Quantum Edge Detection
Dylan Staples and Alex Mao

Abstract

Edge detection is important in data compression and image recognition. Classical methods have existed for some time[1] while quantum algorithms are now starting to be used as they can offer increased speed and accuracy. We have compiled code that takes an image and uses IBM's quantum computers and Qiskit to run a Quantum Hadamard Edge Detection[2]. The code takes a classical image in the form of a matrix, translates it into a state vector using log_2(N) + 1 qubits (where N is the number of pixels), and computes the change of intensity using a Hanamard gate[2]. For low-resolution images (32x32 pixels) the algorithm was largely successful in identifying edges. Two operations need to be done, one for the horizontal edges and one for the vertical which can be done by simply transposing the image. The resulting plot although accurate to the edge outline suffers from a low-resolution display method. If another Quantum edge detection project was attempted more hardware should be allocated to test for the edge detection speed of higher resolution images and compare to classical algorithms.

Objectives

Our objective is to use Quantum algorithms for edge detection of simple images and to make a more accurate and faster algorithm than classical methods for edge detection.

Background

The quantum and classical versions of edge detection use similar mathematical means[1] to determine where an edge is by comparing the intensity of adjacent pixels using derivatives[2]. If the value of the derivative is large, that signifies an abrupt change from 1 pixel to another constituting an edge[1,2]. To compute the derivatives, images are convoluted with a filter. The filters used are gradient operators and a specific one is the Prewitt Operator[2]. Both the first and second derivatives of adjacent pixels are useful. The 1st derivative gets the actual intensity changes and the 2nd derivative determines where the maximum value is when the 2nd derivative equals 0. There are a few methods to compute the derivatives but the most standard is the Canny Edge Detector which uses an operator that is the first derivative of a Gaussian function[1]. Problems with classical edge detection methods include noise, determining the threshold that constitutes edges, and sometimes needing help to identify a true edge[1]. Some issues of noise can be mitigated by smoothing the image, but since the 2nd derivative is important in the calculation even a small amount of noise will be prominent[1].

Method

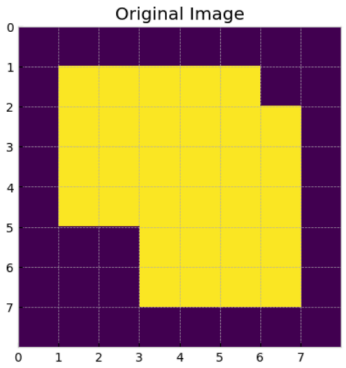
For the code, we started by creating a simple image as a matrix that was effectively black and white. The quantum circuit part of the code should be compatible with much larger images and color images as long as the intensity data of each pixel is known[2]. After a more complex image is uploaded, all that needs changing is the number of qubits used and the dimensions of the image in pixels. The next step is to take the intensity of each pixel and convert it into a normalized state vector[2]. This is done for both the horizontal and vertical axes by taking the intensity value of each pixel and dividing it by the summed intensity of all pixels. The image state vectors are initiated in the quantum circuit using 6 qubits[2]. One auxiliary qubit is used in operation to enable the calculation of both the even and odd pixel differences effectively saving a second iteration of the circuit. To accomplish the simultaneous calculations the auxiliary qubit is acted on with a Hanamard gate which duplicates the coefficients so that c0 is the 1st and 2nd entry of the image state, c1 is the 3rd and 4th, and so on. All qubits are then acted on with a shifting matrix[2]

$$M_{\text{shift}} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

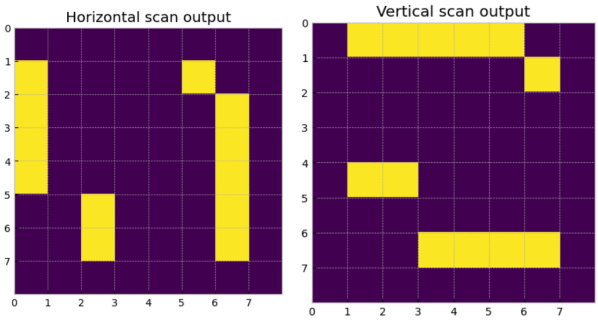
Effectively the circuit computes the differences of the even pixels then shifts the image within the circuitry and computes the differences of the odds. This is then measured on the auxiliary qubit in state |1> and |0> returns the gradient values for horizontal and vertical respectively[2].

Results:

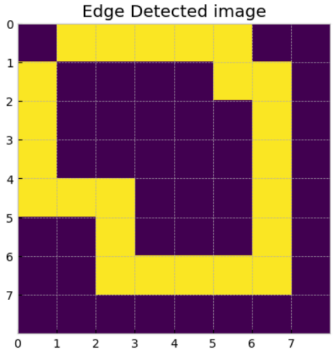
The algorithm was first done with the simple case using the following image:



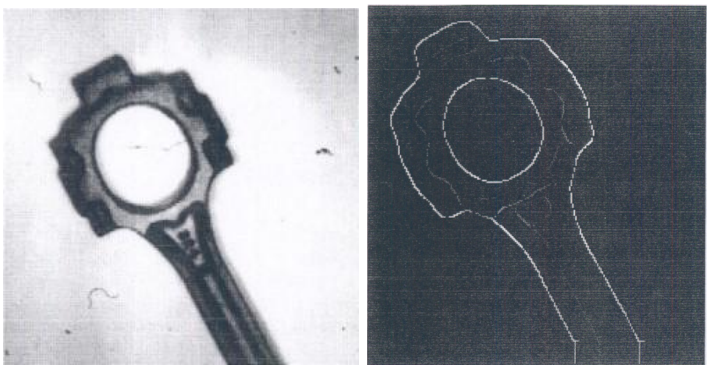
Using the quantum qubit method of normalizing the intensity data and obtaining the horizontal and vertical scan of the image resulted in the following set of images.



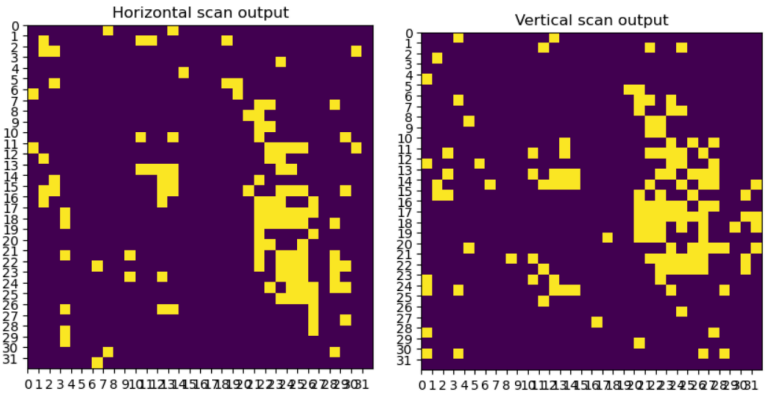
The combination of the two results in the following image:



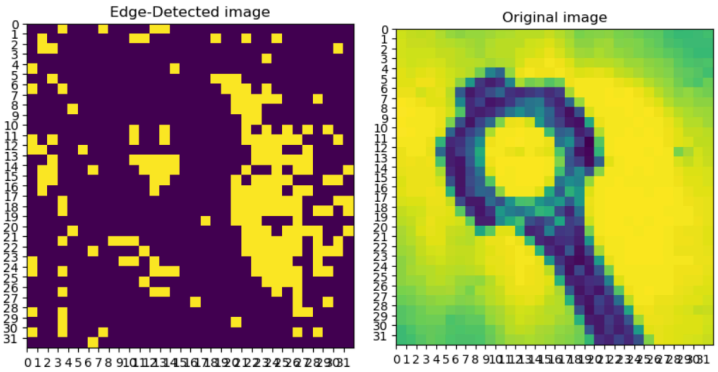
For the bigger image case, we used an image that had also undergone classical edge detection[4]. The image and their results are below.



Our original image[4] was much larger than the previously analyzed one, (32x32), we attempted to use 256x256 resolution for the analysis but the limited computer memory would not allow us to use that many qubits(18). After applying our algorithm we got the following results for the horizontal and vertical edge detection:



Combining these two results and comparing the resulting image to the original image we have:



Our results used the simulator state vector resource by IBM. We attempted using the actual quantum computer but it kept returning errors. We believe the reason is that there may have been an issue running the code as according to the website it only has 5 qubits while our code already uses 7 qubits for just the small image. In addition to this, many times we tried to accommodate for 5 qubits in our code resulted in nonsense images for the edge detection. Despite this, if we did have access to enough qubits to run our code we would expect the general shape of our image to be shown but with many inconsistencies between each run due to the noise of a real quantum computer.

Discussion:

The experimental results showed an effective quantum edge detection simulation result for the image provided. Despite the 32x32 size limitation, the outline of the original image is visible enough to the human eye to understand its general shape. Starting this project with the coding

seemed fairly difficult with debugging and making sure the code used the correct version of Quiskit. Our code also did not seem to work effectively with the real quantum computer since our results required more than only 5 qubits and did not return any results. Our method using the quantum algorithm had the main limitation of hardware which we mentioned as not having enough hardware computer memory to use more than 15 qubits to render a 256x256 resolution image and not enough qubits for the real quantum computer. For future work, we could increase our hardware allocation or hardware memory itself with more computing power or qubits to allow for more accurate edge detection.

Conclusion:

This project successfully demonstrated the ability to use quantum computing and quantum algorithms to detect image edges and create an outline for a 32x32 image. The algorithm converts all of the pixels into a normalized vector space and uses the Hanamard gate to calculate the relative changes in intensity. The changes in intensity were first detected horizontally and then vertically then combined to result in a complete edge-detected image. Although our edge detection of the original image was not perfect, the rough outline of what is there can be clearly shown. The simple image results were shown earlier and the outline results were fairly strong and clear. These errors in image edge detection could be attributed to the low memory count and can be alleviated in the future with higher computer memory to access larger qubit amounts for higher-resolution images.

References

[1] - Classical Edge Detection (link)

[2] - A Quantum Edge Detection Algorithm - Giacomo Cavalieri, Dario Maio (link)

[3] - GitHub Code 1: roysuman088/quantum-edge-detection.ipynb (link), 2: Quantum-Hadamard-Edge-detection/QC_QHED.ipynb (link)

[4] - Machine Vision, 3rd Edition by E. R. Davies (link)

Python Code for Canny Edge Detection (link)