is a open access python package that enables the general public to conduct quantum computation in IBM's quantum computers. In every experiment conducted, 1024 rounds of measurements are done, and the result is presented in bar plot, and the number on the top shows the normalized probability of a certain state shows up in the measurement. For example, if 0.5 shows up on a bar chart of $|10\rangle$, then out of 1024 experiments, there are a total of 512 experiment that ends up with measurement result $|10\rangle$.

4.2.1 Simple case with a 2-by-2 image

First, we perform our algorithm on a simple case with a 2-by-2 image shown in Fig. 4.2, and we wonder if IBM Qiskit quantum computer can identify the difference between pixels. Specifically, this image corresponds to the following matrix:

$$\begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

Then, we conduct a horizontal scan, which means we compare c_1, c_2, and we compare c_3, c_4.

The circuit implementation of this 2-by-2 image is shown in Fig. 4.3, where the "init" block is the initialization function in Qiskit, which helps us prepare the initial state. Then, the IBM Qiskit simulation result of this circuit is presented in Fig. 4.4.

In the measurement result, the $|10\rangle$ position corresponds to $c_3 + c_4$, and the $|11\rangle$ position corresponds to $c_3 - c_4$. Since we only care about the difference between pixels, we ignore $|10\rangle$ and take only $|11\rangle$ into account. We would expect $|01\rangle$ to be zero, and $|11\rangle$ to be nonzero, so the simulation matches our expectation. Therefore, we are ready to conduct actual experiment using picture in Fig. 3.2.

4.2.2 Experiments using IBM Qiskit

Here, we compare the horizontal stripes and vertical stripes according to the scheme shown in Fig. 3.2. The encoded image of horizontal scan is shown in Fig. 4.5, and the encoded image of vertical scan is shown in Fig. 4.6.

The results of the measurement made by IBM Qiskit quantum computers are shown below. Fig. 4.7 is the result for horizontal scan, and Fig. 4.8 is the result for vertical scan. When each task is sent to Qiskit quantum computers, there are 1024 measurements done in the quantum computers, and the resulting chart shown the percentages of each state appeared in measurements among all of the possible states

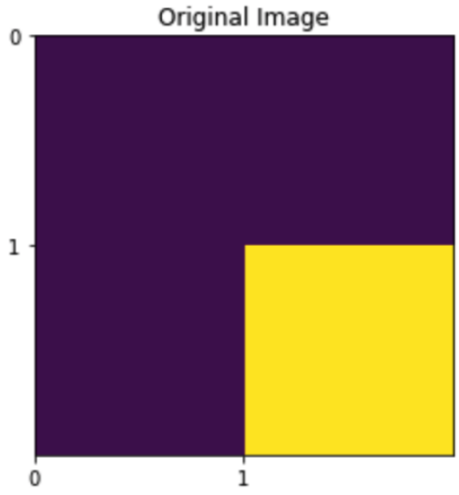

Figure 4.2: A 2-by-2 image to test our algorithm.

Figure 4.3: The circuit of testing the 2-by-2 image in Fig. 4.2.

in 1024 measurements. Therefore, reading from the height of the result graph, we can roughly know the probability of obtaining each state from a certain circuit diagram.

From the horizontal scan result, we can see that the $|0001\rangle$ state stands out from all other states with ending qubit = 1. From the vertical scan result, we can see that the $|0101\rangle$ state stands out by a small amount from all other states with ending qubit = 1. We do not expect to see $|1101\rangle$ being the largest among all states with last qubit = 1, but $|1101\rangle$ is actually the largest. Therefore, there are large errors within measurements.

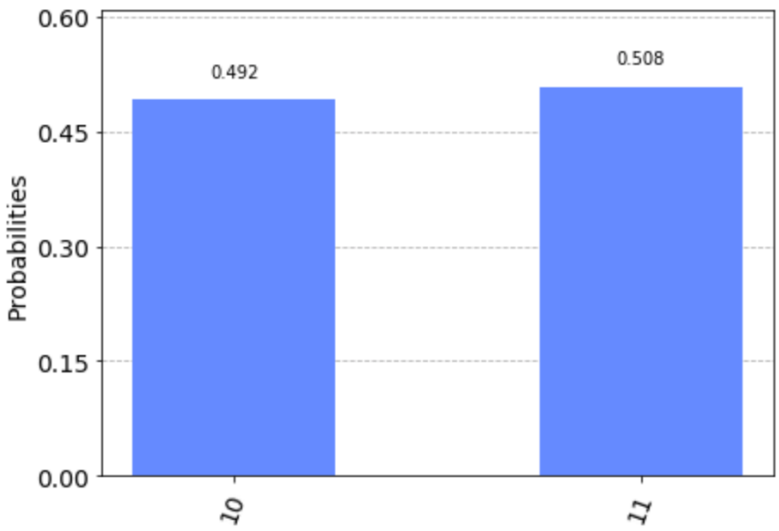

Figure 4.4: The IBM Qiskit simulation of the circuit shown in Fig. 4.3.

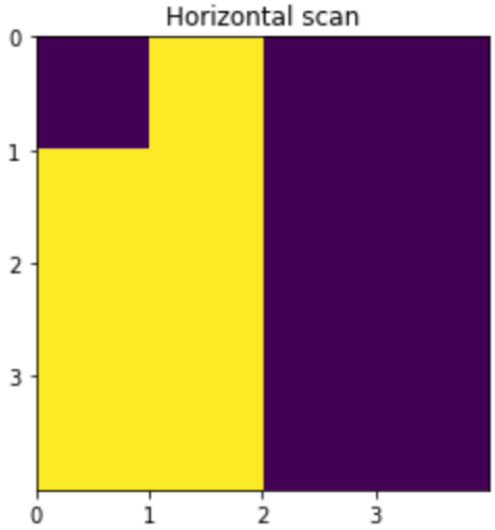

Figure 4.5: Horizontal scan according to Fig. 3.2

4.3 Improvements

Since we encode the image as a vector with entries $(c_i + c_{i+1})$ and $(c_i - c_{i+1})$, when we do measurement each time, there may be high possibility that a state would fall

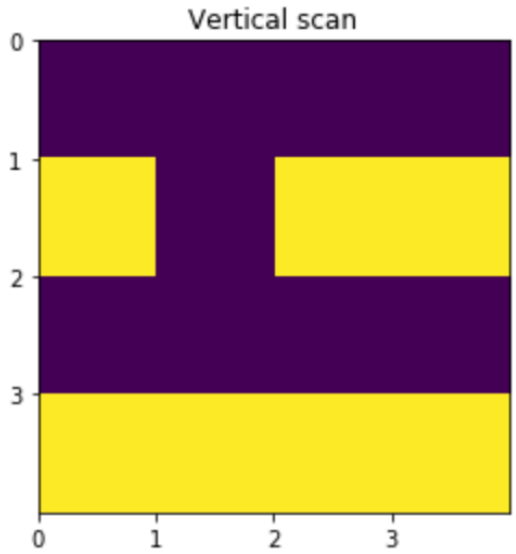

Figure 4.6: Vertical scan according to Fig. 3.2

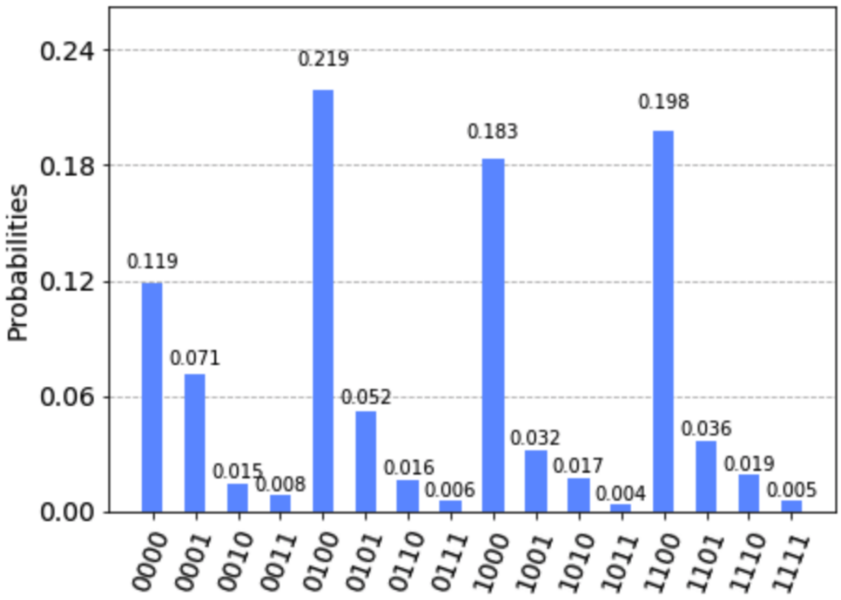

Figure 4.7: Horizontal scan result.

into $(c_i + c_{i+1})$. Since the encoding method gives us a vector with all nonnegative entries, we have $c_i + c_{i+1} > c_i - c_{i+1}$. Therefore, it may not be easy to directly see the

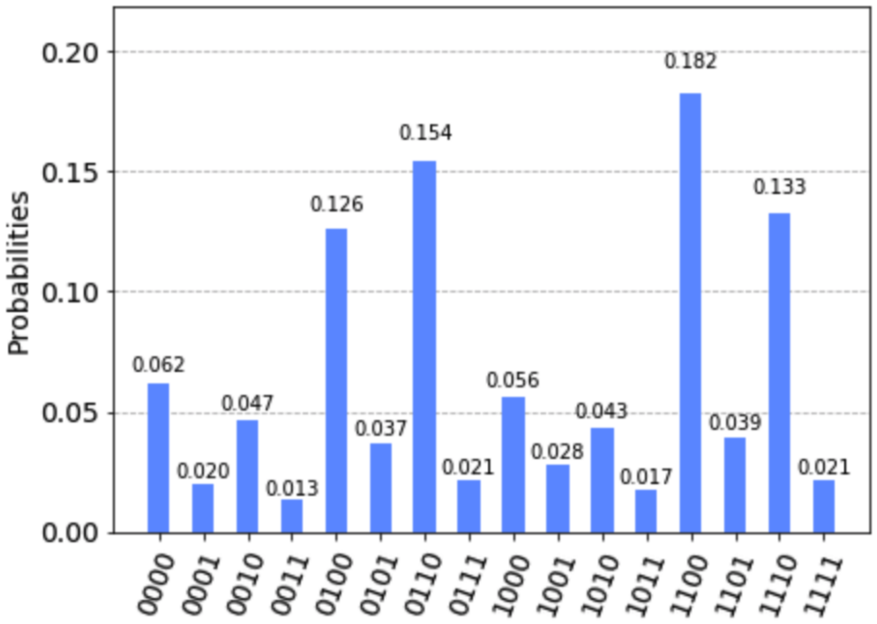

Figure 4.8: Vertical scan result.

difference between c_i and c_{i+1} in our measurement.

Therefore, we would like to have a way to ignore the large positive entries (c_i+c_{i+1}). In our pre-treatment of the image, we can do modulo-2 arithmetic to reduce the amplitude of the positive entries $(c_i + c_{i+1})$. For example, if we want to compare two similar pixels $c_1 = 1$ and $c_2 = 1$, then $c_1 + c_2 = 0 \mod 2$, and $c_1 - c_2 = 0$. Therefore, we can reduce the amplitude of the positive entries $(c_i + c_{i+1})$, and we can better see the difference in $(c_i - c_{i+1})$.

Chapter 5

Conclusion

Here, we would like to discuss the advantage and disadvantages of our algorithm.

5.1 Advantages

Our image recognition algorithm has complexity $\mathcal{O}(1)$, since it only uses one Hadamard gate. Thus, this algorithm is fast to compare two images and determine whether they are similar. Even though it is a simple algorithm, it is powerful and has result better than many other image encoding methods.

5.2 Disadvantages and Possible Improvements

Since our algorithm compare two images by taking horizontal and vertical stripes, it requires some pre-treatment to the sample and tested images. Therefore, it is a bit complicated. It would be easier if we can directly encode two images into the quantum circuit and compare horizontal and vertical stripes at once.

One possible improvement of this algorithm is conducting more rounds of experiments (more than 1024 rounds each time), and we may get a clearer result among all measurements with last qubit = 1. Also, we can set a range for each pixel. For example, if the intensity ¿ 0.5, we set it to 1; if intensity ¡ 0.5, we set it to 0. Then, the edge detection algorithm can easily tell the difference between 1 and 0. This method

would work best if the pictures are almost clear black and white.

Also, when we encode images into quantum state, we only use real numbers as entries here. However, we can also use complex number with real and imaginary parts. The probability of getting this state as a measurement result is the square of the absolute value of the complex number. Therefore, we may have more freedom or develop a better algorithm when we take imaginary parts into account.

References

[1] M. Nakahara and T. Ohmi. *Quantum Computing: From Linear Algebra to Physical Realizations*. CRC Press, 2008. ISBN: 978-0-7503-0983-7.

[2] F. Yan and S. E. Venegas-Andraca. *Quantum Image Processing*. Springer, 2020. ISBN: 978-981-32-9330-4.

[3] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[4] Angelo Bassi and Dirk-André Deckert. "Noise gates for decoherent quantum circuits". In: *Phys. Rev. A* 77 (3 2008), p. 032323.

[5] Vivek V. Shende and Igor L. Markov. *On the CNOT-cost of TOFFOLI gates*. 2008. arXiv: 0803.2316 [quant-ph].

[6] Xi-Wei Yao et al. "Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment". In: *Phys. Rev. X* 7 (3 2017), p. 031041.