# Representation of Quantum Images

Matt Harding and Aman Geetey

December 2018

## 1 Motivation

Data retrieval, encoding, and storage are processes considered easy and commonplace in classical computing. However, they remain challenging on the quantum realm. One of these challenges is storing and processing image data efficiently. As more people are delving into this area, the strive for researchers is to reduce the number of qubits used to represent quantum images and the number of simple operations necessary to store and operate on the images.This would be required to avoid decoherence and for the better utilization of the quantum communication channel. A relevant representation making use of quantum properties is essential for a useful treatment of images, putting mathematical devices like the quantum wavelet transform to avail.

## 2 Overview

In our research of the representation of quantum images we have reviewed literature spanning the algorithms which have created a foundation for the field and the ensuing representations which have paved the way to the state of the art. In addition we have expanded our knowledge by implementing an algorithm for ourselves in order to better understand the nuances of how it works. We go through the entire process of both implementing directly what was in the texts as well as working through the aspects which are not covered in detail in what we've read. The results of our testing are touched on as well as the implications of the progress and shortcomings of our approach. We finish with a discussion of what we can look forward to in the future of this field and for ourselves as we continue our research on this topic.

After going through all the literature we can positively infer that the field of storage and retrieving images from a quantum computer still remains largely a nascent engineering application. The best available algorithms are the FRQI and NEQR and yet both of these algorithms cannot be used on a large, highly detailed image, because apart from the difficulties related to the actual implementation, we cannot extend these to rectangular images, which are used quite profusely in modern day computing

## 3 Literature Review

In order to be able to operate on images with quantum computers there first needs to be a way to represent an image in a quantum state. One of the first methods proposed for doing so is the Qubit Lattice. To understand this method it is important to remember the representation of a single qubit on the Bloch sphere. In order to capture the essence of an image there needs to be a way to detect the different frequencies which make up the image. The qubit lattice method makes use of hypothetical machines which can detect frequencies of light and convert them to a quantum state. In this representation qubits are set up in i rows and j columns to form a 2 dimensional matrix of qubits. These qubits will store the different frequencies (pixels) which make up the image. For each of the qubits set up this way there must also be k - 1 qubits in an identical state which will be set up behind the first qubit, forming a lattice structure. To store a color the machine detects a frequency it will store in a qubit, and in the k - 1 qubits behind it, which need to be identical. The frequency is stored in the angle $\theta$. This angle can store any frequency because of the continuous nature of the angle. This process continues until all of the pixels of the image are stored. As important as it is to have a way to store an image it is equally important to be able to retrieve the image. Once the image has been stored it can be retrieved by measuring the qubits corresponding to the different frequencies. The reason we needed k copies of each qubit storing a different frequency is because many measurements of the qubit are necessary to recover what the original state was. The more qubits which can be measured this way, the more accurate the measurement can be. Let $\alpha$ and $\beta$ be the two possible states a qubit can be in after it is measured. Also, let $M_\alpha$ be the number of measurements for which see a state of alpha and let $M_\beta$ be the number of measurements which see a state of $\beta$ after measurement. Using the fact that the probability of obtaining a state of $\alpha$ is $cos^2(\theta/2)$, we get that as k approaches infinity, $cos^2(\frac{\theta}{2}) = M_\alpha/(M_\alpha + M_\beta)$. Then, solving for $\theta$, the angle can be converted back into the frequency that was stored in that qubit. Doing the same procedure for each set of qubits with identical state, all of the frequencies of the original image are recovered, and the full image is retrieved.

After the Qubit Lattice proposal for storage of quantum images, researchers began developing representations which will solve other problems which arise when dealing with quantum images. One of those issues is efficient compression of quantum images. In order to implement an efficient compression algorithm for quantum images Latorre et. al. devised a new representation of an image known as a real ket. To get the image into this representation the images pixels are split into 4 quadrants, with each one corresponding to a different numerical representation (1 for upper left, 2 for upper right, 3 for lower left, and 4 for lower right). To get to finer grained levels in the image each one of these quadrants is again split into 4 groups, with the same orientation of numbers representing the different quadrants. This process can be continued all the way down to the individual pixels. Each of these regions is represented in the real ket by a qudit (a quantum state with 4 basis states), so that at the second level, the ket will include a qudit for the quadrant at the highest level, followed by a qubit which represents the position of the pixel within the quadrant of the higher level.

The real ket representation given above was an improvement in quantum image representation, but to reduce the number of qubits necessary to store an image and efficiently create the state of the image a new algorithm was created called Flexible Representation of Quantum Images (FRQI). In FRQI representation, the image is stored in a state given by the following:

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

in which theta encodes the colors of the image and $|i>$ encodes the positions of the pixels in the image. Getting from an an initial state $(|0>^{\otimes(2n+1)})$ to the FRQI state requires the use of a unitary operation, which we will call P. First, Hadamard gates are applied to each of the qubits in the initial state. Using controlled rotation gates on the current state transforms it into the FRQI state. Referring to the controlled rotations as R and the Hadamard gates as H, we can say that P = RH. This process uses 2n Hadamard gates and $2^{2n}$ controlled rotations. These controlled rotations can be implemented by $C^{2n}R_y(2\theta)$ and NOT operations. It has also been shown that $C^{2n}R_y(2\theta)$ can be broken down into $2^{2n}1$ simple operations $R_y(\frac{2\theta_i}{2^{2n}-1})$and $R_y(\frac{-2\theta_i}{2^{2n}-1})$, and $2^{2n}-1$ NOT Operations. Therefore, the total number of simple operations to get from the initial state to the FRQI state is $2n+2^{2n}(2^{2n-1}-1+2^{2n}-1-2) = 2^{4n}-3(2^{2n})+2n$ which is quadratic in $2^{2n}$. The number of gates used to get to the FRQI state can still be quite large when considering the depth of large images how many pixels large images require. In order to reduce the number of simple gates required, a process called Quantum Image Compression can be used. The majority of the gates in the transformation process are rotation gates using some angle $\theta$. Making use of the fact that human vision is limited and there are many colors which are indistinguishable to humans, the angles encoding the colors in the image can take discrete values. Since controlled rotation operators with the same angle affect pixels of the same color irrespective of its position the same way, these rotations can be grouped together by rotation angle. This, along with the fact about human vision, can greatly reduce the number of rotation gates necessary to carry out operations.

Along with the transformation and retrieval of images, there are also other operations to be performed on quantum images. With FRQI these operations can be done using unitary transformations. These operations are grouped into 3 categories, each with its own unitary transformation: those dealing with colors only (e.g. color swapping), colors at specific positions (swapping colors at specific positions), and combinations of colors and positions. With FRQI, the number of simple gates needed for operations in the first category is only one gate. The number of gates necessary for operations in the second category depends linearly on the number of positions involved in the operation. The number of gates necessary for operations in the third category depends on what the specific operation is, but a unitary with $O(log(N))$ is used, for example the Quantum Fourier Transform, quantum wavelet form, etc. the complexity becomes $O(log(N))$. The complexities of these operations show a reduction in complexity from previous algorithms and show the power of FRQI representation of quantum images.

The next step in quantum image processing is not only representing, retrieving, and doing operations on quantum image, but also securing the image by use of encryption. While procedures for this process have been proposed, the Flexible Representation of Quantum Color Images proposes the most easily applicable and efficient approach yet to the encryption 3 of quantum images. The FRQCI model requires $2n+1$ qubits for an image of size $2^nX2^n$. This model stores the color information and position of each pixel in a balanced superposition state.The initial state needed for this method is a state of $2n + 1$ qubits all in the state $|0>$. Hadamard gates are then applied to each of the first 2n qubits, leaving the last in the $|0>$ state. Then $2^{2n}$ controlled rotation gates are applied to this state to get the final FCRQI state. The qubit containing the color information is represented in the form $cos(\theta_k/2) + e^{i\phi_k} sin(\theta_k/2)$. In this model the R color information is stored in $\theta$ and the G and B information is stored in $\phi$.

Color operations on images stored in the FRQCI state are relatively simple because of the way the state stores color. For color changes in R, since R is stored only in $\theta_k$, the change is only allowed to change $\theta$ and not $\phi$. G and B information is only stored in $\phi$, so a change in G or B is only allowed to change $\phi$ and not $\theta$, which can be done with a rotation about the Z axis. The same idea applies when swapping two colors. When R is one of the swapped colors, there is first a necessary rotation about an axis $n_k$, then a rotation about Z. However, when R is not involved in the swap, a rotation about the Z axis is all that is needed.

Of all the advantages of the FRQCI method, one of the most important is the method used to encrypt quantum images in an FRQCI state. First, the position of the pixels must be scrambled. To achieve this a random rotation of the qubit representing a pixel is done about an axis $n_k$ where $\phi_{k0}$ is the value of $\phi$ for the pixel being rotated, and then another random rotation is done about the Z axis. Essentially this process transforms the pixels from a 2D matrix into a vector, then rearranges them back into a matrix in a random order. The encryption process is not done, however, because an attacker could still learn information about the image because the color values have not changed, only their positions. This is done using a random rotation applied to the phase of the color qubit. A random rotation of $\theta_k$ and $\phi_k$ is applied to scramble the values of the colors. These random values of $\theta_k$ and $\phi_k$ act as the keys which are used to encrypt and to decrypt the image.

Most of the algorithms that we have encountered involve a profuse amount of qubits. They require an additional 1 qubit for the storage of the intensity of a pixel. Although Quantum representation gives us additional advantage wherein only $log(m)$ qubits are required for image representation, qubit economy in terms of the current hardware is still a problem. The 2-D QSNA representation encodes the pixel information, both the position and the intensity in a 2-D quantum state. The total image is represented by a matrix of appropriate size. For example, a 4 X 4 image would be represented by a 4 X 4 matrix. The elements of the matrix are the row and column location vectors where the location of the pixel is encoded using bit strings of appropriate length. The vectors are basically bit strings in Dirac notation (bra-ket) where column location vector refers to ket and row location vector refers to bra, respectively. Each element is a tensored product of each vector. The tensor product displays a 2-D quantum state in the Liouville space. This tensored approach utilizes the quantum property of superposition, which was absent from one of the very earlier and popular approaches, the Qubit Lattice. Also, observe that the matrix representation is not strictly N X N. We can obtain a matrix for 2 X 4 size and It would remain consistent with 4 the given picture. Therefore, an image with these dimensions would require 1 qubit (0 and 1) and 2 qubits (00,01,10,11) for row and column vector respectively. This expands the the applicability of quantum representation since now both rectangular and square images can be represented using 2-D QSNA, which is absent in some of the more popular methods, namely the NEQR and FRQI. Now that the pixel location is specified, the intensity of each constituent pixel of the image can also be encoded in the matrix. We know that the square of the amplitudes should all sum to one. We can encode the amplitudes in the following manner. For each and every scalar amplitude (p,q) we can define it to be equal to the square root of its amplitude, divided by the sum of total amplitudes of each pixel, where (p,q) represents the pixel on $p^{th}$ row and $q^{th}$ column. Defining in this way, the total number of qubits used is reduced. An additional advantage is the low possibility of the 2-D QSNA to undergo decoherence.

## 4 State of the Art

Quantum image representation began with purely hypothetical and impractical design with the qubit lattice. Over time the proposed representations have become more sophisticated and efficient. As of now, the state of the art in this field are the FRQI and NEQR representations. They use relatively low numbers of qubits to represent an image and make use of few numbers of gates to implement operations on the stored images. The FRQCI representation also adds an efficient way to encrypt quantum images using random rotations as the keys. In other respects the 2D QSNA representation, does have advantages over the previously mentioned representations, however. It claims it is a much simpler to implement and provides an avenue in which we can encode information about the location of a pixel and its intensity, and it provides an additional advantage over the other methods that it can represent rectangular images, instead of being limited to square images. However it remains a challenge to actually isolate the quantum states required for the methodology to work.

## 5 Our Implementation

We choose to implement the FRQI representation on QISKIT, one of the extensively used quantum simulators available online. The implementation was done partly for proof of concept that we can store and retrieve quantum image (and the information associated with it) successfully out of a quantum computer, and also to check how efficient the whole process is and how can we retrieve an image without compromising the overall structure. We would later see as ascertained by the results obtained that we need to create a lot of copies of the same image in the FRQI scheme in order to obtain the precise image that we had entered. The image format used was the bit matrix, as the FRQI scheme is supported only for those images. Separate algorithmic design is needed for the images based on vector graphics. We did the implementation for the 2 X 2 and the 4 X 4 case.

In a bit matrix format, each pixel has a certain combination of RGB values, hence both the position and the RGB values for a specific pixel needs to be transmitted for any application. The first part of our implementation was to find a specific algorithm for encoding the RGB array into something applicable in the quantum domain. The FRQI uses one qubit in superposition to encode the color information, and it does so by creating a superposition state involving an angle $\theta$. $\theta$ needs to be in the range $[0, \pi/2]$ Hence we need to convert three values, the RGB into one angle in the range mentioned above, and which should be distinct

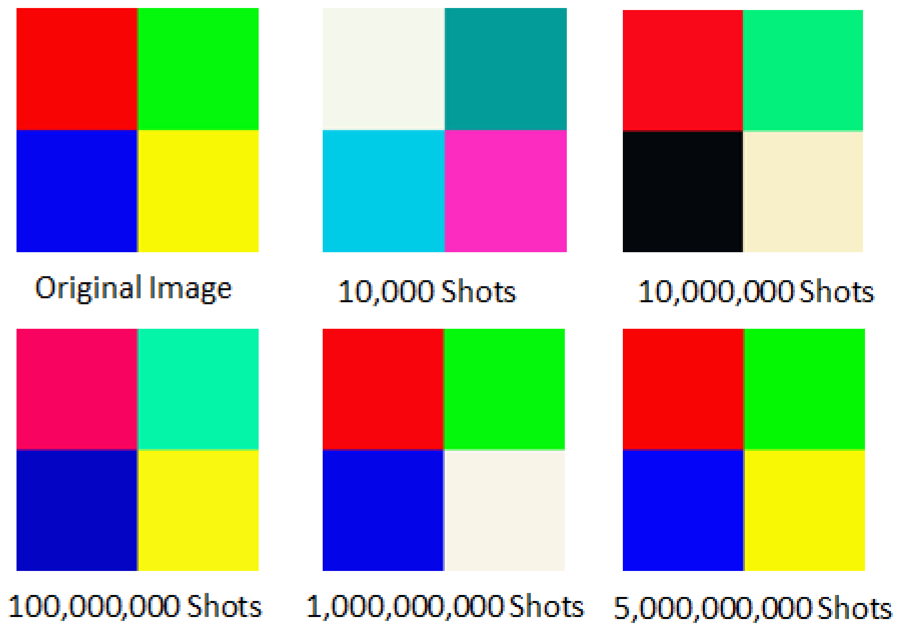


Figure 1: Images retrieved with differing numbers of measurements. With 5,000,000,000 shots the original image is retrieved.

for each of the different combinations of R, G, and B values.

The quantum circuit to represent the image involves two basic operations, to create the superposition of all states which were initially in the zero state, and then perform the controlled rotation operations which encode the information of color with each state. The final state I($\theta$) represents the qubits in superposition where each bitstring (n-1) represents the position of a pixel, tensored with one qubit which is used for encoding the information of color. The first part is pretty familiar, the cascaded hadamard operation. The next part is performing controlled rotations on each state in such a way that every operation acts like a counter and rotates only the last qubit of the current state with the angle required for that qubit.

## 5.1 Color Encoding

With any quantum image representation it is an important step to figure how the colors will be represented. With FRQI the colors of each pixel are encoded into angles in $[0, \frac{\pi}{2}]$. However, the papers we read on FRQI did not expand on how they actually went from RGB values to angles. To do this transformation we continually divided the space between 0 and $\frac{\pi}{2}$ to account for the red, green, and blue aspects of each pixel. For the red value we divided the space into 256 sections to represent each of the 256 possibilities for this color. To encode the green value each of the subdivisions created to stored the Red value was further divided into 256 regions, each of which would store a particular value for Green. To encode the Blue value we further divided each of the subdivisions for the Green value into 256 subdivisions, each of which would store a particular value for Blue. To implement this encoding we used

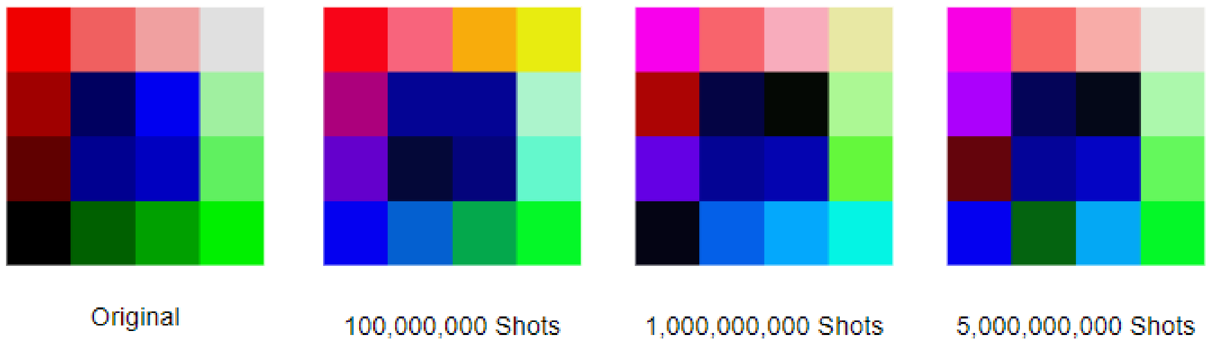


Figure 2: Images retrieved for 4x4 images. With more pixels, the chances that pixels will be inaccurate increases. Even with 5,000,000,000 shots some pixels are not correct.

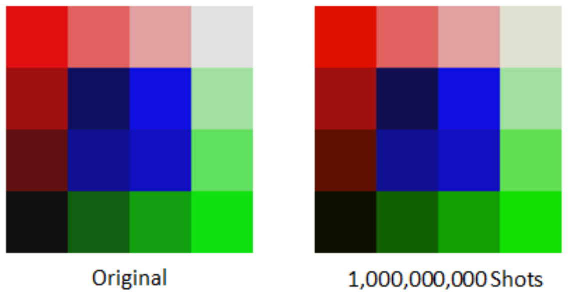


Figure 3: Retrieval of Image using only 16 values for R, G, and B.

the following equation,

$$\theta = \left(\frac{\pi}{2}\right) \times \left(\frac{R}{256} + \frac{G}{256^2} + \frac{B}{256^3}\right)$$

After retrieving the image angles this process was reversed to get back the original RGB values for each pixel. Once the Red value was found we were able to figure out which of the $256^2$ subsections which stored the Green value the angle resided in, to recover the Green value. Finally, it was discovered which of the $256^3$ subsections the angle resided in to recover the Blue value. Using this process of encoding and retrieving the colors means that the difference in angles encoding the Blue values are $\frac{\pi}{2} \times \frac{1}{256^3}$, so about 16,000,000th of a radian. This requires great precision, which can cause issues in retrieving accurate colors.

## 5.2 Storage and Retrieval

The storage of the image was implemented very much like was laid out in the papers which discuss FRQI. The process involves using Hadamards and Controlled Y-Rotations, as has been previously mentioned, to take a state of 2n + 1 qubits in the $|0>^{\otimes 2n}$ state to the $I(\theta)$ state.

Once we had an image stored in the $I(\theta)$ state, we were able to retrieve the color storing angles for each pixel in a similar way as was proposed in the qubit lattice. The state storing the image was measured many times, so we were very likely to get each possible basis state as the outcome enough times. With each of these measurements the results of the last 2n qubits corresponded to a particular pixel of the image. For each pixel outcome we stored the frequency of measuring 0 and the frequency of measuring 1 for that color qubit corresponding to that pixel. Using the frequency of 0 outcome for the pixel, $M_0$, and frequency of measuring 1 for the pixel, $M_1$, we can use the following equation,

$$Prob(|0>) = cos^2\left(\frac{\theta}{2}\right) = \frac{M_0}{M_0 + M_1}$$

to estimate the probability of measuring 0 for that pixel. Then we could solve for $\theta$ in order to retrieve an estimation of the color for that pixel. In this procedure, the more measurements that are made the more accurate the $\theta$'s will be.

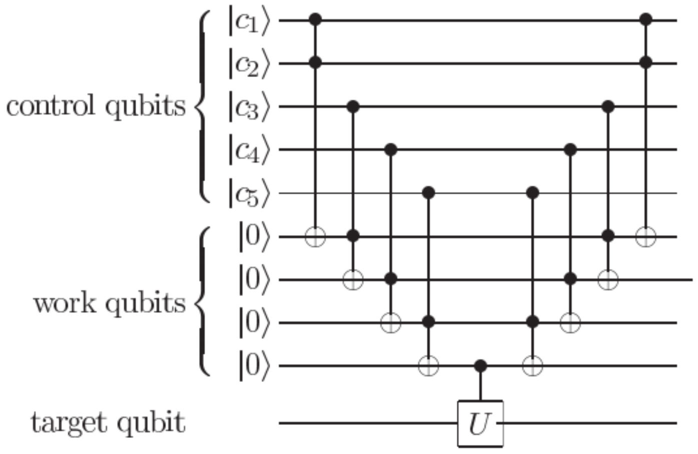
## 5.3 Preliminary Tests

When we first used this implementation to store and retrieve quantum images we used very simple images which were 2 pixels x 2 pixels. We were mainly interested in finding out how many shots (measurements) would be required to get back the original image. Because the difference in angles which needed to be distinguished to recover the blue value were very small (about 1/16,000,000 radians), accurate retrieval of images proved to be very difficult. We would need more than 5,000,000,000 shots (which would take a very long time with quantum simulations on our machines) in order to get the accuracies we would need. Instead of continually increasing the number of shots, we decided to reduce the number of possible values for R, G, and B from 256 to 64. This way the angles distinguishing between different blue values were not as small and accurate retrieval was achievable.

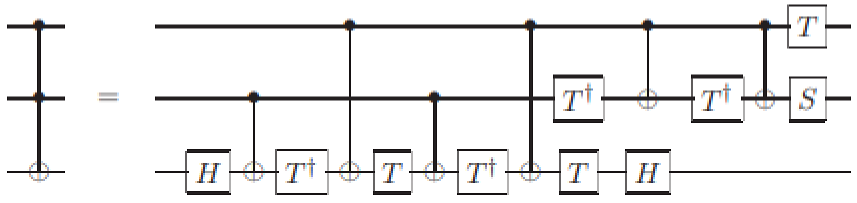
The results of using our implementation on a 2x2 pixel image are shown in Figure 1. We tried many different values between 10,000 and 5,000,000,000 and the image was not at all recognizable until about 10,000,000 shots. Finally, with 5,000,000,000 shots we were able to recover the original image.

## 5.4 The 4x4 Case

We have seen that using a combination of the NOT and the CNOT gates we can implement a counter like circuit which converts the current state to $|1111>$ and we can then apply the controlled rotation on the last qubit which takes it to $(cos(\theta_i)|0> + sin(\theta_i)|1>) \otimes |1111>$, we would have to do a 4 qubit controlled rotation in order to achieve that. As we have seen in the previous cases, we can break these higher operations down into smaller parts, for example, in the 2 X 2 case, we broke down the 2 qubit controlled operation into operations that consisted of only CNOT and Controlled- Y rotations. A similar breakdown can be achieved for the 4 qubit controlled case. However, as the number of controlling qubits increases, the breakdown as mentioned in the paper becomes more difficult to obtain. Extensive circuit analysis is needed for the application. So for the implementation of the controlled operation, we instead found a breakdown of higher order controlled operations with the use of toffoli gates. According to the scheme we can breakdown any higher order operation using Toffloli gates and some ancilla qubits. The circuit is given below.



This breakdown compensates in terms of the actual circuit design complexity but in terms of the computational complexity this breakdown is actually more costly than the one specified in the paper. As is evident from the fact that we are using Toffoli gates for the breakdown which themselves need to be build from universal gates. A breakdown of toffoli gates in terms of H, T, S and CNOT gates is given below.



For each controlled rotation, we would need at least 6 Toffoli gates, coupled with one controlled -Y Rotation gate and we would need to repeat that 16 times for the 4 X 4 case.

### 5.4.1 Other Image Operations

In addition to storing and retreiving images we were interested in implementing operations on the images while they were stored in the $I(\theta)$ state. One of these operations was pixel shifting. This refers to shifting each pixel either to the right by some factor. This was implemented by a series of controlled gates. First a triple controlled NOT gate was used with the first three qubits as the controls and the last position qubit as the target. Then a toffoli gate was used with the 3rd qubit as the target and the 1st two as the control. Next, a CNOT gate was used with the second qubit as the target and the first as the control and finally, a NOT gate on the first qubit. This operation can be repeated to get the the desired number of shifts. Figure 4 shows the results of shifting each pixel one position to the right in our 4x4 image.

Another operation we implemented was color manipulations of certain pixels, in order to

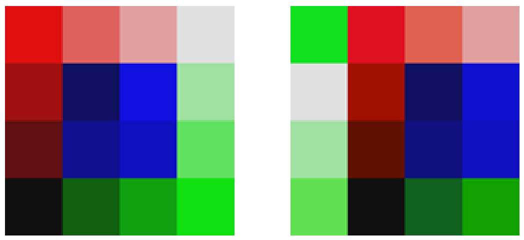


Figure 4: Left is the original image. Right is the image after every pixel has been shifted to the right.

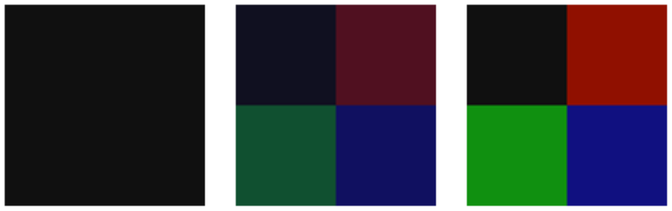


Figure 5: The progression of increasing the intensity of the pixels in R, G, and B. First increase is 4 levels in a 16 value system, and the second is an increase in 8 levels.

either increase or decrease the intensity of R, G, or B for those pixels. In order to achieve these manipulations further controlled rotations were used to rotate the qubits corresponding to the pixels for which color manipulation was desired. To increase the intensity of a particular color the qubit would be a rotation by the size of the subdivisions for that color, so $(\frac{\pi}{2} \times \frac{1}{256})$ for red, $(\frac{\pi}{2} \times \frac{1}{256^2})$ for green, and $(\frac{\pi}{2} \times \frac{1}{256^3})$ for blue. To decrease the intensity of this color the same process is used, but with the negation of the angle used to increase it. The results of increasing the intensity of different colors in different pixels are shown in figure 5. Increasing or decreasing the intensity of multiple colors in a single qubit can also be done in a similar fashion.

## 5.5 Implementation Review

We have have learned much about the nuances of the FRQI algorithm from our simulation. Being able to store even simple images and retrieve them with satisfactory precision, as well as doing operations on an image while in the quantum state is a big step in our understanding of the field of quantum image representation. However, we were unable to fully retrieve an image using the the full 256 RGB values generally used to store images. We hope to be able to refine our encoding mechanisms to account for this as well as make our implementation more efficient in general. Implementing different algorithms as well as more operations on quantum images could further enhance our understanding of this field in the future and we look forward to expanding on this work.

## 6 Future Work

As far as quantum image processing has come in recent years, there are still lots of problems waiting to be solved. In particular this area has a similar problem to all in the field of quantum computing: implementation on an actual quantum computer. While simulations have allowed many issues to be worked out, getting working digital to quantum and quantum to digital interfaces are vital to future work in the area. This can be worked on once it becomes more practical to produce quantum computers. Another open problem is the further reduction in the number of qubits necessary to represent an image. When being stored the problem of decoherence can become an impediment to the storage of any information in a quantum computer. The more qubits used, the more probable it becomes that one of the qubits will react with the outside world and lose its information. Storage of quantum images is essential to many quantum machine learning applications, such as image recognition. This requires the quantum computer to have the ability to recognize different colors within the quantum state of the image, which is an area of future research in this field.

Bit matrix or raster images are one of the common methodologies for the representation of images in a computer. Most of the quantum image representation algorithms that we have seen focus only on this pattern. The bit matrix representation, however has some disadvantages. The image is defined for a certain size (resolution) and gets blurred when we try to change the resolution. Another method of image representation is called the vector graphics. It uses mathematical equations to alter the properties associated with a certain image. The implementation is of course more complicated than the bit matrix representation which is quite intuitive in its structure. Future work could be geared towards finding an analogue of these methods in quantum computing. The raster images are also generally larger in file size compared to vector graphics.

## 7 References

- Venegas-Andraca S. E.: DPhil thesis: Discrete Quantum Walks and Quantum Image Processing. Centre for Quantum Computation, University of Oxford, 2006.
- Latorre, J.I.: Image compression and entanglement. arXiv:quant-ph/0510031 (2005)
- Le P.Q., Dong F., Hirota K.: A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Inf Process (2011) 10:6384.doi:10.1007/s11128-010-0177-y
- Panchi Li and Xiande Liu: Color image representation model and its application based on an improved FRQI. International Journal of Quantum Information. Vol. 16, No. 1 (2018)
- Madhur Srivastav, Subhayan R. Moulick and Prasanta K. Panigrahi : Quantum Image Representation Through Two-Dimensional Quantum States and Normalized Amplitude. arXiv:1305.2251