King Fahd University of Petroleum & Minerals College of Computer Science and Engineering Department of Computer Engineering

COE-530 (Quantum Computer & Architecture)

Quantum Signal Representation and Processing Using Quantum Fourier Transform

Sarah Alwadaah

Abstract

The interest in quantum image processing is growing in recent years as classical techniques can improve their performance by exploiting quantum properties. Quantum Fourier transform (QFT) is one of the most important algorithms of quantum computing and quantum information processing and the heart of many applications of quantum computing and simulation. One of the classical Fourier Transform important application is in image and signal processing, here lies the importance of exploring QFT as it demonstrates exponential speed ups compared with best classical counterparts[5]*. In this paper we'll analyze representation of images in Hillbert space and the QFT role in quantum information processing, image processing in particular.*

Term-202

May 18th, 2021

1. Introduction

With every year, technology advances rabidly and the amount of information being used increases, specially visual data as Image processing is extensively used in fast growing markets. Storing and processing massive visual data efficiently is the key technology to be developed urgently. Quantum information processing (QIP) [1] that uses qubits instead of bits has two key features: one is that quantum storage capacity increases exponentially in comparison with the traditional storage as n qubits can store 2^n data at the same time [2], the other relies on the unique computing performances of quantum coherence, entanglement and superposition of quantum states.

A subfield of QIP is quantum image processing that deals with processing and representation of quantum images. This field is new and has been gaining attention lately with few models developed to convert images into quantum computational basis like qubit lattice, flexible representation of quantum images, QImR and many others. Newer model preform better and they differ in the type of image (colored/grayscale) or if video visual representation is needed.

One of the core algorithms of quantum information processing is quantum Fourier transform. The complexity of implementing QFT on 2^n elements is $O(n^2)$[2] while fast Fourier transform, one of the best classical algorithms, computes the discrete Fourier transform (DFT) with the complexity $O(n2^n)$ [7]. QFT offers exponential speed up of the computation of Fourier transforms

Here, we'll implement signal output into computational basis with the newly introduced model for quantum image representation, flexible representation of quantum images (FRQI), we'll process the output into Fourier basis with quantum Fourier basis (QFT). In this paper, the terms signal and images are used interchangeably as handling images if the same as signals after reading.

2. Background

This paper's objective it to represented images to a computational basis using flexible representation of images (FRQI) and processing the image with Quantum Fourier Transform (QFT). Below is a brief exploration of the nature of the two process:

2.1 Flexible Representation of Quantum Image (FRQI)

Quantum image processing, which utilizes the characteristic of quantum parallelism to speed up many processing tasks, is a subfield of quantum information processing. The first steps taken in the area of quantum image processing have involved proposals on representations to capture and store the image on quantum computers. Various representations for images on quantum computers were proposed, one was flexible representation of quantum image (FRQI) was proposed by P.Q. Le can be used for polynomial image preparation, image compression, and image processing techniques. The technique is based on integrating the pixel value and position information in an image into a (n+1) qubit quantum states as follows[4]:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}(\cos\theta_k\,|0\rangle + \sin\theta_k|1\rangle)\ |k\rangle$$

Where the angle θ_k in a single qubit encodes the pixel value of the corresponding position 2D $|k\rangle$. The results of simulations for different experiments, including images storage, retrieval and line detection in binary images, are done by applying the quantum Fourier transform as the processing operation[4][6]. For the 2D images, the position information $|k\rangle$ includes two parts: the vertical and horizontal coordinates. In $2n$-qubit systems for preparing quantum images, the vector $|k>=$

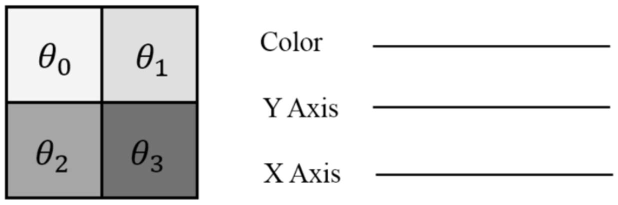
$$|y\rangle|x\rangle = |y_{n-1}y_{n-2}\cdots y_0\rangle|x_{n-1}x_{n-2}\cdots x_0\rangle,$$

$$x_j, y_j \in \{0,1\},$$

for every j = 0, 1, . . . , n encodes the vertical location by means of the first n-qubit $y_{n-1}y_{n-2} \cdots y_0$ and the location along the horizontal axis by using the second n-qubit $x_{n-1}x_{n-2} \cdots x_0$. The FRQI state is a normalized state, *i.e.*, $\||I(\theta)\rangle\| = 1$, as given by[11]:

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

An example of a 2 × 2 image in quantum FRQI can be seen represented as:



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

As a theory, given a vector $\theta = (\theta_0, \theta_1, \cdots, \theta_{2^{2n}-1})$ of angles, there is a unitary transform P that is composed of polynomial number of simple gates to turn quantum computers to the FRQI state. To achieve this unitary transform P, from which the initialized state $|0\rangle^{\otimes 2n+1}$ is changed to $|H\rangle$ and then to $|I(n)\rangle$[11] where $|i>$ is same as $|k>$ mentioned above:

$$\begin{tikzcd}
 & |H\rangle=\frac{1}{2^n}|0\rangle\otimes\sum_{i=0}^{2^{2n}-1}|i\rangle \arrow[dr, "\text{Step 2}"] & \\
|0\rangle^{\otimes 2n+1} \arrow[ur, "\text{Step 1}"] \arrow[rr, "\mathcal{P}"] & & |I\,(n)\rangle=\frac{1}{2^n}\sum_{i=0}^{2^{2n}-1}(\sin\theta_i|0\rangle+\cos\theta_i|1\rangle)\otimes|i\rangle
\end{tikzcd}$$

First, applying the transform $H = I \otimes H^{\otimes 2n}$, where *I* is the 2D identity matrix and *H* is the Hadamard gate, on $|0\rangle^{\otimes 2n+1}$, produces the state $|H\rangle$. Then, consider the rotation matrices $R_y(2\theta_i)$ (along the y-axis by the angle $2\theta_i$) and controlled rotation matrices R_i ($i = 0, 1, \cdot\cdot\cdot, 2^{2n} - 1$):

$$\mathcal{H}(|0\rangle^{\otimes 2n+1}) = \frac{1}{2^n} \otimes \sum_{i=0}^{2^{2n}-1} |i\rangle = |H\rangle$$

$$R_y(2\theta_i) = \begin{pmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{pmatrix},$$

$$R_i = (I \otimes \sum_{j=0, j\neq 1}^{2^{2n}-1} |j\rangle\langle j|) + R_y(2\theta_i) \otimes |i\rangle\langle i|.$$

the unitary transform $P = RH$ turns quantum computers from the initialized state, $|0\rangle^{\otimes 2n+1}$ to the FRQI state in the first equation.

2.2 Quantum Fourier Transform (QFT)

The original Fourier transformation was discovered in the early 1800s by the French mathematician Joseph Fourier, who was at that time focusing on the analytic theory of heat. Fourier Transform, or, in particular, Discrete Fourier Transform (DFT) is a tremendously useful mathematical tool that can be used to process signal frequencies. As such, it has a very rich spectrum of application scenarios across a wide array of scientific domains such as digital sound processing, image processing or numerical algorithms

Quantum Fourier transform was discovered in June 1994 internally in an IBM Research Division by an American mathematician Don Coppersmith. It provided a breakthrough for the famous factoring algorithm developed by Peter Shor later in the same year. It provides a dramatic speed up over the classical counterpart. In the classical case, the most efficient implementation of the Fourier transform, the Fast Fourier Transform, have the complexity of $O(n2^n)$ to compute the result. On the other hand, the quantum version of it can achieve the same result with $O(n^2)$ algorithm complexity, resulting in a spectacular exponential performance improvement. Quantum Fourier transform is analogous to FFT in the sense that quantum state representation is naturally power of two based, while the most common FFT algorithms require an even power of two for the number of data points used. QFT in a nutshell, is the transform which takes the amplitudes of a state of N dimensions and computes the Fourier transform on these amplitudes. It can be performed 2^n elements as[7]:

$$\sum_{x=0}^{N-1} a_x |x\rangle \rightarrow \sum_{x=0}^{N-1} \tilde{a}_x |x\rangle = \sum_{x=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{-xy} a_y |x\rangle$$

Which is also a unitary operation as seen below[7]:

$$\begin{aligned} U_{QFT}U_{QFT}^{\dagger} &= \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \omega_N^{yx} |x\rangle\langle y| \sum_{x'=0}^{N-1} \sum_{y'=0}^{N-1} \omega_N^{-y'x'} |y'\rangle\langle x'| \\ &= \frac{1}{N} \sum_{x,y,x',y'=0}^{N-1} \omega_N^{yx-y'x'} \delta_{y,y'} |x\rangle\langle x'| = \frac{1}{N} \sum_{x,y,x'=0}^{N-1} \omega_N^{y(x-x')} |x\rangle\langle x'| \\ &= \sum_{x,x'=0}^{N-1} \delta_{x,x'} |x\rangle\langle x'| = I \end{aligned}$$

3. Literature Survey

The main objective of this paper is to processing a seismic image by preparing it in a qautnum representation then processing to quantum Fourier basis. Processing images with QFT is very new and not many literature is found in this area and to the best of our knowledge all were used in either watermarking or enhancing security[9]. Only one paper was found as the first experimental study towards practical applications of quantum computers for image processing was made recently in 2019 [9] by Ola Al-Taani, Ali Mohammad Alqudah and Manal Al-Bzoor where they processed images classically using FFT and in a quantum context using QFT to compare between the two.

In that study, the model used to represent the image is Quantum Probability Image Encoding (QPIE), a model for quantum image representation [9] that encodes the image pixel values in probability amplitudes and their positions into computational basis states. The model speeds up the computation time required for either image representation, which represent its main potential in the field for highly efficient image processing in the big data era.

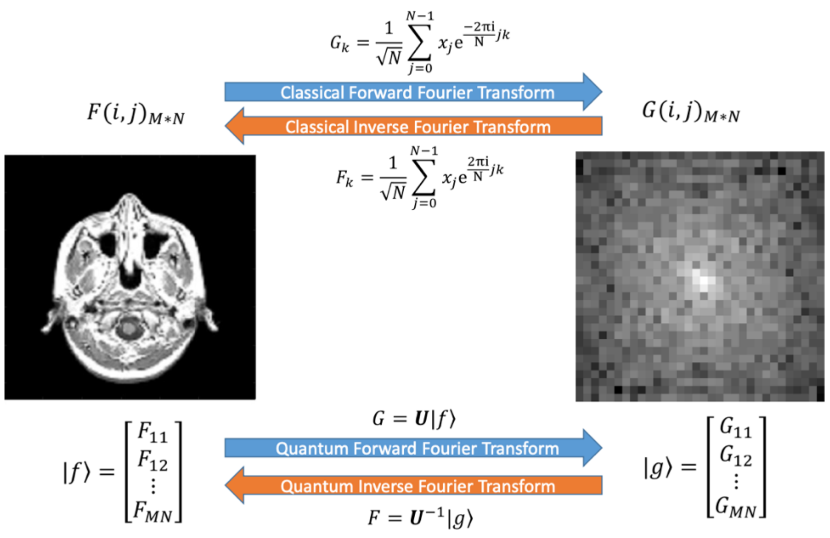
Given a 2D image $F = (F_{i,j})_{MxL}$, F represents the pixel value at position (i, j) with $i = 1, \ldots , M$ and $j = 1, \ldots , L$. The quantum representation model adopted in this paper is to convert the $(M \times L)$ matrix F into a column vector f mapped into a pure quantum state of n qubits where $|k\rangle$ represents the (i, j) position for each pixel with $n = \lceil log_2(ML)\rceil$; and c_k represents the pixel value[9]:

$$|\mathrm{f}\rangle = \sum_{\mathrm{k}=0}^{2^{\mathrm{n}}-1} \mathrm{c_k}\,|\mathrm{k}\rangle \quad , \qquad \mathrm{c_k} = \begin{cases} \dfrac{\mathrm{F_{i,j}}}{\sqrt{\sum_{\mathrm{k}=0}^{2^{\mathrm{n}}-1} \mathrm{F_{i,j}}^2}}, & \mathrm{k} < \mathrm{ML} \\ 0 & \mathrm{k} \geq \mathrm{ML} \end{cases}$$

Classically, the image is represented as a matrix; and it is encoded with 2^n bits. The image transformation is conducted by matrix computation. In contrast, the same image is represented as a quantum state; and encoded in n qubits. The quantum image transformation is performed by unitary operator **U**[9]:

$$U = \mathrm{QFT}_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & w_n & w_n^2 & w_n^3 & \cdots & w_n^{N-1} \\ 1 & w_n^2 & w_n^4 & w_n^6 & \cdots & w_n^{2(N-1)} \\ 1 & w_n^3 & w_n^6 & w_n^9 & \cdots & w_n^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & w_n^{N-1} & w_n^{2(N-1)} & w_n^{3(N-1)} & \cdots & w_n^{(N-1)(N-1)} \end{bmatrix}$$

The result of the paper after implementing different measures of image quality for both FFT or QFT images, in general, QFT results are shown to be similar to FFT in all cases. It concluded that it might be of interest not to read the output image itself only but to find some significant statistical characteristics or useful global features about image data. It is a possibility to explore this area instead of decoding the image explicitly. And output of the image in both after and before implementing the transform:

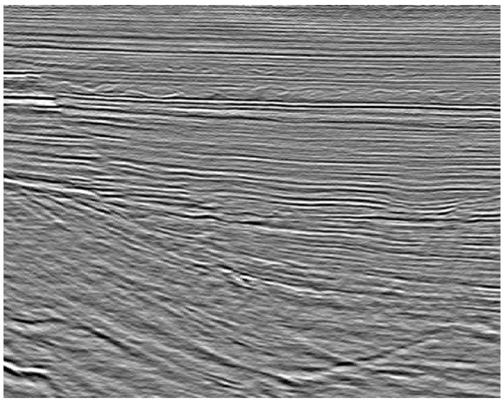

$G_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{-2\pi i}{N} jk}$
Classical Forward Fourier Transform
$F(i,j)_{M*N}$
Classical Inverse Fourier Transform
$G(i,j)_{M*N}$
$F_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i}{N} jk}$
$G = \boldsymbol{U}|f\rangle$
$|f\rangle = \begin{bmatrix} F_{11} \\ F_{12} \\ \vdots \\ F_{MN} \end{bmatrix}$
Quantum Forward Fourier Transform
Quantum Inverse Fourier Transform
$|g\rangle = \begin{bmatrix} G_{11} \\ G_{12} \\ \vdots \\ G_{MN} \end{bmatrix}$
$F = \boldsymbol{U}^{-1}|g\rangle$

The objective of the above paper to compare between classical and quantum Fourier transform, it also uses QPIE. In these two points this paper is different, as the main goal to process signals in quantum Fourier transform and we use FRQI an a quantum image representation model.

4. Theoretical Model

4.1 Classical part:

In image proccing we apply filters and algorithm based on numbers and mathematical models, the first thing we need to do is convert our image to an n-dimentional array of number with each element representing the pixel value for that specific index. For greyscale image, array is 2 dimentional (M-by-N) array while it's 3 dimentional (M-by-N-by-3) for colored ones[8]. Here since seismic signals data are processed to output grayscale representation of the signal, our array will be 2D with each element a single number pixel value representing the brightness of the pixel. In byte image, this number is stored as an 8-bit integer giving a range of values from 0 to 255 where 0 is black and 255 is white[9]. The result of converting 1212x960 image to an array:



$\longrightarrow$

```
[[114 168 168 ... 124 124 124]
 [114 168 168 ... 124 124 124]
 [110 210 210 ... 120 120 120]
 ...
 [161 133 133 ...  94  94  94]
 [161 133 133 ...  94  94  94]
 [161 126 126 ...  70  70  70]]
```

This array is very big since our image is 1212x960, this will consume computational power in processing and will require high number of qubits later. The next step is resize this array to a 32x32 array, small enough to do simpler work without distorting the original enough to lose information. The resizing is done to turn a 1212x960 array to a 32x32 is by finding the ratio between the two sizes then performing 2D average calculate to get values of elements in the new array:

$$\left\lceil \frac{1212}{32} \right\rceil = 38 \text{ elements} \qquad\qquad \left\lceil \frac{960}{32} \right\rceil = 30 \text{ elements}$$

Now we knoe will need to find the average of the first 38x30 elements in the original array to make up the first 1 element in the new 32x32 array. The non-existent elements needed in the above calculation, at the end of the array edges, will be considered as white padding of 0 value. The original array is to big (1212x960 =**1163520** elements), the software (Matlab) is not able to show it. As a simplified example we consider the 8x8 array we want to make a 4x4 so we take each 2x2 array average:

```
[100 110  90 224 255 120 190  24]
[255 120 190  24 200 110  76 224]
[200 110  76 224   0 210  56  98]
[ 20 210  56  98 100 110  90 224]
[100 110  90 224 255 120 190  24]     [146 132 171 128]
[255 120 190  24 200 110  76 224]  →  [135 113 105 117]
[200 110  76 224   0 210  56  98]     [146 132 171 128]
[ 20 210  56  98 100 110  90 224]     [135 113 105 117]
```

(100+110+255+120)/4 = 585/4 = 146
We take the integer value, eliminating any fractions. We also move from left to right going down row by row. The next calculation: (90+224+190+24)/4 = 132 and so on, on the right is the 4x4 new array. As for our original image we want to resize after converting to an array of number, the same concept was applied programmatically to produce the following:

```
[[114 168 168 ... 124 124 124]    [133 128 129 ... 117 119 120]
 [114 168 168 ... 124 124 124]    [135 137 133 ... 140 144 142]
 [110 210 210 ... 120 120 120]    [117 120 122 ... 131 130 130]
 ...                           →  ...
 [161 133 133 ...  94  94  94]    [141 138 113 ... 121 132 132]
 [161 133 133 ...  94  94  94]    [123 126 148 ... 141 135 126]
 [161 126 126 ...  70  70  70]]   [140 125 119 ... 121 119 129]
```

Now after preparing the image we move to the next step to encode in quantum states using FRQI to be able to use it in our QFT quantum algorithm.

3.2 Quantum Part

The main purpose of the Flexible Representation of Quantum Images (FRQI) [6] is to provide a quantum representation of images that allows an efficient encoding of the classical data into a quantum state. In this case, encoding the classical image into a quantum state requires a polynomial number of simple gates [2].

The quantum state representing the image the following normalized state where the basis is the pixel position and amplitude is the color value[6]:

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

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

With an example of a simple 2×2 image given below, corresponding θ angles (color encoding) and associated kets (position encoding)[6]:

$\theta_0, \|00\rangle$	$\theta_1, \|01\rangle$
$\theta_2, \|10\rangle$	$\theta_3, \|11\rangle$

In this representation, angles θ_i equal to 0 means the pixels are black and θ_i values are equal to $\pi/2$ means the pixel is black[6].

In our case, to reach the final state above, two steps are needed: Hadamard transforms and controlled rotation transforms are needed to achieve the unitary transform above from which the initialized state $|0\rangle^{\otimes 2n+1}$ is changed to $|H\rangle$ and then to $|I(n)\rangle$. Initially for each element in out array can be represented as $|0\rangle^{\otimes 2n+1}$, n here in the FRQI image size is 2^nx2^n since our array is 32x32, in FRQI it's 2^5x2^5, n is 5. So the first element positioned 0 (array index 0) is the quantum state $|0>^{\otimes 11}$. The first step we need to do to turn out states from the array into a superpositioned states by applying tensor product of 2n Hadamard matrices $H^{\otimes 2n}$. The resulted $H^{\otimes 2n}$ by multiplying tensor product 10 time will look like:

$$H = \frac{1}{\sqrt{2}}\begin{array}{c|cc} & \mathbf{0} & \mathbf{1} \\ \hline \mathbf{0} & 1 & 1 \\ \mathbf{1} & 1 & -1 \end{array}, \quad H^{\otimes 10} =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & & & \\ 1 & -1 & 1 & -1 & 1 & -1 & & & \\ 1 & 1 & 1 & 1 & & & \ddots & -1 & -1 \\ 1 & -1 & 1 & -1 & \ddots & & & -1 & 1 \\ 1 & 1 & & & & -1 & -1 & -1 & -1 \\ 1 & -1 & & \ddots & & -1 & 1 & -1 & 1 \end{bmatrix}$$

First, applying the transform H = I $\otimes$ H $^{\otimes 2n}$, where *I* is the 2D identity matrix and *H* is the Hadamard gate, on $|0\rangle^{\otimes 2n+1}$, produces the state $|H\rangle$,

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

In this paper, we'll try to process a seismic image from earth. the FRQI model for representing images is used to transform this image to computational basis. First we'll resize our image to 32*32 pixels. In FRQI, number of qubits used is 2n+1 here $2^5 = 32$ meaning we'll use 11 qubits.

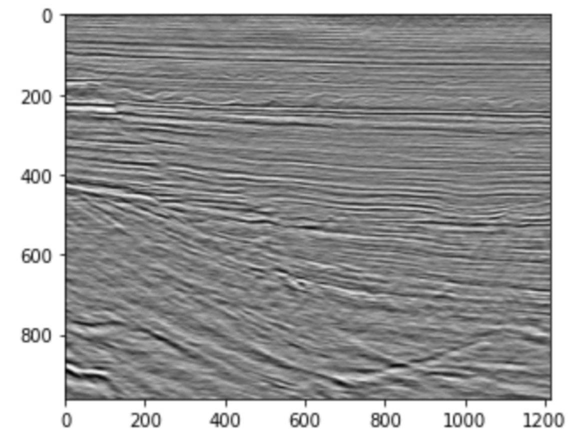
For the QFT, for each array element we calculate phase shift theta pi/$2^{(11-i)}$ for each qubit I, and perform swap in the middle. We measure the output and after simulation we return to 2D array after flattening then inserting the elements from the raveled array into the new array using the same kind of index ordering as was used for the flattening.

5. Implementation and Evaluation

This experiment was done using python with jupyter notebook. Qiskit was installed in the environment for the quantum part of the process in Mac OS. The ipynb attached show detailed steps with comments explaining different functions and process. The file also mentions references used for certain codes portions and python files.

First we import out needed libraries, qiskit circuit composer, numpy and mathematical functions, matplotlib for plotting outputs and PIL pillow to manage images. Then we create a circuit of 11 qubits for frqi that initializes the state with 2n+1 and n is 5 (refere to section 3). We also use 2 ancilla qubits for error correction that decrease error rate [reference in code].

Then we'll import our image that has the size 1212x960. We'll normalize the image by resizing and converting to a 2D array of pixel values.



```
Image as array values in grayscale 0 to 255

[[133 128 129 ... 117 119 120]
 [135 137 133 ... 140 144 142]
 [117 120 122 ... 131 130 130]
 ...
 [141 138 113 ... 121 132 132]
 [123 126 148 ... 141 135 126]
 [140 125 119 ... 121 119 129]]
```

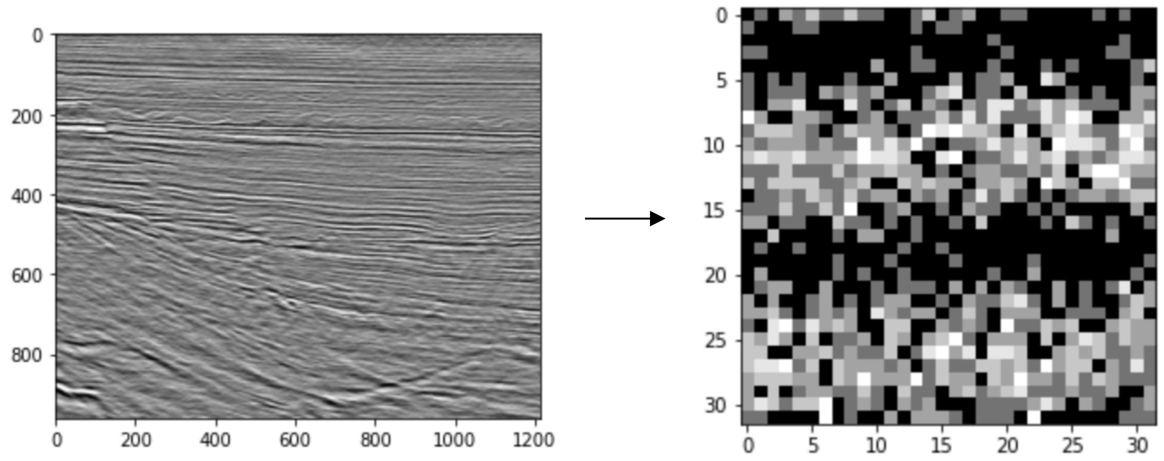
Then we flatten the array to 1D temporarily until we finish the quantum part as it significantly reduces the size for processing. Now, we encode our image to computational basis by using FRQI code [referenced github repository in the code] based on Qiskit. The code implements Hadamard and a series of polynomial gate to each pixel/element in the array encoding an image of size m = n x n (pixels) into log2(m) qubits.

Next, we apply QFT based on the Qiskit code referenced. For each qubit i in the 11 qubits, the function will perform pi/2^{11-i} controlled shifts and swaps to finally return a a circuit of depth 78924 which is the length of the longest path with a highest number of gates executed, hence the circuit cannot be viewed.

Now we simulate the code in qasm simulator by running it 5000 times and record the time. The simulation was done in 78.65 seconds, a little more than a minute.

```
time: (78.6580560207367, QasmSimulator('qasm_simulator'))  seconds
```

Finally, we encode the simulated output to appending each simulated count in a new array and reverse the normalization process to get it back to 2D array to finally get the output, a Fourier transform of our original image.



One of the main goals in quantum computing is to design quantum algorithms that are more efficient (i.e. faster) than their classical counterparts. This goal usually has an implicit assumption that both quantum and classical algorithms are to be executed on general-purpose computers. Unfortunately, the quest for QImP algorithms running on general-purpose computers has left behind the superiority of quantum image processing algorithms especially with respect to time complexity.

6. Conclusion:

In conclusion, this paper demonstrated the quantum image processing method using FRQI to represent images in computational basis and QFT to process to Fourier basis. The simulation proved to be fast very high run times and gave the correct output in Fourier transform although the process was not compared to classical Fourier transform. Another simulation can be done using other image representation model QImR as done on previous work mentioned in literature survey. The QImP model of representing images offers exponential speedup needs to be analyzed and tested.

7. References

[1] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631-633, 2014.

[2] A. Vlasov, "Quantum computations and images recognition," *arXiv:quant-ph/9703010*, 1997.

[3]S. Venegas-Andraca and S. Bose, "Storing, processing, and retrieving an image using quantum mechanics," *Proceedings of Quantum Information and Computation, International Society for Optics and Photonics*, vol. 5105, pp. 137-148, 2003.

[4]P. Le, F. Dong, and K. Hirota, "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations," *Quantum Information Processing*, vol. 10, no. 1, pp. 63-84, 2011.

[5]X. Yao, H. Wang, Z. Liao, M. Chen, J. Pan, J. Li, K. Zhang, X. Lin, Z. Wang, Z. Luo, W. Zheng, J. Li, M. Zhao, X. Peng, and D. Suter, "Quantum image processing and its application to edge detection: theory and experiment," *Physical Review X*, vol. 7, no. 3, pp. 1-14, 2017.

[6] Qiskit, "Quantum Image Processing - FRQI and NEQR Image Representations": https://qiskit.org/textbook/ch-applications/image-processing-frqi-neqr.html

[7] J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297-301, 1965.

[8] MATLAB Function Reference: imread Function

[9]©2003 R. Fisher, S. Perkins, A. Walker and E. Wolfart.

[10] Fei Yan [1,2,*], Abdullah M. Iliyasu [2,3] and Zhengang Jiang

[11]"Quantum Computation-Based Image Representation, Processing Operations and Their Applications " (2014)