Quantum Image Processing: Edge Detection

Using Quantum Probability Image Encoding (QPIE) and Quantum Hadamard Edge Detection (QHED)

Roman Tudor-George

June 4, 2025

Outline

1. Project Description
2. Related Work and Research
3. Quantum Probability Image Encoding (QPIE)
4. Quantum Hadamard Edge Detection (QHED)
5. General Approach & Architecture
6. Implementation Details
7. User Manual
8. Demonstration of Results
9. Conclusion

Project Description (Part 1: Overview)

- **What it is**: This project explores a quantum algorithm for performing edge detection on classical digital images.
- **Core Idea**: Leverage quantum mechanics principles, specifically superposition and quantum operations, to identify sharp changes in pixel intensities.
- **Key Terms Defined**:
  - **Edge Detection**: Identifying boundaries within an image where brightness changes significantly.
  - **QPIE (Quantum Probability Image Encoding)**: A method to represent pixel values as probability amplitudes of a quantum state.
  - **QHED (Quantum Hadamard Edge Detection)**: An algorithm using Hadamard gates to compute differences between neighboring pixel data encoded in a quantum state.

Project Description (Part 2: Scope & I/O)

- **Input**: A classical digital image (typically grayscale).
- **Output**: A binary image highlighting the detected edges.
- **Input Constraints**:
  - Images are typically normalized.
  - Due to current simulator/hardware limits, large images are processed by splitting them into smaller blocks (e.g., 32x32 pixels).

Related Work and Research

- **QHED (This Project's Basis)**: Yao et al. (2017) introduced Quantum Hadamard Edge Detection using QPIE. Achieves $O(1)$ complexity for edge detection (excluding state preparation) [1].
- **QSobel**: Zhang et al. (2015) proposed QSobel, which utilizes FRQI encoding. It has an $O(n^2)$ complexity and generally requires more qubits and more complex operations compared to QHED [2].
- **QFT-based Methods**: Other approaches apply the Quantum Fourier Transform (QFT) for edge detection by analyzing image features in the frequency domain. Qubit efficiency and complexity vary depending on the specific QFT-based algorithm and image representation used [3].

---

[1] Yao, X.W., et al. (2017). Quantum image processing and its application to edge detection. *Physical Review X, 7*(3), 031041. [arXiv:1801.01465]

[2] Zhang, Y., Lu, K., Gao, Y. (2015). QSobel: A novel quantum image edge extraction algorithm. *Science China Information Sciences, 58*(1), 1-13. [Springer]

[3] Yan, F., Iliyasu, A.M., Venegas-Andraca, S.E. (2016). A survey of quantum image representations. *Quantum Information Processing, 15*(1), 1-35. [Springer (Survey)]

QPIE: Concept & Example

The QPIE representation uses the probability amplitudes of a quantum state to store pixel values of a classical image. Quantum amplitude encoding loads classical data into a quantum state by using the data values as amplitudes of a superposition state.

For example, with a 2×2 image ($N = 4$ pixels, requiring $n = 2$ qubits), the quantum state is:

$$|\psi\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle$$

QPIE: Coefficients & Generalization

Each coefficient c_i corresponds to a pixel intensity after normalization, calculated as:

$$c_i = \frac{I_{yx}}{\sqrt{\sum I_{yx}^2}}$$

Here, I_{yx} is the pixel intensity at position (y, x), and the denominator normalizes the quantum state such that $\sum |c_i|^2 = 1$.

Generalizing for_combo for n-qubits (encoding 2^n pixel values):

$$|Img\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle$$

- $|Img\rangle$: Quantum state representing the image.
- n: Number of qubits.
- c_i: Normalized complex amplitudes (derived from pixel values).
- $|i\rangle$: Computational basis state.

Quantum Hadamard Edge Detection (QHED) - Concept

The Hadamard gate (H) acts on a qubit as:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

It creates an equal superposition. For edge detection, it helps compute differences.

Applying H to the first (ancilla) qubit results in the superposition:

$$\frac{1}{\sqrt{2}} \left(|b_0 b_1 b_2 \ldots 0\rangle + |b_0 b_1 b_2 \ldots 1\rangle \right)$$

Quantum Hadamard Edge Detection (QHED) - Transformation

Consider two neighboring pixels encoded in states differing by the last qubit, e.g., $|b_{n-1} \dots b_1 0\rangle$ and $|b_{n-1} \dots b_1 1\rangle$. Applying H to the last qubit (via $I_{2^{n-1}} \otimes H_0$) transforms the amplitudes c_k, c_{k+1} associated with these states:

$$c_k|b \dots 0\rangle + c_{k+1}|b \dots 1\rangle \xrightarrow{I \otimes H} \frac{c_k + c_{k+1}}{\sqrt{2}}|b \dots 0\rangle + \frac{c_k - c_{k+1}}{\sqrt{2}}|b \dots 1\rangle$$

The amplitude $c_k - c_{k+1}$ highlights the gradient (edge).

QHED - Matrix Representation

The operation $I_{2^{n-1}} \otimes H_0$ (Hadamard on the 0^{th} qubit) is:

$$I_{2^{n-1}} \otimes H_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 \end{pmatrix}$$

Applied to a state vector $(c_0, c_1, \dots, c_{N-1})^T$:

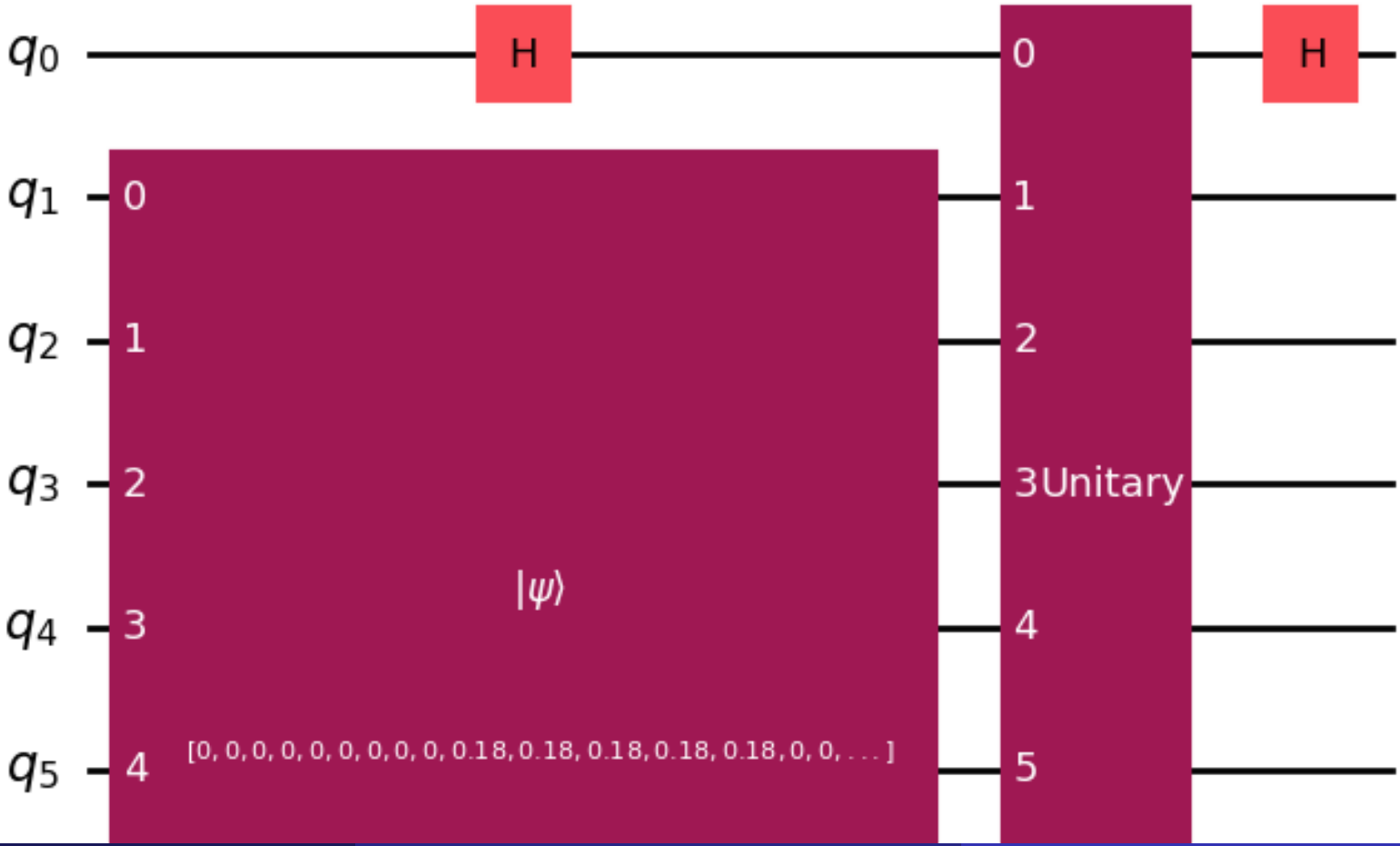
$$\rightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{pmatrix}$$

Pre-processing & Image Handling

- **Gaussian Blur**:
  - A low-pass filter applied to the input image.
  - Smooths the image, reducing noise and minor details.
  - Helps prevent false edge detection from insignificant intensity variations.
- **Image Splitting**:
  - Large images are divided into smaller, manageable parts (e.g., 32x32 pixels).
  - This is crucial for simulation on classical computers due to the exponential growth of the statevector space with the number of qubits (e.g., a 32x32 image part requires 10 qubits for $2^{10} = 1024$ pixels).
  - Each part is processed independently.
- **Horizontal and Vertical Scans**:
  - Edge detection is performed separately for horizontal and vertical directions.
  - For vertical edges, the image part is processed as is.
  - For horizontal edges, the transpose of the image part ($img\_part.T$) is used before QPIE.

Quantum Circuit for Edge Detection

The following diagram illustrates the core quantum circuit structure used (example for fewer qubits than 10 for visual simplicity, but the principle scales):



Simplified QHED Algorithm Flow

Algorithm Steps:

1. Apply Gaussian blur to the input image.
2. Divide the image into 32x32 pixel blocks.
3. For each block:
    1. *Horizontal Scan*: Encode block into quantum state ($|\psi_H\rangle$) using QPIE.
    2. Apply QHED circuit: Hadamard on ancilla, cyclic shift, Hadamard on ancilla.
    3. Simulate to get statevector and extract horizontal edges.
    4. *Vertical Scan*: Encode block transpose ($|\psi_V\rangle$) and repeat QHED circuit.
    5. Simulate to extract vertical edges.
    6. Combine horizontal and vertical edges (e.g., bitwise AND).
4. Reconstruct full edge map from all blocks.

Edge Detection:

- Check statevector amplitudes (e.g., c_{2i+1}) for differences.
- Apply threshold (e.g., $|\text{amplitude}| > e^{-15}$) to identify edges.

Post-Simulation Processing

Simulation:

- Uses statevector simulator to compute the quantum state directly.
- State evolves via unitary operators: $|\psi_{out}\rangle = U|\psi_{in}\rangle$.

Edge Map Construction:

- Collect horizontal and vertical edge data from all 32x32 blocks.
- Combine edge information to form a unified edge map.
- Apply post-processing (e.g., noise reduction or edge thinning) if needed.

Output:

- Generate a visual representation of the edge map.
- Edges highlight significant intensity changes in the original image.

User Manual: Running the Project

To build and use this project:

1. **Prerequisites**:
   - Python (3.8+ recommended).
   - Qiskit: ‘pip install qiskit qiskit-aer‘
   - NumPy: ‘pip install numpy‘
   - Image manipulation library (e.g., OpenCV or Pillow): ‘pip install opencv-python‘ or ‘pip install Pillow‘
2. **Input Image**:
   - Prepare a digital image (e.g., ‘.png‘, ‘.jpg‘). Grayscale images are typically used, or color images are converted to grayscale.
3. **Code Structure (Assumed)**:
   - A main Python script (‘.py‘) containing the functions for:
     - Image loading and pre-processing (Gaussian blur, splitting).
     - ‘amplitude$_e$*ncode*1()‘*function*.*Quantumcircuitconstruction*(*asshown*).
- Simulation and post-processing.
- Image reconstruction and display/saving.

4. **Execution**:
   - Run the Python script.

Demonstration of Results

The algorithm was tested using the Qiskit Aer statevector simulator.

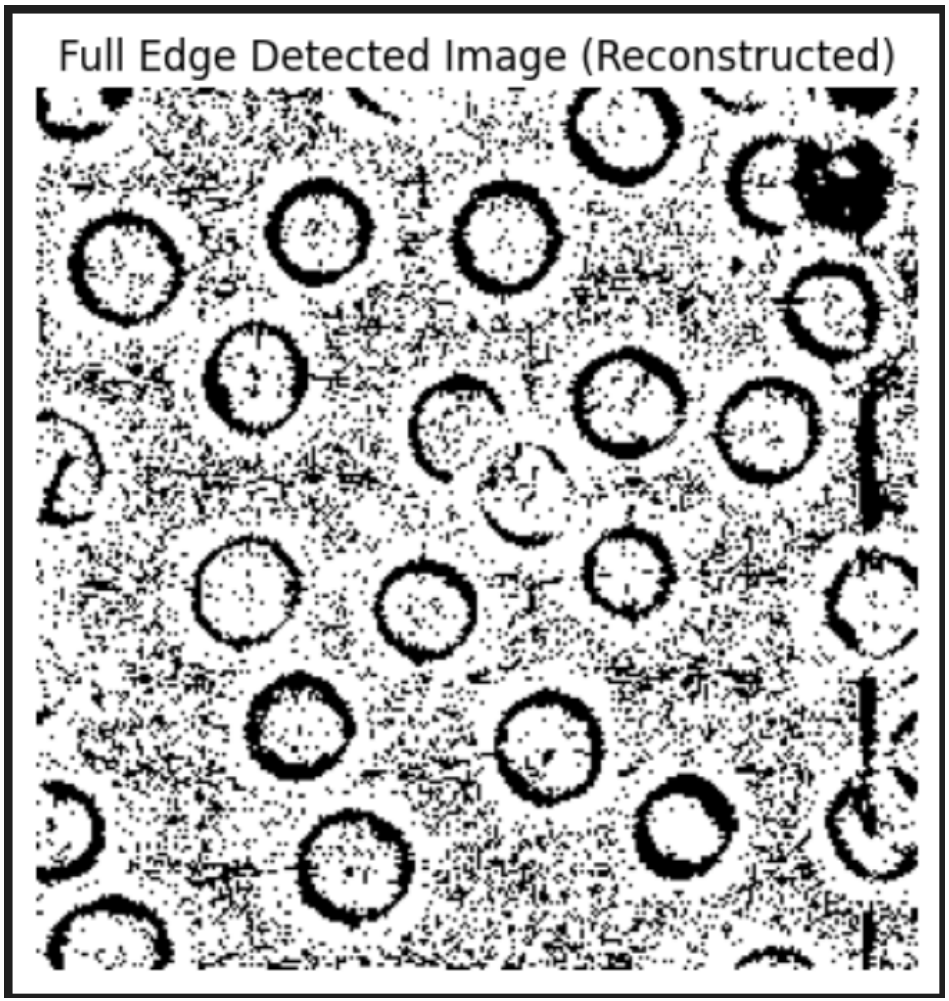


Figure: Input Image

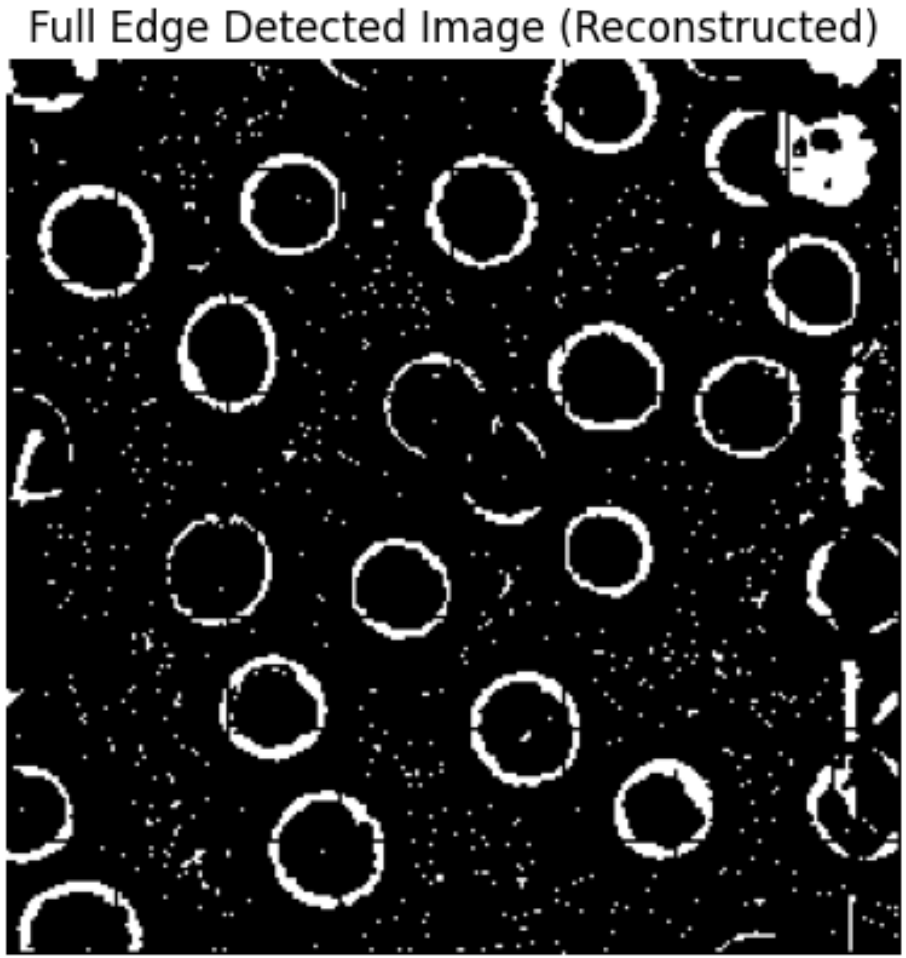


Figure: Final Edge Map

Conclusion Outlook

- **Project Recap**: We explored quantum edge detection using Quantum Probability Image Encoding (QPIE) for compact representation and the Quantum Hadamard Edge Detection (QHED) algorithm.
- **Quantum Advantage Potential**:
  - QPIE enables storing N-pixel images with only $\log_2 N$ qubits, offering significant data compression.
  - Quantum algorithms, like QHED, aim to leverage quantum parallelism for efficient image feature extraction, promising speedups for large images compared to pixel-wise classical methods, as explored in QImP research.
- **Important Note on Speedup**: Any classical pre-processing steps (e.g., image filtering, splitting before encoding) are performed on classical computers and do not contribute to quantum speedup. The potential advantages lie within the quantum processing stages.
- **Outlook**:
  - This work demonstrates the principles of quantum edge detection

Thank You!

Questions?