# QCourse 570-1 Project Report: Implementation of FRQI Method using Qiskit

Guillermo "Bill" Gonzalez

University of Texas at San Antonio

**Abstract**

The FRQI (Flexible Representation of Quantum Images) method of Quantum Image Representation (QIR) was first published in 2009[8] and has remained one of the most referenced techniques in Quantum Image Representation. This paper describes an implementation of the FRQI method using an algorithm written in Python using the Qiskit library in order to fully understand some initial steps required moving towards performing real-world Quantum Image Processing (QImP).

**Keywords** FRQI, Quantum Image Representation

## 1 Introduction

Quantum Image Processing (QImP) presents an interesting realm in the quantum computing universe that is not as commonly discussed in quantum related media sources as are other surging areas of quantum computing such as encryption, optimization, chemistry, material science and the quantum internet. However, quantum image processing should not be ignored as it has the potential to further progress in many emerging deep tech fields such as autonomous vehicles, local and national security, product manufacturing and robotics to name a few. Some research has shown that quantum image processing could offer significant advantages over current classical methods. For example, edge detection using a quantum version of the Sobel method, QSobel, has been shown to be theoretically faster.[10] Of particularly interesting use case is edge detection related to astronomy or more succinctly, image processing of astronomical images otherwise known as space imagery.[6, 2]

In the realm of quantum image processing, much research has been done regarding the topic and related topics such as quantum image representation (QIR), quantum image compression, quantum image encoding, quantum image encryption, and quantum image retrieval. This paper describes a study in quantum image representation and the steps take to develop a working implementation of the FRQI circuit.

Developing a stable Quantum Image Representation is the first step to realizing the full potential of Quantum Image Processing. Even simple experiments are of interest in order to develop working models when we do climb out of the NISQ era.

Guillermo "Bill" Gonzalez: billgonzalez@webheadtech.com, http://www.webheadtech.com, Thanks to Dr. Abu Yakaryilmaz and QWorld.

## 2 Motivation

The amount of literature, including unpublished and published research papers and one textbook on the topic of quantum images[1] covering quantum image representations is increasing yearly. However it is hard to find easily reproducible working implementations of the techniques and methods to study in-depth in a local and controllable environment to facilitate further learning. There are at least two publicly available GitHub repositories[2, 3] as example implementations, but the team that wrote one of the examples did not have a focus on quantum image representation but on image classification. The second example did not seem to comprise a complete working implementation and if so, it only addressed hard-coded image sizes and utilized an alternate gate synthesis that created a distraction from the original circuit design. This paper documents the steps taken towards the development of working implementation to further personal research and satisfy a curiosity.

## 3 Literature Review

Papers covering various quantum image representation techniques, methodologies and algorithms were studied. There are now more than 30 various representations that have been presented and some are "enhancements" to some of the initial representations like FRQI and NEQR among others. Researchers have presented reviews of at least 15 methods[5] up to 28 methods[9]. Some researchers have even taken the time to categorize the various representations and study their complexity based on the number of qubits, the circuit depth and quantum volume.[5]

The main topic of this paper was to identify and implement one of these quantum image representation techniques so the focus was placed on FRQI due to the fact that it was one of the first proposed techniques and is easy enough to understand. Most of the seminal papers written covering FRQI were studied to gain an understanding of the basic methodology.[6, 10]. Much of the existing literature to this day include discussions of FRQI. In addition numerous technical papers related to improvements in FRQI were also reviewed. A video on an enhanced FRQI (EFRQI) was also viewed to gain a better understanding of the FRQI method.[7]

The basic idea behind this representation is that FRQI encodes the intensity value of a pixel by rotating a single qubit to a specific angle. Additional qubits are utilized to represent the pixel location of a specific pixel within an image.

A paper from 2018 identified two shortfalls of FRQI at that time which have since been overcome.[4]

1. We can only deal with small images
2. We cannot deal with rectangular images

The true goal of this literature review was to find information on actual programs that implemented any of these algorithms. Research was conducted on many of the different algorithms available like real ket, FRQI, NEQR and many others, but because most implementation examples focused on FRQI, focus was placed on FRQI for its advantages, simplicity and available information.

```
[[  0   0   0   0]
 [255 255 255   0]
 [255 255 255   0]
 [  0   0   0   0]]
```

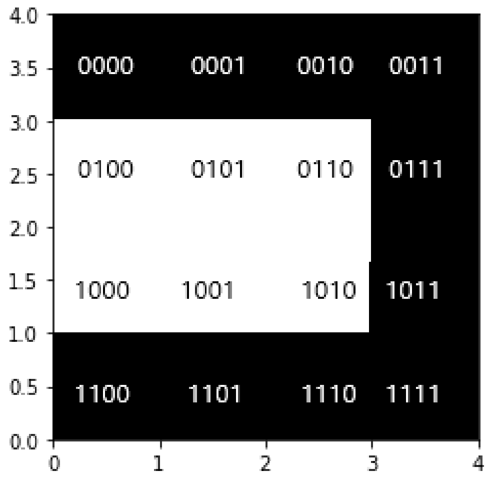

Figure 1: Initial sample image selected for this paper.

## 4 Methods

Initial steps involved executing in-text FRQI Jupyter noteboks provided as part of a an IBM Qiskit Textbook chapter[10]. The simple examples provided in this Qiskit Textbook were executed, but the examples were simplistic and only covered manually created FRQI circuits utilizing rudimentary gate operations in a hard-coded fashion. The greater task was to implement a suitable encoding mechanism that could work for a square image of any size. The next step involved generalizing the examples and defining a function to take an arbitrary, binary image, i.e. black and white pixels, and generate an FRQI quantum circuit that could be executed and then measured to retrieve the original image.

Initially, work was performed on writing some of the circuits and image representations by hand for a small $n$ ($n$=*1*, $n$=*2*) in order to confirm the FRQI circuit creation code devised for this paper. The hand-designed circuits were compared to the generated circuits to verify that the rotations are created correctly within the FRQI circuit. The FRQI function is modified to accept an array of pixel intensities.

For the first part of the function a counting algorithm to increment the pixel positions using bit masking to increment the pixel position was developed. This counting function generates the quantum gates that increment the pixel position and composes most of the quantum circuit. In other words, most of the qubits are used to track the position of the pixel being encoded.

I then pulled in a sample set of data, a collection of simple shapes, but for this first project I wanted to encode a single image.Fig. 1 is of an initial image used for experimental purposes with the pixel positions overlaid on the image for the corresponding plot of qubit measurements. Additional images were utilized as well for comparison.

# 5 Experiment

## 5.1 Development Environment

Much work was performed in setting up a development environment in order to implement the FRQI method. Initial attempts involved working with Google Colab, Cirq and a few other platforms. The completed solution was implemented on an Anaconda and Jupyter stack utilizing a single Python project leveraging Qiskit.

## 5.2 Classical image preparation

Just like standard data preparation in data science workflows, data must be prepared.

### 5.2.1 Image Selection

We start with very basic images. We start with a few shapes. We use circles, triangles, squares and pentagons.

### 5.2.2 Import Images as Grayscale

We import the images using the cv2 library Convert the images to grayscale and they are read in.

### 5.2.3 Resize Images

We resize the original 200-pixel by 200-pixel images to 16-pixel by 16-pixel images.

### 5.2.4 Thresholding

Thresholding is the binarization of an image. In general, we seek to convert a grayscale image to a binary image, where the pixels are either 0 or 255. A simple thresholding example would be selecting a threshold value T, and then setting all pixel intensities less than T to 0, and all pixel values greater than T to 255. Thresholding is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e., below the threshold or above the threshold. In Computer Vision, this technique of thresholding is done on grayscale images. So initially, the image must be converted in grayscale color space.

### 5.2.5 Otsu Thresholding

In Otsu Thresholding, a value of the threshold isn't chosen but is determined automatically. A bimodal image (two distinct image values) is considered. The histogram generated contains two peaks. So, a generic condition would be to choose a threshold value that lies in the middle of both the histogram peak values.

### 5.2.6 Binary Thresholding

Binary Thresholding uses the pixel intensity and if it is greater than the set threshold use 255 (white) otherwise set to 0 (black).

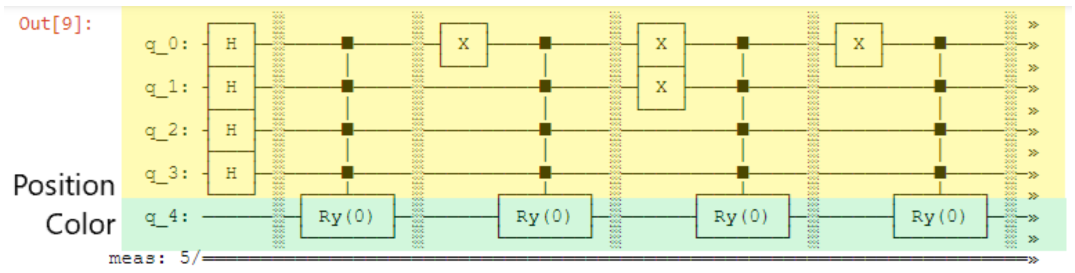

Figure 2: Start of FRQI circuit for 4 x 4 image.

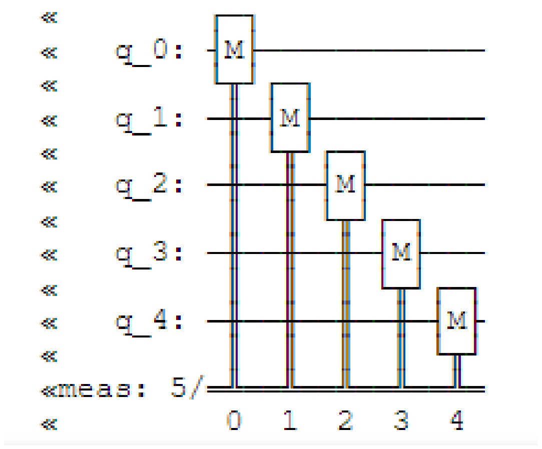

Figure 3: End of FRQI circuit for 4 x 4 image indicating measurement of the qubits which represents image retrieval.

### 5.3 Quantum encoding

We must first put image data into a quantum state. There are various techniques available. Some of the earliest are FRQI Flexible Representation of Quantum Images) and NEQR (Novel Enhanced Quantum Representation). Fig. 2 shows the first few gate combinations required for the first four pixels. The qubits highlighted in yellow represent the position of the pixel and the last quibit is used to encode the color of the pixel.

### 5.4 Quantum image retrieval

Fig. 3 shows the measurement portion of the FRQI circuit which represents image retrieval.

## 6 Results

I was able to ultimately duplicate results discussed in the Qiskit Textbook section covering FRQI circuit execution.[10]

### 6.1 Basic Simulator

Fig. 4 is a representation of an image using an IBM Simulator, specifically the aer_simulator, that shows that for the test image selected for this paper, the pixels at positions 0000, 0001, 0010, 0011, 0111, 1011, 1101, 1110 and 1111 are preceded by a 0 representing the color black. The pixels at positions 0100, 0101, 0110, 1000, 1001 and 1010 are preceded by a 1 indicating that those pixels are colored white.

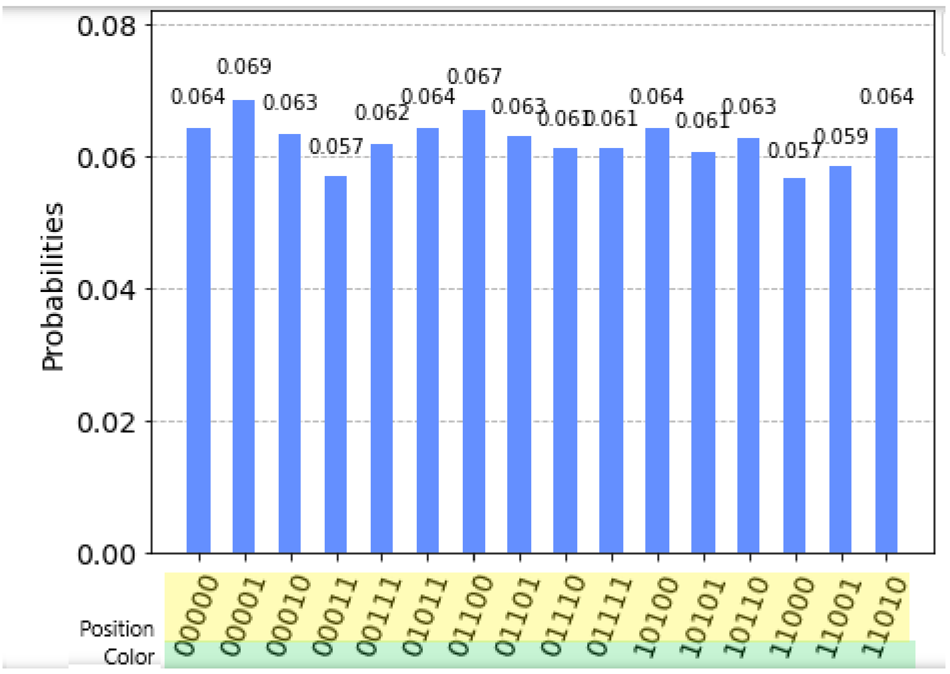

Figure 4: Results quantum image representation at retrieval using an IBM Simulator.

### 6.2 Advanced Simulator

Fig. 5 shows the results of the sample image retrieval on Fake Athens. In this case, all colors (0 and 1) are represented for all pixels which is not a desired results. This is due to the configuration of the Fake Athens simulator which more closely resembles a real quantum device. To get closer to what would actually be run on a real device by feeding the transpiler with a device coupling map (for instance, Athens). The optimization level is set to 3.

## 7 Future Work

- Execute the algorithm developed for this paper against a true quantum computer.
- Implement a compression function to group same colored pixels to reduce the number of gates and gate depth.
- Utilize some novel circuit design in an attempt to further the number of gates and gate depth.
- Utilize a quantum image representation technique and then to perform some quantum image processing that could be shown to be better than current classical methods.
- Implement simple quantum image processing experiments on real quantum computers versus simulators

## Acknowledgement

Many thanks to Dr. Abu Yakaryilmaz and QWorld for all the support.

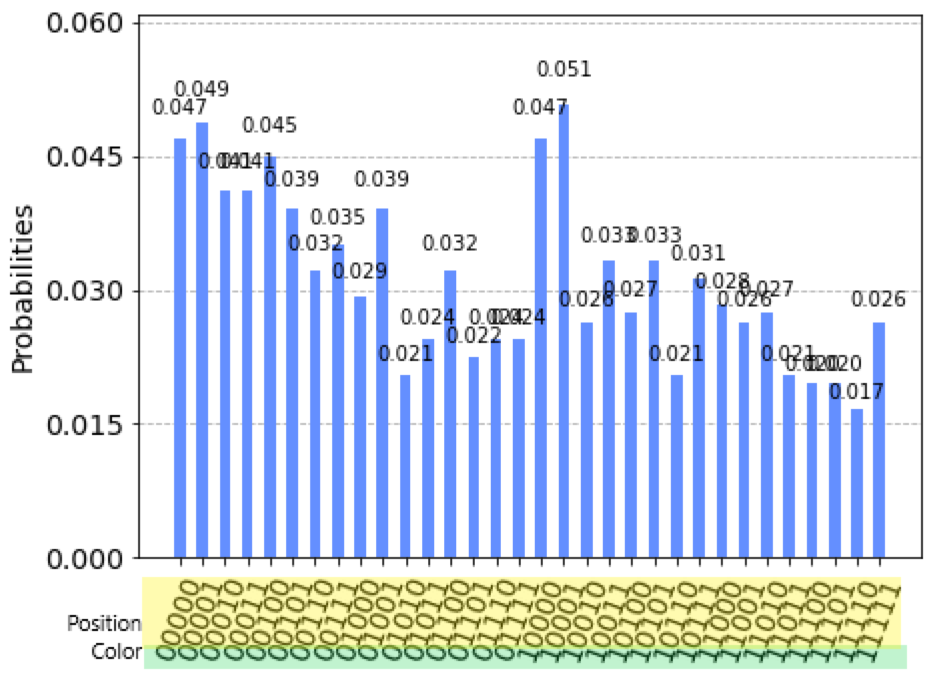

Figure 5: Results quantum image representation at retrieval using the Fake Athens IBM Simulator which more closely resembles a real quantum computer. These measurements indicate very noisy results.

## References

[1] Salvador E. Andraca and Fei Yan. *Quantum Image Processing*. Springer, 2020.

[2] Lucia Garcia. *citiesatnight*. https://github.com/Shedka/citiesatnight. 2019.

[3] Matt Harding. *Quantum-Image-Processor*. https://github.com/mharding15/Quantum-Image-Processor. 2019.

[4] Matt Harding and Aman Geetey. *Representation of Quantum Images*. https://www.cs.umd.edu/class/fall2018/cmsc657/projects/group_6.pdf. Project report of Course CMSC657 , Fall 2018, University of Maryland. Dec. 2018.

[5] Marina Lisnichenko and Stanislav Protasov. "Quantum image representation: a review". Feb. 2022.

[6] Robert Loredo and Mehdi Bozzo-Rey. *How To Start Experimenting With Quantum Image Processing*. 2021. URL: https://medium.com/qiskit/how-to-start-experimenting-with-quantum-image-processing-283dddcc6ba0.

[7] Nourhan Nasr. *Quantum Image Processing*. 2022. URL: https://www.youtube.com/watch?v=USxaZI1XPjg&t=476s.

[8] Phuc Le Quang et al. "Flexible Representation of Quantum Images and Its Computational Complexity Analysis". In: *Proceedings of the Fuzzy System Symposium* 25 (2009), pp. 185–185. DOI: 10.14864/fss.25.0.185.0.

[9] Jie Su et al. "An improved novel quantum image representation and its experimental test on IBM quantum experience". In: *Scientific Reports* 11.13879 (2021).

[10] The Qiskit Team. *Quantum Image Processing - FRQI and NEQR Image Representations*. 2021. URL: https://qiskit.org/textbook/ch-applications/image-processing-frqi-neqr.html.