Creating Hybrid Images Using a Quantum Computer

By

Thomas Farina

Matthew Anderson (Advisor)

* * * * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Computer Science

UNION COLLEGE

May, 2021

Abstract

THOMAS FARINA Creating Hybrid Images Using a Quantum Computer. Department of Computer Science, May, 2021.

ADVISOR: Matt Anderson

In this thesis we examine at how hybrid images are created on a quantum computer. We wish to find how feasible it is to create a hybrid image, seeing if there are any difficulties in creating it. We start with a background of what hybrid images are and how a quantum computer works at a basic level. We then look at the possible image representations that are used, as well as different filtering algorithms to implement. An attempt is made to implement a filtering algorithm on a simulated quantum computer, but did not completely match the results of the earlier work.

Contents

List of Figures

List of Tables

1 Introduction

One of the most important topics that computer scientists study is image processing. It is a crucial part of computer vision, an area that helps computers interact with the world itself, with algorithms being implemented in self-driving cars, x-rays and MRI's [10]. It has led to some useful applications such as face detection and object recognition. One of the more interesting applications of computer vision is creating hybrid images. Hybrid images are a combination of two images, where one image can be seen close up while the other image can be seen from a distance [9]. An example of this is shown in Figure 2. Some applications of hybrid images include creating textures that can be seen from different viewing distances as well as unique messages seen from certain distances.

Although the field is well developed and has applications in the real world, one of the next big steps for computer vision is applying it with quantum computers. Quantum computers were first proposed by physicist Richard Feynman and are seen as faster versions of modern computers [3]. This combination is called *quantum image processing*. Figure 1 outlines the 3 steps that are considered when a quantum computer processes an image. The steps are:

1. Store the data of the image by translating it from the classical system to the quantum system. This is also known as quantum image preparation and has multiple ways of being represented, with each method of storing the image commonly called an image representation.

2. Process and analyze the quantum image using a quantum algorithm. Some existing classical algorithms have been reformulated for quantum computers.

3. Once the algorithm is done, the output is then translated back to the classical system, where it can be used or displayed as normal.

There are a few reasons why quantum image processing should be studied and developed. One of the reasons is that the current algorithms for computer vision take a long time to run. One example is that Google's TensorFlow can take 65 hours using 100 GPU's to train the machine learning algorithm for image classification, with the algorithm itself having 78% accuracy [7]. Tasks like this could be sped up using quantum computing. Another reason why this is important is that quantum computers need images. Eventually quantum computers will be used in conjunction with classical computers and will need algorithms to verify that quantum computer works. Creating these algorithms will help make this task possible once quantum computers are ready to take the next step.

The main purpose of this thesis is to see if it is feasible to create a hybrid image on a quantum computer. For this research, the main goal is to see how easy it can be to create a hybrid image, taking into account how

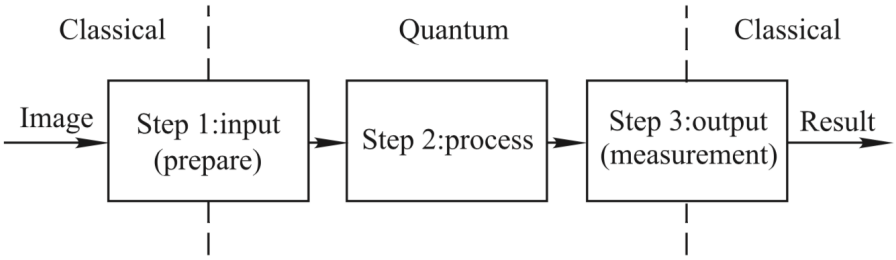


Figure 1: How quantum image processing works. The image is first transformed into a representation that a quantum computer then processes and finally returns to the classical system where the image is returned.[1].

Figure 2: An example of a hybrid image of a jaguar and an elephant. Upon seeing this it should look like a jaguar. Standing back a few feet and/or squinting will help see the elephant in the photo [9].

the image is represented and how the filtering is done. The algorithm is ran using a simulated quantum computer, to perform the same steps as quantum image processing.

What makes this thesis unique is that there have been no previous studies on this specific topic to date. Hybrid images are not widely used in computer vision and researchers have not attempted to try it in a quantum computer. This would be an interesting application to see how filtering is done on an image in a quantum computer. It is also be interesting to see how it would affect the different image representations and whether or not each representation needs a unique hybrid image algorithm. After implementing a filtering algorithm in a quantum computer, I was able to filter an image but did not get the same results as the authors of the filtering algorithm.

In this thesis, Section 2 talks about the necessary background information, talking about how hybrid images are made, what quantum computing is, how circuits are shown in the a quantum computer, and the

different image representations shown throughout. Section 3 talks about the different filtering methods that are done in a quantum computer. Section 4 explains what methods I used to simulate a filtering algorithm on a circuit and how it was implemented. Section 5 talks about what I got from my experiments, with Section 6 wrapping up the work done over the past few terms and a future direction on where this project can go.

2 Background

Before looking at the different ways that an image can be filtered in a quantum computer, we first look at how hybrid images are created and what a quantum computer is. We then see all of the different image representations that are used later on to apply to the different filtering methods.

2.1 Computer Vision/Hybrid Images

Computer vision is an area of study that focuses on how computers interact with the physical world. The primary goals is to interpret the world that humans see using various images while trying to reconstruct its many properties like color, shape, and size [10]. There are many different areas of study within computer vision but the main area this paper focuses on is image processing. In this case an image with a size of $n \times m$ is a 2-D collection of pixels with a length of n pixels and a width of m pixels. In general, image processing is the transformation or modification of the image at the pixel level to give a desired output. This is considered the first process in computer vision for most applications since it converts the image to something that the computer can use and do operations on.

One of the operations that a computer can do on an image is create a hybrid image. Hybrid images is a technique that creates a new image based on two images of the same size [9]. Both of the input images are in the hybrid image, but each image can only be seen by looking at it from a certain distance. This is due to the certain frequencies that an image displays. A *frequency* of an image is the rate of change of the color value with respect to its space. To get information about the frequencies of an image, we apply a *Fourier transform*. When looking at an image, *low frequencies* look at large features and the form of an object, while *high frequencies* show smaller features and the details of an object. In order to obtain these frequencies from an image, a *filter* is used to suppress unnecessary information. A *low-pass filter* is used to get the low frequencies, while a *high-pass filter* is used to obtain the high frequencies. To get a hybrid image from two images, one must apply a low-pass filter on one image and a high-pass filter on the other image. Combining these two results in a hybrid image, where the low frequencies are seen from a distance and

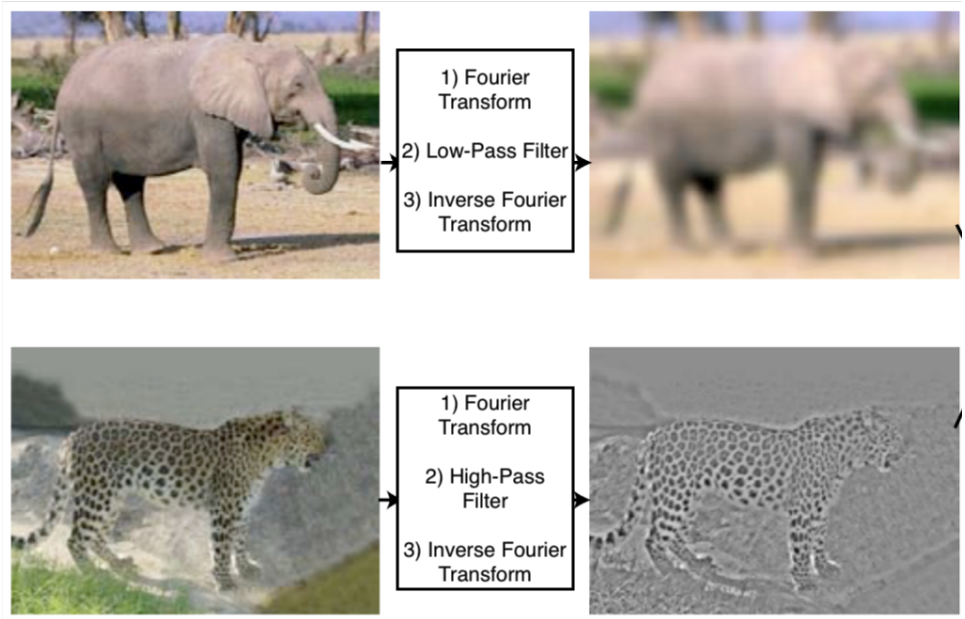


Figure 3: How a hybrid image is created. To get the low frequencies of the elephant, a Fourier transform is applied on the image, followed by a low-pass filter to get the desired frequencies, and an inverse Fourier transform is applied to observe the low frequencies. The steps are similar for obtaining the high frequencies of the jaguar, with the filter being a high-pass filter instead. Merging the two results in Figure 2 [9].

HYBRIDIMAGE($image1, image2, filter$)
1 $lowPass =$ apply $filter$ to $image1$
2 $highPass =$ apply $filter$ to $image2$
3 $highPass = image2 - highPass$
4 $hybridImage = lowPass + highPass$
5 **return** $hybridImage$

Figure 4: Pseudo-code to create a hybrid image given two images and a filter.

the high frequencies are seen up close. Figure 3 shows this process for two images, which is the resulting hybrid image from Figure 2.

A pseudo-code of the algorithm is shown in Figure 4. The input for the algorithm is two images of the same size and a filter to use, which is generally a low-pass filter. Line 3 uses the fact that subtracting the low-pass filtered image from the original results in a high-pass filtered image.

2.2 Quantum Computing

Currently, the computers used today utilize the bit, the basic unit of information. A bit can only be in two states, 0 or 1. However in quantum computing, the idea is that we use a quantum bit as the basic unit of information, known as the qubit. Instead of having two states, a qubit can be in some combination of 0 and

1 at the same time, which is known as a *superposition*. For one qubit, we can describe this as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,.$$

Note that $|\rangle$ represents a state of a qubit with α, β as complex numbers [8]. The states $|0\rangle$ and $|1\rangle$ are known as computational basis states and are the main states of a qubit. When measuring the states of a qubit we can only get two possible states, either 0 with probability $|\alpha|^2$ or 1 with probability $|\beta|^2$. Since they are probabilities, this means that

$$||\,|\psi\rangle\,|| = |\alpha|^2 + |\beta|^2 = 1.$$

The notation $||$ means the magnitude of a complex number, which for a given complex value $z = a + bi$ is found by the equation $|z| = \sqrt{a^2 + b^2}$. We can generalize this for a $m-$array qubit with m different basis states as

$$|\psi\rangle = \sum_{i=0}^{m-1} \alpha_i\,|i\rangle \quad \text{with} \quad ||\,|\psi\rangle\,|| = \sum_{i=0}^{m-1} |\alpha_i|^2 = 1.$$

By doing this, it creates a larger problem space for a qubit to use compared to the bit. As an example of this, a 4-bit piece of data needs 64 total bits to store all possible values from 0000 to 1111, since there are 16 possible values to store, each requiring 4 bits to save each value. In a quantum computer, this can be accomplished with only 4 qubits since every qubit acts like a 0 and a 1 and can take on every value from 0000 to 1111 due to superposition [1]. In general, given n qubits with two states, they can represent 2^n possible combinations of 0 and 1.

To use these qubits there are *quantum logic gates*. They are the basic operations of quantum computers. They are similar to what classical logic gates are but are made specifically to manipulate qubits.

2.3 Quantum Circuit Diagrams

When describing what qubits do when executing an action, it is common to represent it by drawing a circuit diagram to show what is happening. In a quantum circuit, the initial state of the qubits are shown on the left, while the resulting states are shown on the right. What is between them are the gates that describe the actions being executed.

Figure 5 shows two common gates used when designed a quantum circuit. A NOT gate acts like a standard not gate in the classical system, flipping the value by either changing 0 to a 1 or vice versa. A

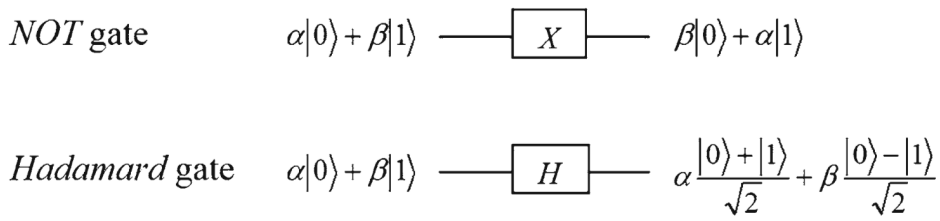


Figure 5: A NOT gate and a Hadamard Gate

Input		Output	
Control	Target	Control	Target
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Figure 6a: A table showing all of the possible inputs and outputs for the C-NOT gate

Hadamard gate is unique in the sense that it acts on one qubit and creates an equal superposition of the basis states, which gives the states an even probability of being either 0 or 1. This is commonly used to initialize qubits. The last important quantum gate is the Controlled NOT gate, commonly referred to as the C-NOT gate. It uses two qubits and they are designated as the control bit and the target bit. The target bit will flip if and only if the control bit is $|1\rangle$. The table in Figure 6 shows the possible outcomes for two qubits passed through. Another gate that is used is a controlled rotation gate, which is a 2-D rotation of the x and y coordinate in 3 dimensions, which in this case is rotated about the z-axis.

One important thing to note here is that all quantum logic gates can be seen as operators and be represented as matrices, as shown in Figure 6. This means that these operators can be done via matrix multiplication, which is very useful when applying these operators to different image representations. More importantly, all of the matrices are *unitary*, meaning that every matrix multiplied by its conjugate transpose is equal to the identity matrix. In other words, for a given matrix A, we have $AA^\dagger = I$, where $A^\dagger$ represents the conjugate transpose. Figure 7 displays how the conjugate transpose is done for any matrix.

2.4 Quantum Image Representation

Over the years there have been many different representations of how quantum computers process and store images from classical machines. Some examples of the various quantum image representations are FRQI and NEQR and are used in some of the quantum image processing algorithms as well. Note that to keep the representation as simple as possible, the image will be represented as a grayscale image, which is an image where each pixel represents the amount of light it receives, with white being the brightest and

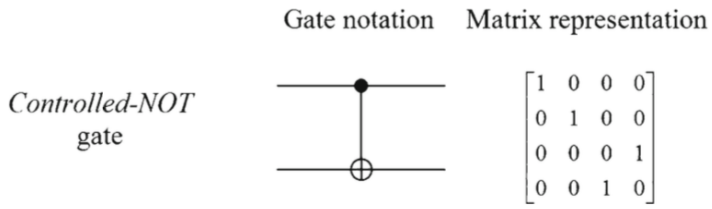


Figure 6b: A C-NOT Gate. In this diagram, the two horizontal lines represent the two different qubits. The black dot on the top line is the control qubit, while the circle on the bottom line represents the target qubit, with the result being stored on that qubit as well [5].

$$A = \begin{bmatrix} 1 & -2-i & 5 \\ 1+i & i & 4-2i \end{bmatrix} \implies A^T = \begin{bmatrix} 1 & 1+i \\ -2-i & i \\ 5 & 4-2i \end{bmatrix} \implies A^\dagger = \begin{bmatrix} 1 & 1-i \\ -2+i & -i \\ 5 & 4+2i \end{bmatrix}$$

Figure 7: How to take the conjugate transpose of a matrix with complex-valued entries. The transpose is taken first, followed by the complex conjugate of all the entries in the matrix.

black being the darkest.

2.4.1 FRQI

This model is the first modern representation that has become the most popular form to represent images in a quantum computer. The FRQI (Flexible Representation of Quantum Images) model [5] is based in the idea of using an angle θ to represent the color information. For a given 2-D image that has the same dimensions, which are $2^n \times 2^n$, $2n$ qubits are used to store the different pixel positions in the image, and only one qubit is used to store the grayscale value of those pixels. This makes this image representation use a total of $2n+1$ qubits to store an image. Here we have $\theta = (\theta_1, \theta_2, \ldots, \theta_{2^{2n}-1})$ representing a vector of angles that encode the colors. Note that

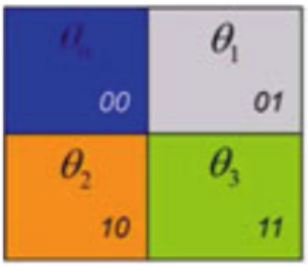
$$\theta_i \in \left[0, \frac{\pi}{2}\right], i = 0, 1, \ldots, 2^{2n} - 1$$

The state $I(\theta)$ stores the state of the image and is given by

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} \left(\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle\right) \otimes |i\rangle .$$

For this we have θ_i storing the color information, $\cos(\theta_i)|0\rangle + \sin(\theta_i)|1\rangle$ encodes the information about the colors, and $|i\rangle$ stores the position information of the image. This state is normalized, which means that $|| \, |I(\theta)\rangle \, || = 1$. Figure 8 shows an example image and the state that the FRQI model will be in.

In order to transform an image from the classical system to the quantum system, there are two steps to



$$|I\rangle = \frac{1}{2}[(\cos(\theta_0)\,|0\rangle + \sin(\theta_0)\,|1\rangle) \otimes |00\rangle) + (\cos(\theta_1)\,|0\rangle + \sin(\theta_1)\,|1\rangle) \otimes |01\rangle)$$
$$+(\cos(\theta_2)\,|0\rangle + \sin(\theta_2)\,|1\rangle) \otimes |10\rangle) + (\cos(\theta_3)\,|0\rangle + \sin(\theta_3)\,|1\rangle) \otimes |11\rangle)]$$

Figure 8: An example image and its corresponding FRQI state. For each color θ_i, the position i is stored at the same state.[5].

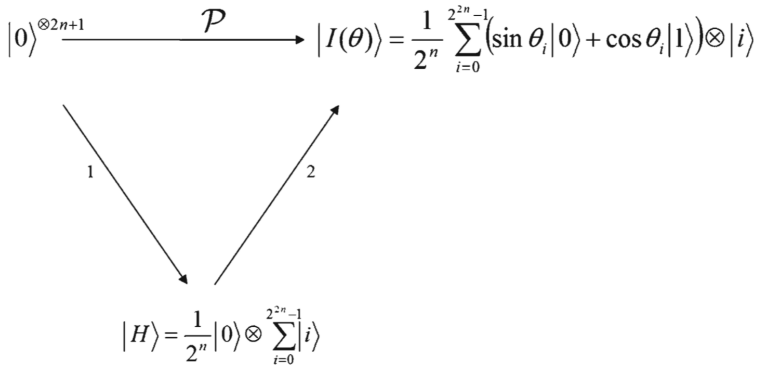
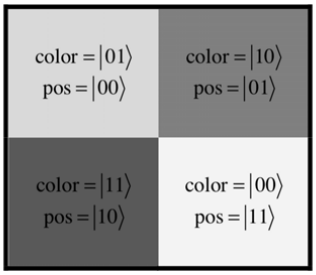


Figure 9: The process of how to prepare an image to be represented in the FRQI model. Here P represents the transformation in one step on top, while the bottom two represent the two steps involved, which uses the Hadamard gates in the first step and the controlled rotations in the second step [5].

take, which are outlined in Figure 9. Before any steps are taken, there are $2n + 1$ qubits initialized to $|0\rangle$, which is written as $|0\rangle^{\otimes 2n+1}$. For the first step, a Hadamard transform is performed on the qubits, which is done using $2n$ Hadamard gates since it is applied to only the position qubits. After this step is performed, the final step takes this new state and applies a controlled rotation transform is used to get the FRQI state. This step uses 2^{2n} controlled rotations, which is done via NOT gates and controlled rotation gates.

2.4.2 Caraiman-Manta

Another image representation used is one proposed by Caraiman and Manta, as explained in [2]. This type of image representation take a grayscale image with size $2^n \times 2^n$ and with 2^m grayscale values. This image



$$|Q\rangle = \frac{1}{\sqrt{2^2}} \sum_{i=0}^{2^2-1} \sum_{j=0}^{2^2-1} \alpha_{ij} |j\rangle |i\rangle =$$

$$= \frac{1}{\sqrt{2^2}} (|01\rangle|00\rangle + |10\rangle|01\rangle + |11\rangle|10\rangle + |00\rangle|11\rangle)$$

$$\alpha_{00} = 0,\ \alpha_{01} = 1,\ \alpha_{02} = 0,\ \alpha_{03} = 0$$
$$\alpha_{10} = 0,\ \alpha_{11} = 0,\ \alpha_{12} = 1,\ \alpha_{13} = 0$$
$$\alpha_{20} = 0,\ \alpha_{21} = 0,\ \alpha_{22} = 0,\ \alpha_{23} = 1$$
$$\alpha_{30} = 1,\ \alpha_{31} = 0,\ \alpha_{32} = 0,\ \alpha_{33} = 0$$

Figure 10: An example image and its corresponding state as described by Caraiman and Manta. In this case the α matrix has each of the positions filled in with the corresponding color, after converting from binary. For example, at position 1 in the image, the color is 2, so $\alpha_{12} = 1$, and the rest of the values in that row are equal to 0[2]

.

representation is stored in the state Q, where

$$Q = |C\rangle_m \otimes |P\rangle_{2n} = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \alpha_{ij} |j\rangle |i\rangle .$$

In this case, we have $|P\rangle$ represented in the form $|y\rangle |x\rangle$ and encode the row and column position of a pixel, respectively [6]. The colors are encoded in $|C\rangle$, using m qubits to represent all the different colors. In general, the color of a pixel at position $|i\rangle$ is

$$|C_i\rangle = \sum_{j=0}^{2^m-1} \alpha_{ij} |j\rangle .$$

The α_{ij} coefficient represents the color of a pixel at position i using a superposition of all possible colors. This means that $\Sigma_{j=0}^{2^m-1} |\alpha_{ij}|^2 = 1$, meaning that each position is normalized. Usually the α_{ij} coefficients are stored in a matrix, where the number of rows represent the total number of pixels in an image and the columns represent the different possible grayscale values. Therefore, the matrix has a size of $N \times L$, where $N = 2^n \times 2^n$ and $L = 2^m$. In general, the pixel at position i has α_{ij} equal to 1 if the color of the pixel is j, otherwise the value is 0. Figure 10 shows how an image is represented in this form.

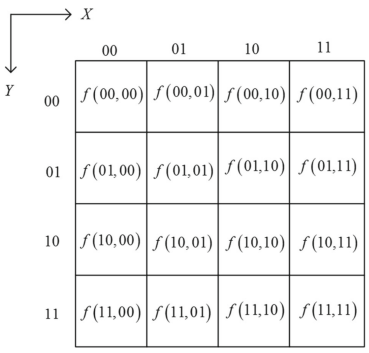


Figure 11: An example image and its corresponding state in the NEQR model. The positions are defined on the x and y axis, starting from 0 and going up to 3. The color at said position would be $f(y, x)$. So at position (y, x), the color is $f(y, x)$[11].

2.4.3 NEQR

One of the modern approaches for a quantum image representation is the NEQR (Novel Enhanced Quantum Representation of digital images) model [11]. The image used for this model has to be of size $2^n \times 2^n$ with a gray range of 2^m for it to be stored. This method uses two qubit sequences that are entangled together so that the entire image is stored in these two sequences. One qubit sequence stores the grayscale values and the other one stores the position information. The representation is stored in $|I\rangle$, where

$$|I\rangle = \frac{1}{2^n} \sum_{Y=0}^{2^n-1} \sum_{X=0}^{2^n-1} |f(Y, X)\rangle \, |YX\rangle \, .$$

For the equation above, $f(Y, X)$ denotes the grayscale value of the pixel at the point (Y, X). The value is then stored as the basis state $|f(Y, X)\rangle$ of a qubit sequence with the position being stored in the basis state $|YX\rangle$, making this image representation use $2n + q$ qubits to store it. Using the tensor product, the image is then stored into the two different sequences. Figure 11 shows an example of a 4×4 image in the NEQR model. What makes this representation different from the one in Section 2.4.2 is how the grayscale values are stored. For the NEQR model, there is a unique mapping that maps each pixel position to a unique grayscale value, while the previous image representation created a matrix to store the grayscale values at that position.

3 Methods and Design

After reviewing all of the background literature, I decided to study different filtering methods that others have proposed and see which ones could be studied and implemented in a quantum computer. All of these

$$\mathcal{W} = \begin{bmatrix} h_{Y-k,X-k} & h_{Y-k,X-k+1} & \cdots & & h_{Y-k,X+k} \\ h_{Y-k+1,X-k} & h_{Y-k+1,X-k+1} & \cdots & & h_{Y-k+1,X+k} \\ & & & \ddots & \\ h_{Y,X-k} & h_{Y,X-k+1} & \cdots & h_{Y,X} \cdots & h_{Y,X+k} \\ & & & \ddots & \\ h_{Y+k,X-k} & h_{Y+k,X-k+1} & \cdots & & h_{Y+k,X+k} \end{bmatrix}$$

Figure 12: A general filter used for correlation of an image. For a given position (Y, X) in an image, the filter is centered at position $h_{Y,X}$. Each of the neighborhood pixels are multiplied by the given coefficients from the filter and then added together to get the new value at (Y, X) [11].

filtering methods are done in either the frequency domain, which deals with the frequencies found in the image, or the spatial domain, which deals with the actual image represented in a matrix, with each matrix value representing the color value at that position.

3.1 Image Filtering in Spatial Domain

One example of quantum filtering in the spatial domain was proposed by [11]. This method involved using the NEQR model of an image to represent it in the quantum computer. To represent this image, the authors assume that the image is of size $2^n \times 2^n$ with 2^q grayscale values. In order to filter the image, the authors decide to do it by correlation. For a given image f and a filter w, the filter moves point to point on the image. For each (x, y) position in the image f, the new value at that position is the sum of products of the filter coefficients and all of the neighborhood pixels, which are all the pixels that w covers. Figure 12 shows a general mask of size $(2k + 1) \times (2k + 1)$ for some integer k that could be used as a process of correlation.

In order to accomplish correlation filtering in a quantum system, the authors decided to use a position shifting transformation to have the ability to shift the image. This makes it easier to access the neighbors of a pixel in a given image. The authors also decided to implement quantum image addition to their filtering method for two images represented in the NEQR model. The circuit for their quantum full adder is shown in Figure 13.

With all of this information, the authors presented the following steps to filter an image I_{YX} that is represented in the NEQR model and a filter $\mathcal{W}$ of size $(2k + 1) \times (2k + 1)$.

1. Prepare h copies of I_{YX}. These can be written as the set $\{|I_{YX}\rangle_1, |I_{YX}\rangle_2, \ldots, |I_{YX}\rangle_h\}$. Note that h is obtained counting up the number of entries in $\mathcal{W}$, i.e., $h = (2k + 1) \times (2k + 1) = 4k^2 + 4k + 1$

2. Obtain the neighborhood image set. By performing position shifting operations on each image ac-

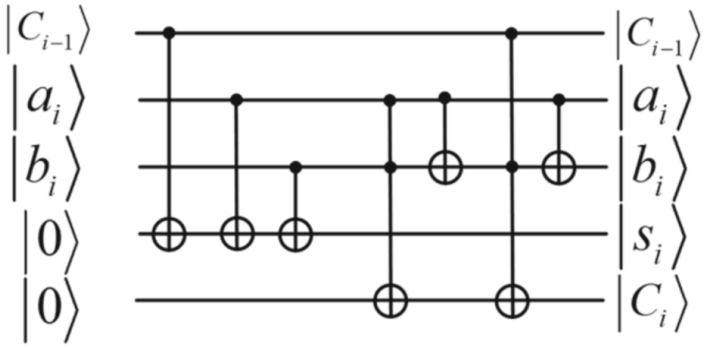


Figure 13: A quantum full adder to add two images together. C_{i-1} represents the carry-in, while a_i and b_i are the two qubits to add. The result of the addition is stored in s_i, with the carry-out represented by C_i. Note that this uses a combination of CNOT gates and CCNOT gates, also known as a Toffoli gate. These act just like a CNOT gate, however both control qubits have to be set to 1 to negate the target qubit [11].

cordingly, we can end up with the set

$$\begin{aligned}
&\{|\mathcal{W}_{Y-k,X-k}|\,|I_{Y-k,X-k}\rangle\,,\\
&|\mathcal{W}_{Y-k,X-k+1}|\,|I_{Y-k,X-k+1}\rangle\,,\ldots\\
&|\mathcal{W}_{Y,X}|\,|I_{Y,X}\rangle\,,\ldots,\\
&|\mathcal{W}_{Y+k,X+k-1}|\,|I_{Y+k,X+k-1}\rangle\,,\\
&|\mathcal{W}_{Y+k,X+k}|\,|I_{Y+k,X+k}\rangle\}
\end{aligned}$$

3. Add up all of the h copies of the image set to obtain the filtered image, which is represented as $|\psi\rangle$.

The corresponding quantum circuit for this filtering method is shown in Figure 14. Looking at the complexity of this algorithm, it would take $O(n^2)$ to transform n-sized NEQR images for the position shifting operations, since a total of n shifts have to be done on n images in the worst case. The number of position shifting operations is dependent on the size of the filter mask, which in this case is $(2k+1)\times(2k+1)$, meaning that the complexity is $O(k^2n^2)$. To add all of the images together, it would take $O(q^2)$ [11]. After that, it would have to be done h times, which is equal to k^2, so the total time complexity for adding the images together would be $O(k^2q^2)$. Combining all of these factors together, we have a time complexity of $O(k^2n^2+k^2q^2)=O(k^2(n^2+q^2))$ to run the correlation on the image.

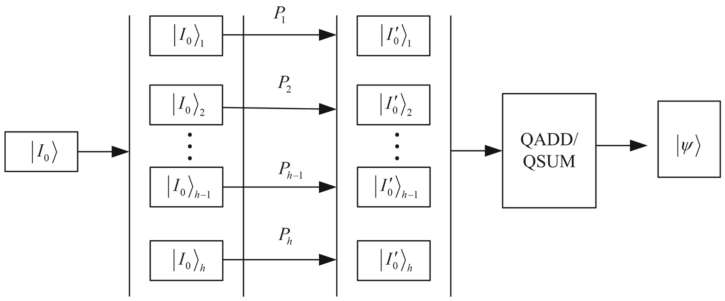


Figure 14: The quantum circuit to filter an image in the spatial domain in a quantum computer. The image is copied h times, a specific shifting operation is done on each image copy for a specific result, and then all of the image copies are added together to get the filtered image, which is stored as $|\psi\rangle$ [11].

3.2 Image Filtering in the Frequency Domain

When dealing with filtering in the frequency domain, one of the most important steps is to look at how the frequencies of the image can be obtained. In the classical system, this usually done by computing the Fast Fourier transform on an image. In this case, there is a quantum analog for the Fourier transform, called the quantum Fourier transform. In general, the quantum Fourier transform is applied to a superposition of states, $|x\rangle$, where there are N states, and is given by the formula

$$QFT_N |x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i x k}{N}} |k\rangle .$$

Likewise, there is also the inverse quantum Fourier transform, since the quantum Fourier transform has to be unitary. This transform applies to a basis state $|x\rangle$, which is used as a scalar. This is represented as

$$QFT_N^{-1} |x\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{-2\pi i x k}{N}} |k\rangle .$$

3.2.1 Cairaman-Manta Frequency Filtering Method

I looked at two different types of filtering in the frequency domain. The first one is proposed by [2] and uses their own image representation as discussed in Section 2.4.2. This image representation takes a grayscale image of size $2^n \times 2^n$ with 2^m grayscale values. Their method to filter the image utilizes the quantum Fourier transform and a quantum oracle, a circuit that 'knows' solutions to a given problem and acts like a black box. The quantum circuit for this filtering method is shown in Figure 15.

Before applying the filtering method, the image is stored in $|I_0\rangle$ using their own image representation.

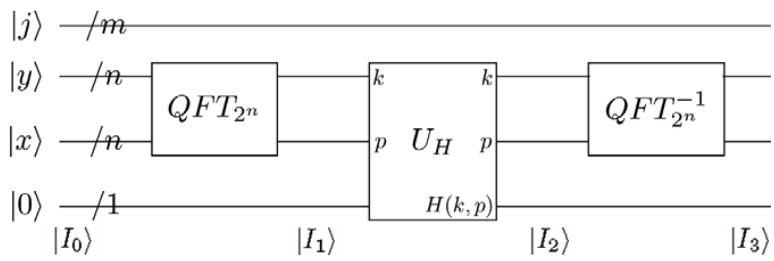


Figure 15: The circuit that was implemented to recreate the filtering process in [2] using a Fourier transform and an oracle. It first applies a quantum Fourier transform, then separates the wanted and the unwanted frequencies using the oracle, then does the inverse quantum Fourier transform to get the filtering result. The slashes on the lines represent the number of qubits used, so $|y\rangle$ and $|x\rangle$ use n qubits each and $|j\rangle$ uses m qubits.

For this method, we have

$$|I_0\rangle = \frac{1}{2^n} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \alpha_{yxj} |j\rangle |y\rangle |x\rangle |0\rangle .$$

For clarity later on, i from the image representation has been changed to yx but still represents position i in the matrix for the alpha values. There is also an additional qubit added on to be used later on by the oracle. Going through the circuit in Figure 15, there are three steps to filtering the image.

1. Apply the quantum Fourier transform to $|I_0\rangle$, specifically on the qubits that stores the x and y position of the image. This new state becomes $|I_1\rangle$, and is produced by

$$|I_1\rangle = \frac{1}{2^n} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \alpha_{yxj} |j\rangle \left(QFT_{2^n} |y\rangle\right) \left(QFT_{2^n} |x\rangle\right) |0\rangle \tag{1}$$

$$= \frac{1}{2^n} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \alpha_{yxj} |j\rangle \left(\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i y k}{2^n}} |k\rangle \right) \left(\frac{1}{\sqrt{2^n}} \sum_{p=0}^{2^n-1} e^{\frac{2\pi i x p}{2^n}} |p\rangle \right) |0\rangle \tag{2}$$

$$= \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k=0}^{2^{2n}-1} \sum_{p=0}^{2^{2n}-1} \alpha_{yxj} e^{\frac{2\pi i y k}{2^n}} e^{\frac{2\pi i x p}{2^n}} |j\rangle |k\rangle |p\rangle |0\rangle \tag{3}$$

$$= \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k=0}^{2^{2n}-1} \sum_{p=0}^{2^{2n}-1} \alpha_{yxj} e^{\frac{2\pi i (yk+xp)}{2^n}} |j\rangle |k\rangle |p\rangle |0\rangle . \tag{4}$$

   In (1), the quantum Fourier transform is applied to $|y\rangle$ and $|x\rangle$, followed by the definition of it in (2). The equation is then simplified, resulting in (4).

2. Use the quantum oracle, defined as U_H, as a way to separate the good frequencies from the bad frequencies. This oracle is defined as $|k\rangle |p\rangle |z\rangle \xrightarrow{U_H} |k\rangle |p\rangle |z \bigoplus H(k,p)\rangle$. For this we have $|0\rangle$ used for

Filter	Freqeuncy Set
Low-pass	$\begin{cases} S_{good} = \{k,p \vert D(k,p) \leq D_0\} \\ S_{bad} = \{k,p \vert D(k,p) > D_0\} \end{cases}$
High-pass	$\begin{cases} S_{good} = \{k,p \vert D(k,p) \geq D_0\} \\ S_{bad} = \{k,p \vert D(k,p) < D_0\} \end{cases}$

Table 1: A table showing the different sets that are considered for S_{good} and S_{bad}, for both a low-pass filter and a high-pass filter. D_0 is the cutoff frequency, and $D(k,p)$ is the Euclidean distance from the origin of the image, in the top left of the corner, to the position pixel (k,p) [2].

$|z\rangle$ and $H(k,p)$ defined as the filter function, with

$$H(k,p) = \begin{cases} 1 & k,p \in S_{good} \\ 0 & k,p \in S_{bad} \end{cases}$$

There are different sets of coordinates that belong to S_{good} and S_{bad}, depending on what filter is used. Table 1 has the sets used for a high-pass filter and a low pass filter. In general, the new state is defined as $|I_2\rangle$ and is equal to

$$|I_2\rangle = \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k=0}^{2^{2n}-1} \sum_{p=0}^{2^{2n}-1} \alpha_{yxj} e^{\frac{2\pi i(yk+xp)}{2^n}} |j\rangle U_H \left(|k\rangle |p\rangle |0\rangle\right) \tag{5}$$

$$= \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle |k\rangle |p\rangle |1\rangle + \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{bad}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle |k\rangle |p\rangle |0\rangle \tag{6}$$

To simplify the equation, we let $\alpha'_{yxj} = \alpha_{yxj} \cdot e^{\frac{2\pi i(yk+xp)}{2^n}}$. This makes the 'good' frequencies set to the state $|1\rangle$ and the 'bad' frequencies to the state $|0\rangle$, which is done in (5), and the result of the oracle is shown in (6).

3. After finding the different frequency sets, we take the inverse quantum Fourier transform and end up with two different images, one image that has the desired frequencies from S_{good}, and the images

suppressed by the filter, S_{bad}. This is defined as $|I_3\rangle$, where

$$|I_3\rangle = \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle \left(QFT_{2^n}^{-1} |k\rangle\right) \left(QFT_{2^n}^{-1} |p\rangle\right) |1\rangle +$$

$$\frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{bad}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle \left(QFT_{2^n}^{-1} |k\rangle\right) \left(QFT_{2^n}^{-1} |p\rangle\right) |0\rangle \tag{7}$$

$$= \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle \left(\frac{1}{\sqrt{2^n}} \sum_{r=0}^{2^n-1} e^{\frac{-2\pi ikr}{2^n}} |r\rangle\right) \left(\frac{1}{\sqrt{2^n}} \sum_{t=0}^{2^n-1} e^{\frac{-2\pi ipt}{2^n}} |t\rangle\right) |1\rangle +$$

$$\frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{bad}}^{2^{2n}-1} \alpha'_{yxj} |j\rangle \left(\frac{1}{\sqrt{2^n}} \sum_{r=0}^{2^n-1} e^{\frac{-2\pi ikr}{2^n}} |r\rangle\right) \left(\frac{1}{\sqrt{2^n}} \sum_{t=0}^{2^n-1} e^{\frac{-2\pi ipt}{2^n}} |t\rangle\right) |0\rangle \tag{8}$$

$$= \frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi ikr}{2^n}} e^{\frac{-2\pi ipt}{2^n}} |j\rangle |r\rangle |t\rangle |1\rangle +$$

$$\frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi ikr}{2^n}} e^{\frac{-2\pi ipt}{2^n}} |j\rangle |r\rangle |t\rangle |0\rangle \tag{9}$$

$$= \frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} |j\rangle |r\rangle |t\rangle |1\rangle +$$

$$\frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} |j\rangle |r\rangle |t\rangle |0\rangle \tag{10}$$

$$= \frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{j=0}^{2^m-1} \alpha^b_{rtj} |j\rangle |r\rangle |t\rangle |1\rangle + \frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{j=0}^{2^m-1} \alpha^g_{rtj} |j\rangle |r\rangle |t\rangle |0\rangle \tag{11}$$

Similar to (1), the inverse quantum Fourier transform is applied to qubits that store the position of the image, which is $|k\rangle$ and $|p\rangle$ in (7) to both parts of the equation. The definition of the inverse quantum Fourier transform is applied in (8), which is then simplified all the way down to (10). The equations are simplified further in (11), where we have

$$\alpha^g_{rtj} = \sum_{k,p \in S_{good}}^{2^{2n}-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} \tag{12}$$

$$\alpha^b_{rtj} = \sum_{k,p \in S_{bad}}^{2^{2n}-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} \tag{13}$$

One thing to note here is that adding both of these different frequencies from (11) together results in the original image.

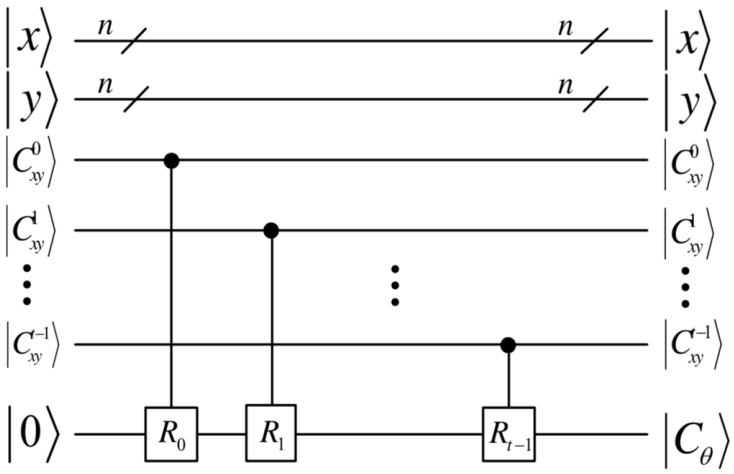


Figure 16: The circuit to transform a NEQR image to a FRQI image. Note that the position qubits are not touched but only the color qubits so that the t different color qubits can be stored in one qubit [4].

3.2.2 Liu, Zhang, Wang, and Lu Frequency Filtering Method

Another strategy to filter in the frequency domain is proposed by [4] that is similar to the method described above. A grayscale image of size $2^n \times 2^n$ with 2^q grayscale values is represented in the NEQR model. The authors decided to transform the image from the NEQR model to the FRQI model to make it easy to perform a quantum Fourier transform on an image. The circuit that takes a NEQR model to a FRQI model is shown in Figure 16. It utilizes a rotation matrix R_i to rotate the different colors and store them in a vector of values, exactly what the FRQI model does.

Once we have the image stored in the FRQI model as mentioned in Section 2.4.1, the filtering method is as follows.

1. Apply the quantum Fourier transform to the image, resulting in the image spectrum $|F\rangle$. This is produced by

$$|F\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (\cos(\theta_i)\,|0\rangle + \sin(\theta_i)\,|1\rangle) \otimes QFT(|i\rangle).$$

2. Split the frequencies by using a threshold value T. By looking at the different θ values for each position, it can either be kept or discard. This is represented in the state $|\xi\rangle$, where

$$|\xi\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} \sum_{\theta \le T} (\cos(\theta_i)\,|0\rangle + \sin(\theta_i)\,|1\rangle) \otimes |i\rangle + \sum_{i=0}^{2^{2n}-1} \sum_{\theta > T} |0\rangle \otimes |i\rangle\,.$$

It is important to note that this operation is not unitary because it is not reversible. This is due to the

threshold throwing away some values, which makes it impossible to retrieve later on.

3. To finish, we take the inverse quantum Fourier transform, giving us the desired filtered image. This is shown as

$$\frac{1}{2^n}\sum_{i=0}^{2^{2n}-1}\cos(\theta_i')\,|0\rangle+\sin(\theta_i')\,|1\rangle\otimes|i\rangle$$

where θ_i' represents the new color values from the new vector that has the filtered values below the threshold.

The resulting filtered image is saved in the FRQI model, which can be used for further operations as desired.

4 Experiment

This section talks about all of the work done to implement a filtering algorithm on a quantum computer. The first simulation is done by directly doing the calculations of the filtering. The second simulation uses a simulated quantum computer, implementing a circuit to filter an image. Both of these simulations use the filtering algorithm proposed by Caraiman and Manta in [2].

4.1 Direct Calculation Simulation

Before implementing any filtering methods, I had to decide how I was going to implement it, whether it would be implementing a circuit on a simulated quantum computer or do the calculations using matrix multiplication. I chose to do the latter first since the calculations are easier to implement and it would not require time to learn and compile new software. The primary goal here is to explore and do preliminary research on the filtering method. I decided to use Jupyter Notebooks so that I can observe the results of any code easily. Although this implementation is not complex to use, it does not actually simulate what the filtering method does.

The filtering method I attempted to do is the one proposed by [2]. I decided to do this one since the authors had proposed a filtering method that was not hard to create the image representation, as well as the actual filtering algorithm. The test image that is used is given in Figure 17. This image is 32×32 and only has 2 grayscale values in it, which is able to be represented in the quantum computer. The first thing I was able to implement was the matrix that was shown in Figure 10. This is how the authors were able to represent the quantum images in their simulation and I was able to do this as well. This matrix, which will be referred to as the alpha matrix, has $32 \times 32 = 1024$ rows and 2 columns. Each column corresponded to a

Figure 17: The test image I used for the filtering method. This is the same one that [2] used in theirs, which makes it easy to compare results

color from the image, and if the color at a position i was j, then $\alpha_{ij} = 1$, and the other column was set equal to zero in that row.

From there my goal is to use the equations in each of the three steps from Section 3.2.1 to apply the filtering method to the image. To represent the different states stored in the quantum system, I saved their state vector in an array. In other words,

$$\frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^{m}-1} \sum_{k=0}^{2^{2n}-1} \sum_{p=0}^{2^{2n}-1} \alpha_{yxj} e^{\frac{2\pi i(yk+xp)}{2^n}} |j\rangle |k\rangle |p\rangle |0\rangle$$

$$\implies |I_1\rangle [j,k,p,0] = \frac{1}{2^{2n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \alpha_{yxj} e^{\frac{2\pi i(yk+xp)}{2^n}}.$$

The top equation is equation (4). To save the state vectors for all the possible qubit states, $|I_1\rangle [j,k,p,0]$ stores the probability of each state in an array. This was done to save all of the values and access them later on for further calculations. After this step, the next step involved separating the values based on the different frequency sets. This is done by separating the frequencies by its Euclidean distance of the pixel from the origin, located in the top left of the frequency domain, where the pixel location is $(0,0)$. To split the pixels up, I took the x and y components of the image and found the distance by computing $\sqrt{x^2+y^2}$. If that distance is less than the threshold, known as the cutoff frequency, it would be a wanted frequency, otherwise it would be discarded. The cutoff frequency used is $D_0 = 6.4$, which is what the authors chose for the test image. All that is left is applying the final step to the image, which is the inverse quantum Fourier transform. Similar to above, I represented the different states as an array, which makes it

$$\frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} |j\rangle |r\rangle |t\rangle |1\rangle +$$

$$\frac{1}{2^{3n}} \sum_{r=0}^{2^n-1} \sum_{t=0}^{2^n-1} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{j=0}^{2^m-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} |j\rangle |r\rangle |t\rangle |0\rangle$$

$$\implies |I_{good}\rangle [j,r,t,1] = \frac{1}{2^{3n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{k,p \in S_{good}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}} +$$

$$|I_{bad}\rangle [j,r,t,0] = \frac{1}{2^{3n}} \sum_{y=0}^{2^{2n}-1} \sum_{x=0}^{2^{2n}-1} \sum_{k,p \in S_{bad}}^{2^{2n}-1} \alpha'_{yxj} e^{\frac{-2\pi i(kr+pt)}{2^n}}.$$

The equation here is represented as equation (9) and rewritten as an array to represent the state vector. This end result should have the good frequencies in $|I_{good}\rangle$, which is the resulting filtered image. A problem that was encountered when trying to implement the filtering method this way was keeping every step of the filtering method normalized. After each step is performed, the probability of it should equal 1, however that was not happening in this case. While there were difficulties in making this simulation work, it provided insight and gave a better understanding of what is going on in this filtering algorithm.

4.2 Quantum Computer Simulation

After dealing with the difficulties of directly calculating the state vector, I decided to that it would be better to use a quantum computer simulator, which would help avoid the problem of each step of the filtering algorithm be normalized. I used the quantum computer simulator Qiskit, which is developed by IBM. My goal here was to implement the circuit proposed by [2] directly into the simulator, which should be easier to do than computing the calculations directly. I was successfully able to implement the filtering algorithm on a quantum circuit, which is shown in Figure 18. I will be going over each gate that is shown in the circuit and explain how they are used.

While my circuit is similar to what [2] proposed, there are a few differences between the two. The biggest one being the use of extra qubits. These qubits had to be used in order to implement the oracle, which will be describe the use of these qubits late on. Although they were added to circuit, they did not interfere with the results later on.

Image Representation Before running the actual filtering algorithm, the first step to do is to put the image in the right image representation for the quantum computer to use. I decided to use the same input image as them, as shown in Figure 17, where the top half of the image has one color and the bottom half of the

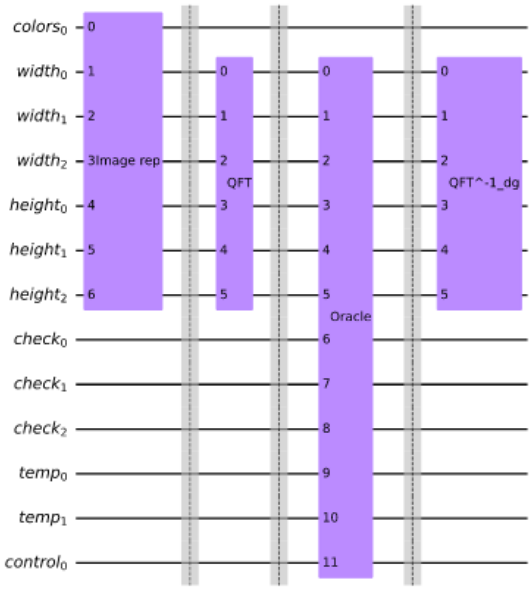


Figure 18: The circuit by [2] done in Qiskit. This is the circuit for the image in Figure 17, but for an 8×8 image.

image has a different color. As an example, we will use the test image and use the image representation on a smaller size, 2×2. This would use a total of 3 qubits, having one represent the height, one representing the width, and one representing the two different colors. More specifically for the colors, we will let the state $|0\rangle$ represent the color on top and let the state $|1\rangle$ represent the color on the bottom. Thus the image representation is

$$|Q\rangle = \frac{1}{\sqrt{2^2}} \sum_{i=0}^{2^2-1} \sum_{j=0}^{2^1-1} = \alpha_{ij} |i\rangle |j\rangle = \frac{1}{2} \begin{pmatrix} \alpha_{00} |00\rangle |0\rangle \\ \alpha_{01} |00\rangle |1\rangle \\ \alpha_{10} |01\rangle |0\rangle \\ \alpha_{11} |01\rangle |1\rangle \\ \alpha_{20} |10\rangle |0\rangle \\ \alpha_{21} |10\rangle |1\rangle \\ \alpha_{30} |11\rangle |0\rangle \\ \alpha_{31} |11\rangle |1\rangle \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

This is the desired matrix that is needed to represent the image. Since all of the qubits in Qiskit start in the state $|0\rangle$, I decided to manipulate the color and the position qubits to get the desired matrix. In the current

state, the qubits are represented as

$$|0\rangle_{height} \otimes |0\rangle_{width} \otimes |0\rangle_{color} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{height} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{width} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{color} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

There are two steps from this state to get the desired image representation. The first step applies a Hadamard transformation on the position qubits, which in this case is applying it to the width and the height qubits. This gives us

$$\begin{aligned} H(|0\rangle_{height}) \otimes H(|0\rangle_{width}) \otimes |0\rangle_{color} &= H \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{height} \otimes H \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{width} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{color} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}_{height} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}_{width} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}_{color} \\ &= \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \end{aligned}$$

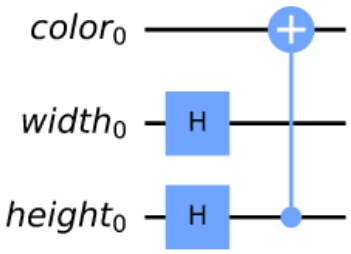


Figure 19: The circuit for a 2×2 size of the test image.

Next, we want to use a CNOT gate. The control is the height qubit and the target will be the color qubit. We will let $CNOT$ represent the operation done on those qubits. Thus the image representation becomes

$$CNOT \left[\frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \left[\frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right] = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

This gives us the desired image representation state. For this example, the matrix above represents the $CNOT$ operation with the height qubit as the control and color qubit as the target. The circuit for this image representation is in Figure 19. In general, to get the image representation of the test image in Figure 17 that has the same color layout and is of size $2^n \times 2^n$, we do the following steps:

1. Apply a Hadamard gate on the position qubits, which will be a total of $2n$ qubits.
2. Apply a CNOT gate where the most significant qubit for the height is the control and the color qubit is the target.

Quantum Fourier Transform After completing the image representation, the next step is to apply the quantum Fourier transform to the position qubits. This is done to obtain information about the frequencies on the position qubits. Using the previous example, it would be applied to 2 qubits, the width qubit and the height qubit, in the 2×2 image of Figure 17. As a circuit, the quantum Fourier transform can be created using Hadamard gates, controlled rotation gates, and SWAP gates, which swaps the states of two qubits.

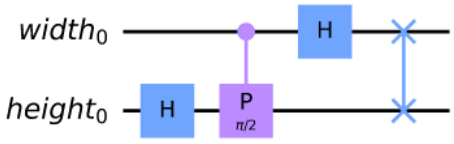


Figure 20: The circuit for the quantum Fourier transform.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

Figure 21: These are the points that are desired for the low-pass filter. The points in green are the "good" points for the filter, while the points in red represent the "bad" frequencies

The textbook that IBM has on using Qiskit has a section on the quantum Fourier transform and some code to create it, which I used in my circuit. This circuit is shown in Figure 20.

Oracle Following the quantum Fourier transform is the oracle, which separates the desired frequencies from the undesired frequencies. In this case, this is done by calculating the Euclidean distance from the origin of the image, which in this case is the top left corner. The cutoff frequency, D_0, is set to 6.4, which is what [2] uses as well. Creating this oracle ended up being the toughest part of the circuit and required quite a bit of time to determine the best way to separate the frequencies. I eventually decided to implement the oracle by seeing which points are within the desired frequency set and which ones are not. This means that the Euclidean distance is previously calculated and the points are determined ahead of time.

For this oracle, a low-pass filter was used, meaning that the points that have their Euclidean distance from the origin less than or equal to the cutoff frequency, which is 6.4, is a desired frequency. Figure 21 lists the points in an $8x8$ grid to show which points are desired in an image. If the size of the image increases, these point are kept as the only desired frequencies and the other points would be undesirable.

To see if the point in the image is a desired frequency, I used a built in circuit in Qiskit called IntegerComparator. This circuit takes a number i that is n qubits long and checks if is greater than or equal to a given integer L. This is also also done to check if L is greater than i. This is expressed in Dirac notation by

$$|i\rangle_n |0\rangle \xrightarrow{IntegerComparator} |i\rangle_n |i \geq L\rangle$$
$$|i\rangle_n |0\rangle \xrightarrow{IntegerComparator} |i\rangle_n |i < L\rangle .$$

A target qubit is given and will flip from $|0\rangle$ to $|1\rangle$, or vice versa, if the IntegerComparator check is true, whether $i \geq L$ or $i < L$. Using this and the list of points that are desired from Figure 21, the possible points with width w and height h are

- if $w < 5$ and $h < 5$
- if $w == 5$ ($w \geq 5$ and $w < 6$) and $h < 4$
- if $w == 6$ ($w \geq 6$ and $w < 7$) and $h < 3$
- if $w < 4$ and $h == 5$ ($h \geq 5$ and $h < 6$)
- if $w < 3$ and $h == 6$ ($h \geq 6$ and $h < 7$)

Each one of these checks for possible points can be implemented on the circuit using a IntegerComparator gate, with the result going to a check qubit. After the checks for a set of IntegerComparator gates, a multi-controlled NOT gate is used on all of the check qubits as the control, with the target being the control qubit. After one set of checks, the check qubits reset to $|0\rangle$ for the next set of gates to check for possible points. The temp qubits are used to implement the IntegerComparator gate since the operation is based on two's complement and subtracts to get the answer. An example of one of the integer comparator points is shown in Figure 22.

Inverse Quantum Fourier Transform Once the oracle separates the desired frequencies, the inverse quantum Fourier transform reverses the process so that information about the frequencies can be read. This is done by performing the same operations that the quantum Fourier transform does but in reverse order, which is done by using some helper methods that Qiskit provides. Like the quantum Fourier transform, the operation is done on the position qubits. The circuit for the step is shown in Figure 23.

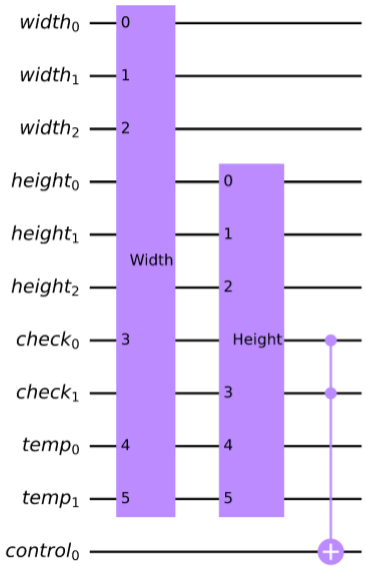


Figure 22: The circuit for the first point check, if $w < 5$ and $h < 5$. A control gate is added to check if both conditions are met, which flips the control bit if both are true.

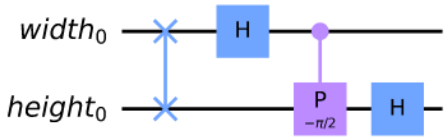


Figure 23: The circuit for the inverse quantum Fourier transform on two qubits.

Measurement All of these steps complete the circuit and filter the image. After running the circuit we get the output as a set of probability amplitudes. To find out what frequencies are desired, equation (11) from Section 2.4.2 is used since this is the final state that is given once the circuit is done. In order to get the gray level of a pixel at the position rt, it would be

$$\frac{1}{2^n} \sum_{j=0}^{2^m-1} \alpha_{rtj}^g |j\rangle$$

for the filtered image and

$$\frac{1}{2^n} \sum_{j=0}^{2^m-1} \alpha_{rtj}^b |j\rangle$$

for the suppressed frequencies, where α_{rtj}^g and α_{rtj}^b are defined in (12) and (13) respectively. To get the correct gray level from the test image, the basis states are changed to the color value that they represent. In this case, we had $|0\rangle$ representing the gray value 170 and $|1\rangle$ representing the gray value 100. As an

example, to find the gray level of the desired frequency at position rt in the test image, the value is

$$\text{gray level}_{rt} = \frac{1}{2^n} \sum_{j=0}^{2^m-1} \alpha_{rtj}^g |j\rangle = x|0\rangle + y|1\rangle = x(170) + y(100),$$

where x is the probability of color $|0\rangle$ at position rt and y is the probability of color $|1\rangle$ at position rt. Note that $m = 1$ here since there are two gray levels in the image. This is done for each pixel to get the new gray level for the desired frequencies and the undesired ones. For some pixels, these values may be negative. To properly display these frequencies, a rescaling of all of the values are done, where the smallest value from the gray levels from a set of frequencies represents the minimum grayscale value, 0, and the largest value represents the maximum grayscale value, 255. This rescaling now makes every pixel have a corrected gray level.

5 Results

Although a circuit was implemented to apply a low-pass filter to the image, it was not successful in getting the correct frequencies. There was a lot of difficulty implementing an oracle that would correctly filter an image. There were many different ideas that I wanted to do to create an oracle, however there were problems implementing the circuit or that too many temporary qubits were required, which would slow down the computation. To check that the low-pass filter worked, the circuit is applied to the image in Figure 17 of size 8×8, with one color in the top half and a different color on the bottom.

While the oracle did not change for the experiment, one aspect of it that did change is how the check qubits were reset after passing through a set of IntegerComparator gates. At first, they were reset by using a reset gate, which makes each check qubit return to the state $|0\rangle$. Figure 24 shows what the low-pass filter of the test image is using this method. Looking at this image using a measurement, we see the distinctions of the low frequencies. The top half of the image is lighter than the bottom half, which is true when looking at the test image.

This method is also applied to the test image of size 32×32. This is also the size that Caraiman-Manta used as well for the test image. Figure 25 has both the image result from the circuit implemented and the result that the authors got. While both images look different, it does show the some similarities with the low frequencies and creates a blur effect that is seen throughout the image. Both images also have the lighter color on top and the darker color on the bottom. The middle section for the two images also has a blend change from light to dark, which is also consistent for both of them. Altogether, the exact results are not the same however the general characteristics of low frequencies are the same. However, this method of

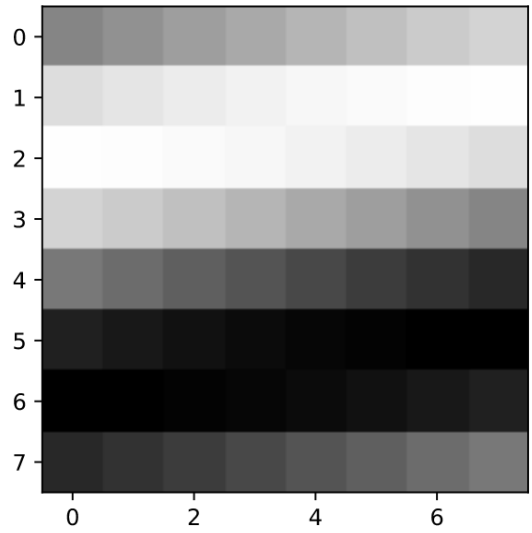


Figure 24: The low frequencies of the test image. This was obtained by passing the image through the circuit in Figure 18.

resetting the control qubits is not useful since the result from the circuit only gives the low frequencies of the image instead of both the low and high frequencies, which is what is suppose to happen. This is due to the fact that the reset gate does a measurement on the qubit, which confounds the final state of the circuit.

As a solution to resetting the check qubits, one way to reset the qubits without making a measurement would be to use the set of IntegerComparator circuits after the initial check is made. This is done due to the fact that these operations are unitary, meaning that the check qubits can be initialized back to its initial state of $|0\rangle$ without interfering with the output by inverting the operation. This also doesn't interfere with the control qubit since the first set of IntegerComparator gates already did the operation, meaning that it will not change the output of the control qubit. This was done and implemented on an 8×8 version of the test image, with the results being shown in Figure 26. Even though the results from this circuit did have both the high frequencies and the low frequencies, these results do not resemble the what low frequencies and high frequencies should be in the test image.

There are a few possibilities to explain this discrepancy between the results I get and what Caraiman and Manta got. One explanation can be due to the representation of the oracle. The oracle presented in [2] does not have any clear directions as to how to use it. While they only care about what the input and the output of the oracle was, there is no clear definition and how one can implement it. This comes from the definition of $D(k, p)$, which is not something that is known to me nor something I found online. Even using the same cutoff frequency as them for my filter did not help either. Despite not getting the same results, there are similar comparisons between the two which makes an argument that the low frequencies were

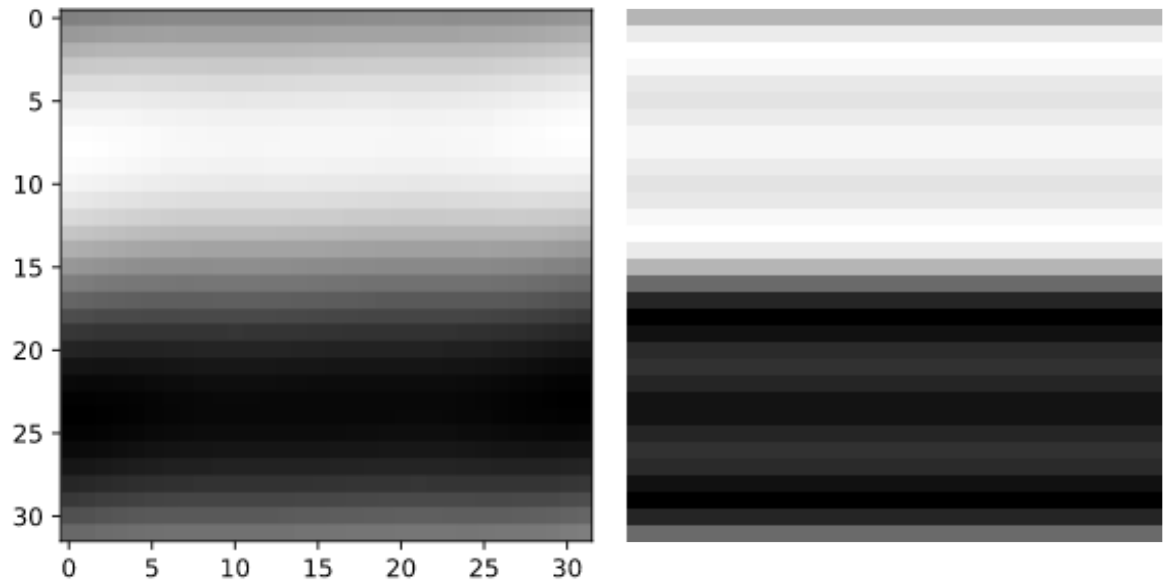


Figure 25: The low frequencies of the input image from Figure 17. The results from the circuit are on the left, while the results from [2] are on the right.

obtained from my results.

6 Conclusion and Future Work

Over the past two terms, I studied quantum image processing, focusing on different filtering algorithms. This required an understanding of hybrid images, quantum computing, and quantum circuit diagrams. I also looked at different image representations to see what would be the best way to represent an image on a quantum computer. I decided to implement the image representation and filtering algorithm proposed by Caraiman and Manta in [2] on a simulated quantum computer. I was not able to obtain a hybrid image this way however I was able to obtain the low frequencies of an image and compare my results to the results of Caraiman and Manta with mixed results. While it is interesting in theory to create a hybrid image on a quantum computer, the current technology available does not make it practical. Creating a hybrid image on a quantum computer would be more feasible to do when quantum computers are developed and more accessible to the public.

In the future, I would implement a better oracle so that the low and high frequencies are correct. Having these results would give the possibility to create a hybrid image when combined with another image. I would also like to run this on multiple different images to test and see if this low-pass filter would work as well. After working on this image filtering algorithm, I would focus on different image representations

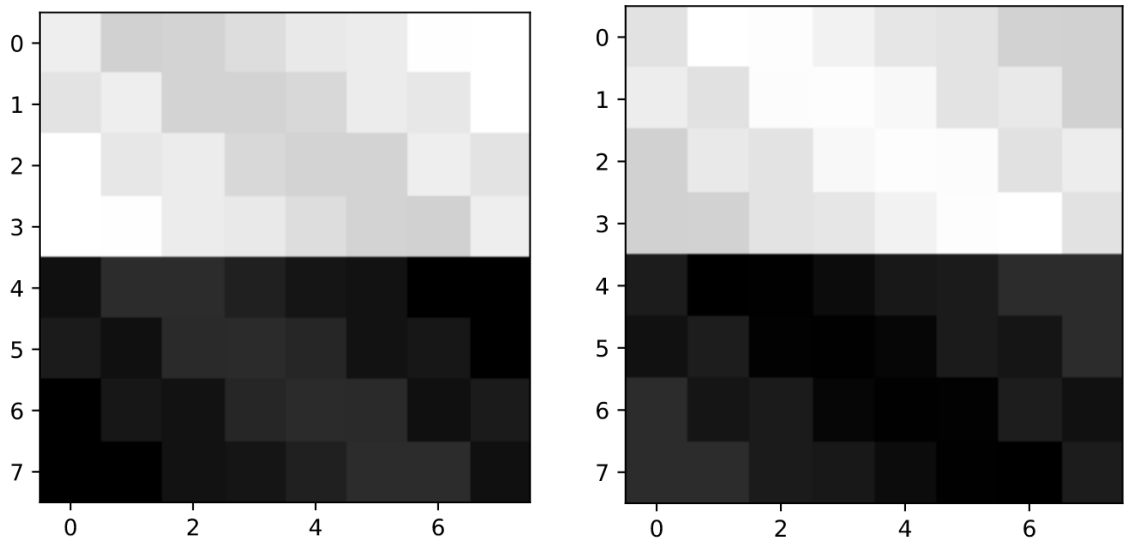


Figure 26: The output from the circuit that reset the IntegerComparator gates by inverting the operation when a check is made. The left image is the low frequencies, the desired ones, and the image on the right shows the high frequencies, the ones that are not wanted.

and filtering algorithms to also see what other possible ways there are to filter images.

7 Acknowledgements

I would like to first and foremost thank my thesis advisor, Professor Matt Anderson, for overseeing this project. There were countless times where I was lost and confused about what to do and he was always there to help me and guide me in the right direction and for that I am beyond grateful. I would also like to thank my fellow thesis student Jasper Bergh for helping me out and giving me advice about what to do for my research. And finally, to my friends and family who have kept pushing me to keep trying and always encouraging me to give my best effort.

References

[1] Yongquan Cai, Xiaowei Lu, and Nan Jiang. "A survey on quantum image processing". In: *Chinese Journal of Electronics* 27.4 (2018), pp. 718–727.

[2] Simona Caraiman and Vasile I Manta. "Quantum image filtering in the frequency domain". In: *Advances in Electrical and Computer Engineering* 13.3 (2013), pp. 77–85.

[3] Richard P Feynman. "Simulating physics with computers". In: *Int. J. Theor. Phys* 21.6/7 (1982).

[4] LIU Kai et al. "A strategy of quantum image filtering in frequency domain". In: *DEStech Transactions on Engineering and Technology Research* ameme (2016).

[5] Phuc Q Le, Fangyan Dong, and Kaoru Hirota. "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations". In: *Quantum Information Processing* 10.1 (2011), pp. 63–84.

[6] Pierre Mascarade. "Quantum Image Filtering in the Frequency Domain". In: *Quantum Information and Computation* (2016).

[7] Derek Murray. *Announcing TensorFlow 0.8 – now with distributed computing support!* Available at`https://tinyurl.com/ybkvbvcc` (2020/05/03).

[8] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[9] Aude Oliva, Antonio Torralba, and Philippe G Schyns. "Hybrid images". In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 527–532.

[10] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[11] Suzhen Yuan et al. "Quantum image filtering in the spatial domain". In: *International Journal of Theoretical Physics* 56.8 (2017), pp. 2495–2511.