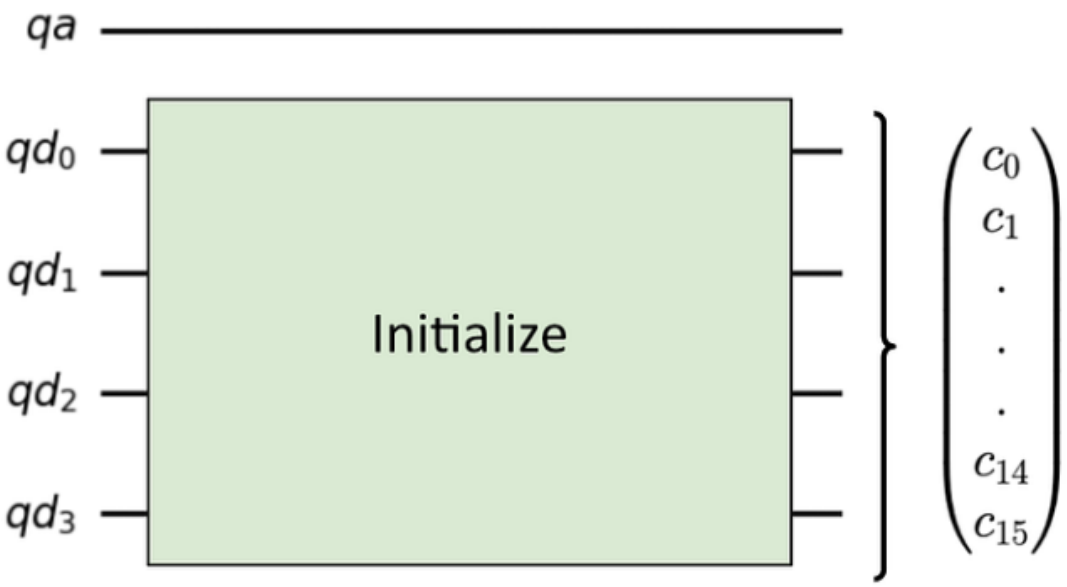
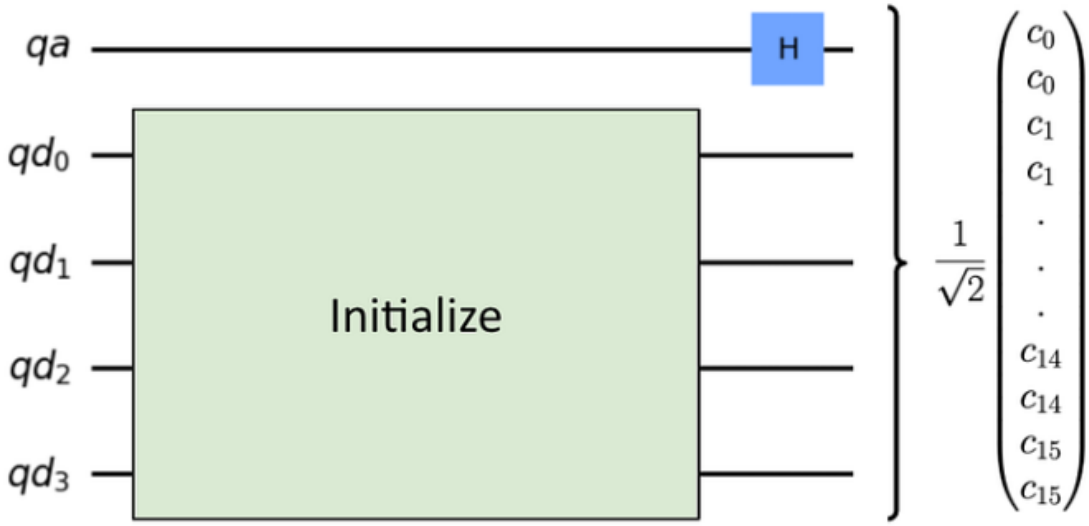
- **Edge Detection** is an essential method to identify components of an image. As its name implies, edge detection is the process of detecting edges of components to distinguish them by their perimeter.
- Normal classical image processing can perform edge detection, but it can sometimes be very inefficient for large images with an abundance of pixel computations.
- Quantum image processing is an efficient alternative with its own method of detecting edges of images.
- In order to perform quantum image processing on images used with classical image processing, we need to convert them to quantum images.
- There are various types of quantum image representation, one being **Quantum Probability Image Encoding (QPIE).**
- One notable advantage of QPIE is that it requires substantially less amount of qubits than pixels in classical images.
- If we have an N-pixel image, we only need $\log_2(N)$ qubits. For every n qubit, we also have the ability to create 2^n states with superposition.
- **Quantum Hadamard Edge Detection (QHED) -**Classically, to determine the edge of an image, we need to determine the pixel-intensity gradients.
- This requires processing each pixel, which leads to a complexity of O(N) for an image of N pixels. Since pixel number grows exponentially with image size, this poses a problem for large images.
- With a quantum algorithm, we can determine the pixel-intensity gradients with complexity O(1), regardless of the image size, by taking advantage of the superposition induced by the **Hadamard gate.**
- This is an exponential speedup. Furthermore, we only need n qubits for an image with 2^n pixels.
- Let's walk through the QHED algorithm for a 4x4=16 pixel image. For this, we will need $\log_2(16) = 4$ data qubits and 1 ancilla qubit. We can encode each pixel's intensity into an amplitude c_i.
- We start by initializing our data qubits to $\sum[i=0 \text{ to } N-1] c_i |i\rangle = (c\_0,c\_1,\ldots,c\_(N-1))^T$ and our ancilla qubit to $|0\rangle$. So our overall state is: $(c\_0,c\_1,\ldots,c\_(N-1))^T \otimes |0\rangle$.

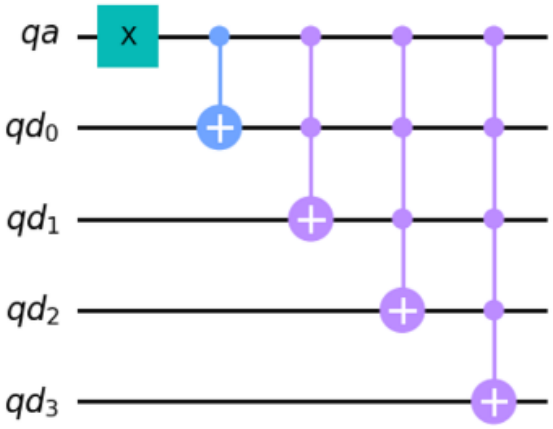
- Then we apply a Hadamard gate to our ancilla qubit. So our overall state becomes (c_0,c_1,…,c_(N-1))^T ⊗ (|0⟩ + |1⟩)/√2. Since the overall number of binary states is doubled, this duplicates each amplitude c_i.



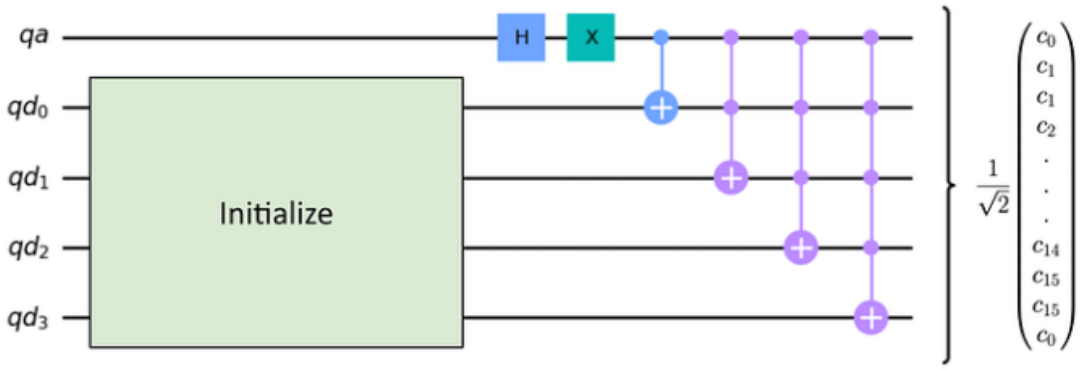


- Now we apply a decrement gate to all the qubits. The function of the decrement gate is to "shift" the assignment of the amplitudes up by one.
- So c_j becomes the amplitude of |j-1⟩. The decrement gate is a unitary operation that is made up of **X-gates and controlled X-gates**. For 5 total qubits, it looks like this

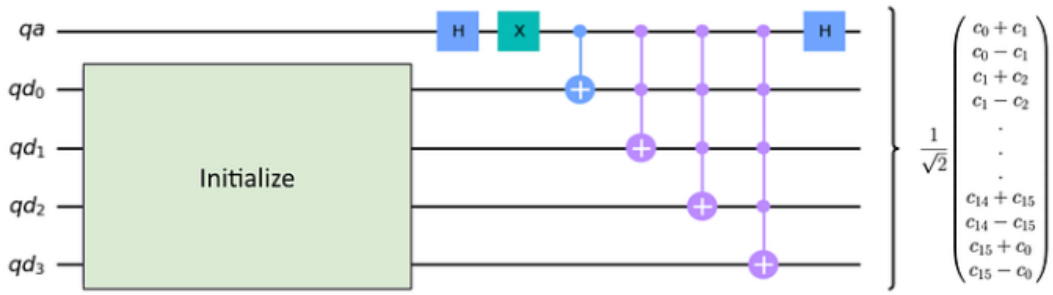
$$D = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$



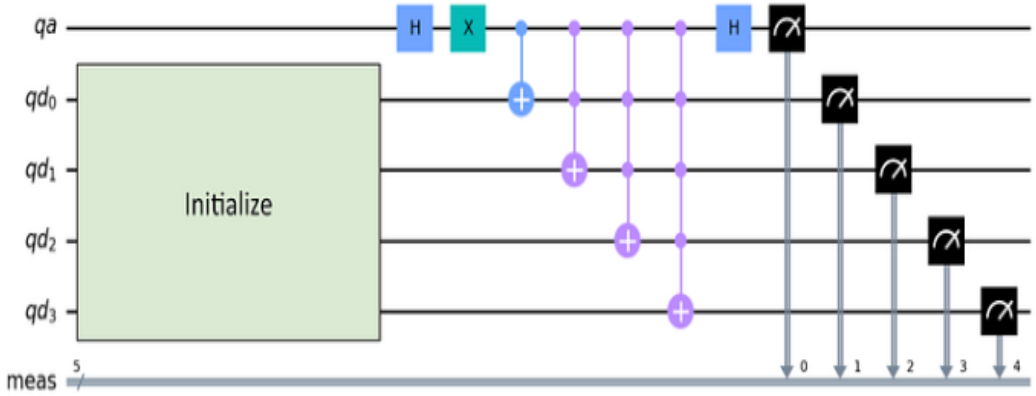
- After applying the decrement gate, we have:



- Finally, we apply a Hadamard to the ancilla qubit. This has the function of adding neighboring amplitudes for even states |j⟩ and subtracting neighboring amplitudes for odd states |j⟩. This is how we encode the gradient information into the wave function.



- Now we can measure all the qubits. We care about the difference between neighboring amplitudes (c_k — c_(k+1)), which is encoded into the amplitudes of the odd states. We are more likely to measure states with larger amplitudes, which corresponds to larger pixel gradients!



- We run this circuit a given number of times to find a distribution of measured states. The odd-numbered states yield the gradient information we want, and we can plot the number of measurements vs. pixel location (|2i+1⟩ corresponds to pixel i).
- To actually do this for a real image, we will need to do a horizontal scan and a vertical scan. The vertical scan is accomplished by transposing the original image and repeating the same steps. In the end, we add up the horizontal and vertical scan measurements to generate our output image.

Horisontal Scan:

1. State preparation ($|\text{Img}\rangle = |01\rangle$) : We can ahieve this with a simple $[X(1)]$ operation.
2. Decrement gate: We can achieve this by a sequence of $[X(0), CX(0,1), CCX(0,1,2)]$ operations.

Vertical Scan:

1. State preparation ($|\text{Img}\rangle = |10\rangle$) : We can ahieve this with a simple $[X(2)]$ operation.
2. Decrement gate: We can achieve this by a sequence of $[X(0), CX(0,1), CCX(0,1,2)]$ operations.

GATES USED IN QUANTUM EDGE DETECTION ARE :-

1. **The Hadamard Gate**
   - The Hadamard gate (H-gate) is a fundamental quantum gate. It allows us to move away from the poles of the Bloch sphere and create a superposition of $|0\rangle$ and $|1\rangle$
   - It has the matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

We can see that this performs the transformations below:

$$H|0\rangle = |+\rangle$$

$$H|1\rangle = |-\rangle$$

2. The X-Gate

The X-gate is represented by the Pauli-X matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

To see the effect a gate has on a qubit, we simply multiply the qubit's statevector by the gate. We can see that the X-gate switches the amplitudes of the states $|0\rangle$ and $|1\rangle$:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

---

3. CXGate

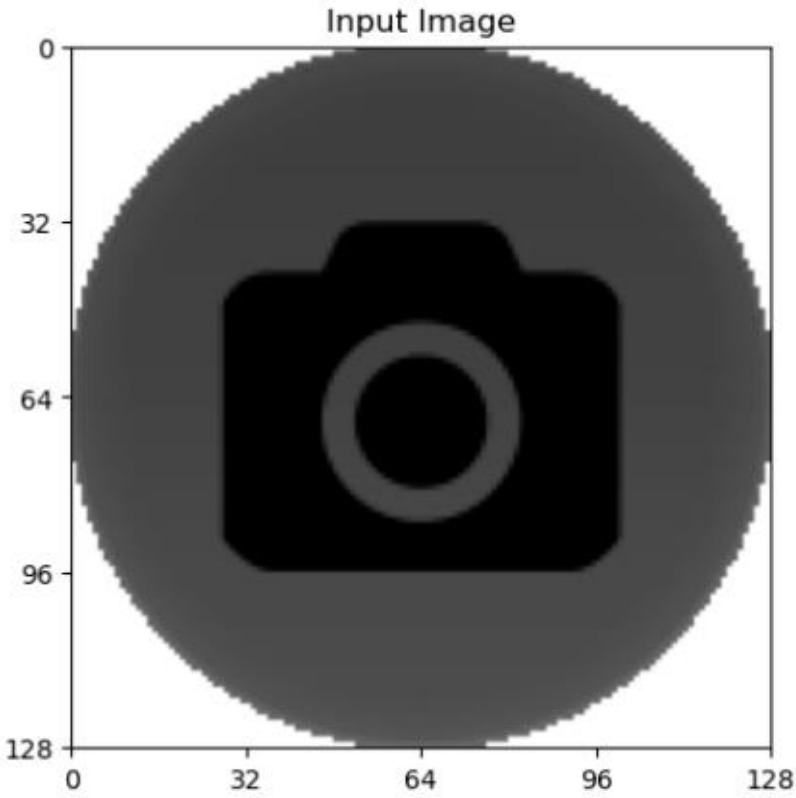
- The CXGate class creates a controlled-X gate, applying the X gate according to the control qubit state.
- The controlled-NOT gate, also known as the controlled-x (CX) gate, acts on a pair of qubits, with one acting as 'control' and the other as 'target'. It performs a NOT on the target whenever the control is in state |1⟩. If the control qubit is in a superposition, this gate creates entanglement.

4. Toffoli gate

- The Toffoli gate, also known as the double controlled-NOT gate (CCX), has two control qubits and one target. It applies a NOT to the target only when both controls are in state |1⟩.
- The Toffoli gate with the Hadamard gate is a universal gate set for quantum computing.

EXAMPLE 1

```
Image shape (numpy array): (128, 128)
```

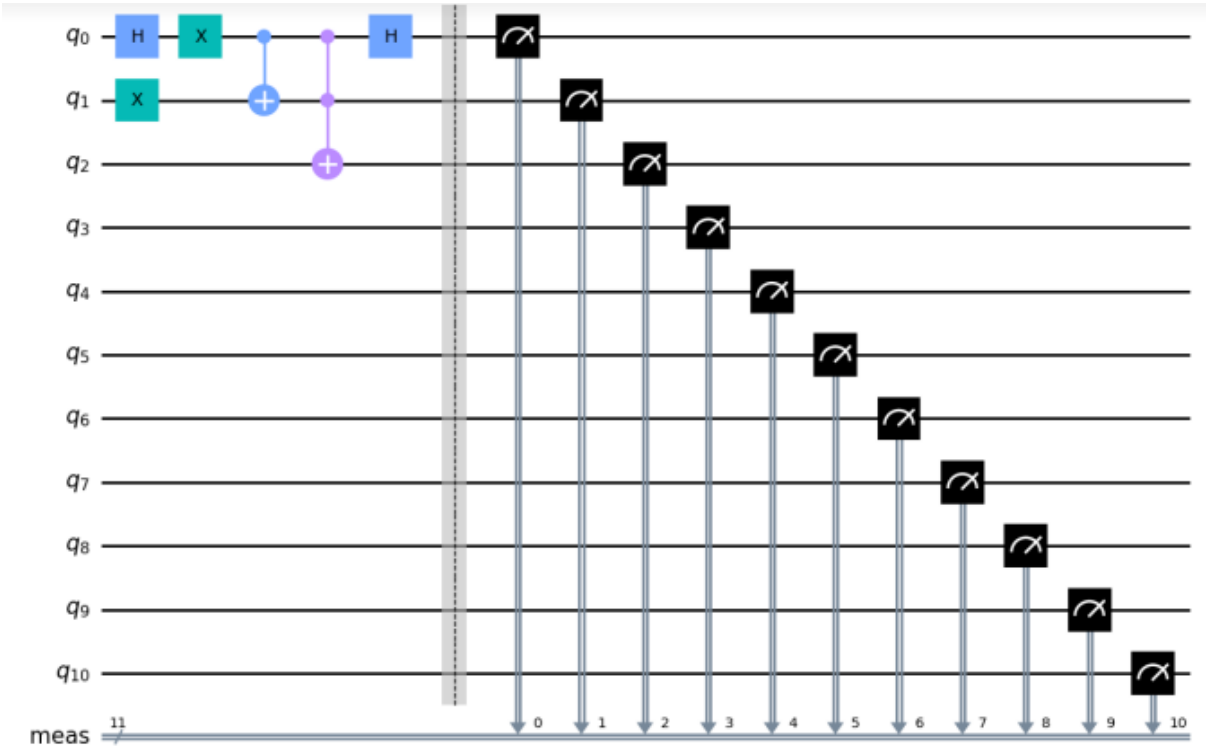


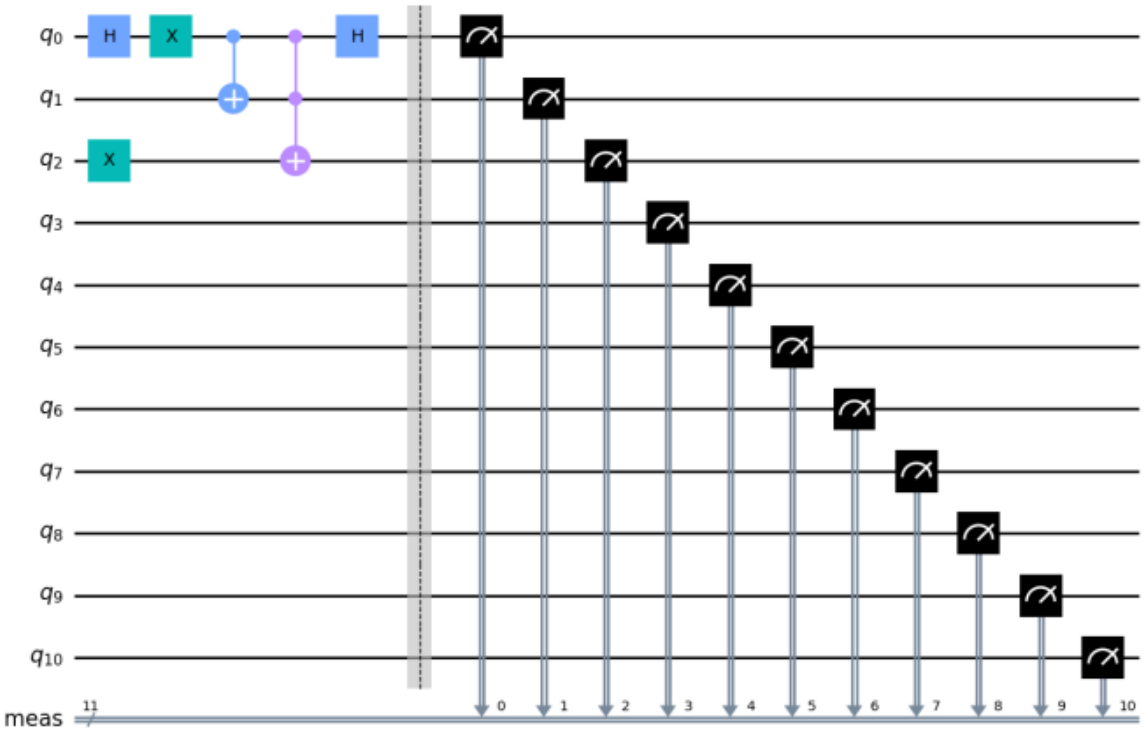
In [52]:

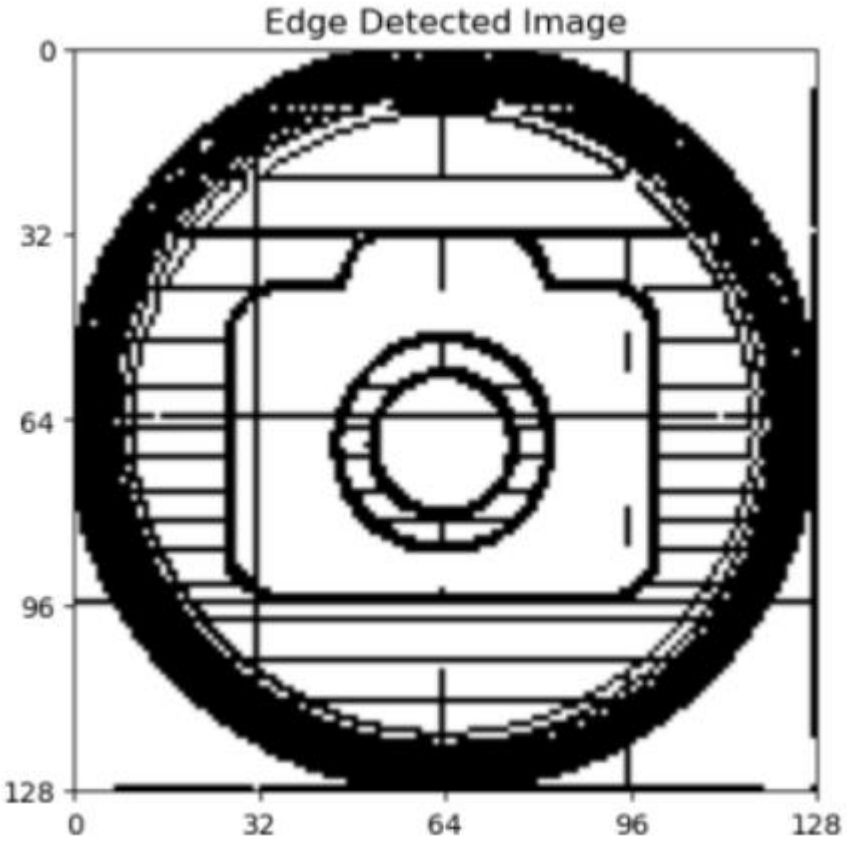
```python
# Initialize some global variable for number of qubits
data_qb = 10
anc_qb = 1
total_qb = data_qb + anc_qb

# Initialize the amplitude permutation unitary
D2n_1 = np.roll(np.identity(2**total_qb), 1, axis=1)
```

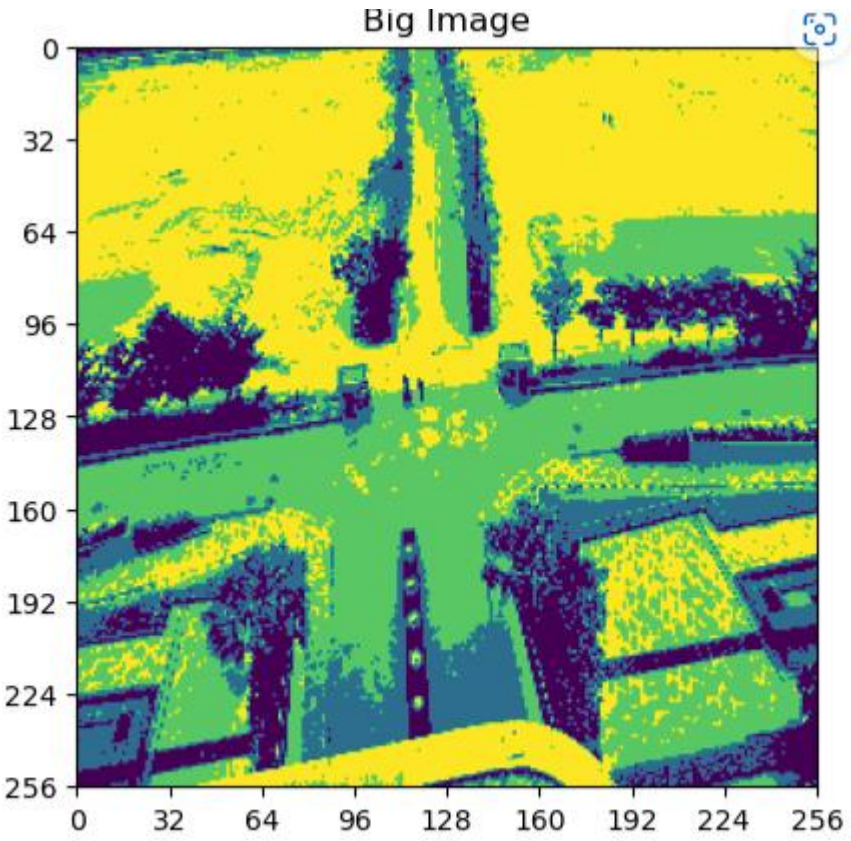
```
# Create the circuit for horizontal scan
qc_small_h = QuantumCircuit(total_qb)
qc_small_h.x(1)
qc_small_h.h(0)
# Decrement gate - START
qc_small_h.x(0)
qc_small_h.cx(0, 1)
qc_small_h.ccx(0, 1, 2)
# Decrement gate - END
qc_small_h.h(0)
qc_small_h.measure_all()
display(qc_small_h.draw('mpl'))
# Create the circuit for vertical scan
qc_small_v = QuantumCircuit(total_qb)
qc_small_v.x(2)
qc_small_v.h(0)
# Decrement gate - START
qc_small_v.x(0)
qc_small_v.cx(0, 1)
qc_small_v.ccx(0, 1, 2)
# Decrement gate - END
qc_small_v.h(0)
qc_small_v.measure_all()
display(qc_small_v.draw('mpl'))
# Combine both circuits into a single list
circ_list = [qc_small_h, qc_small_v]
```




q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
meas
H
X
X
H
11
0
1
2
3
4
5
6
7
8
9
10

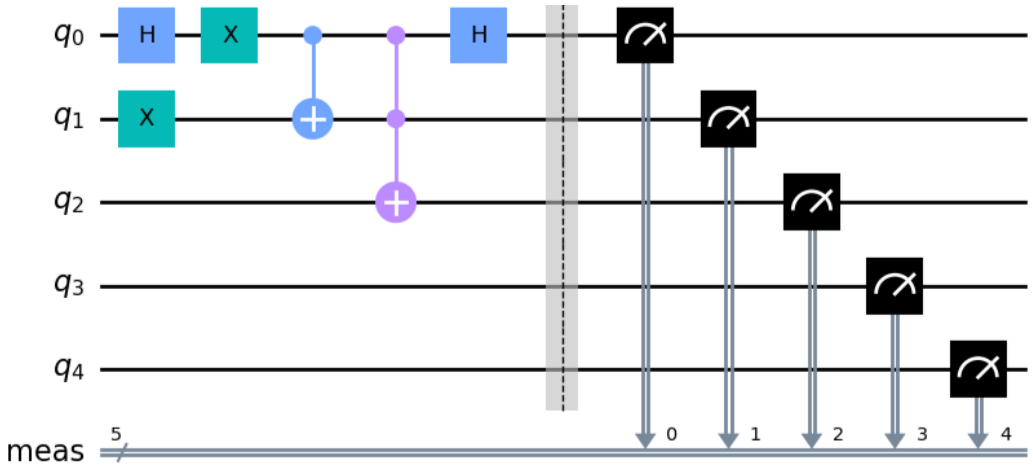

Edge Detected Image
0
32
64
96
128
0
32
64
96
128

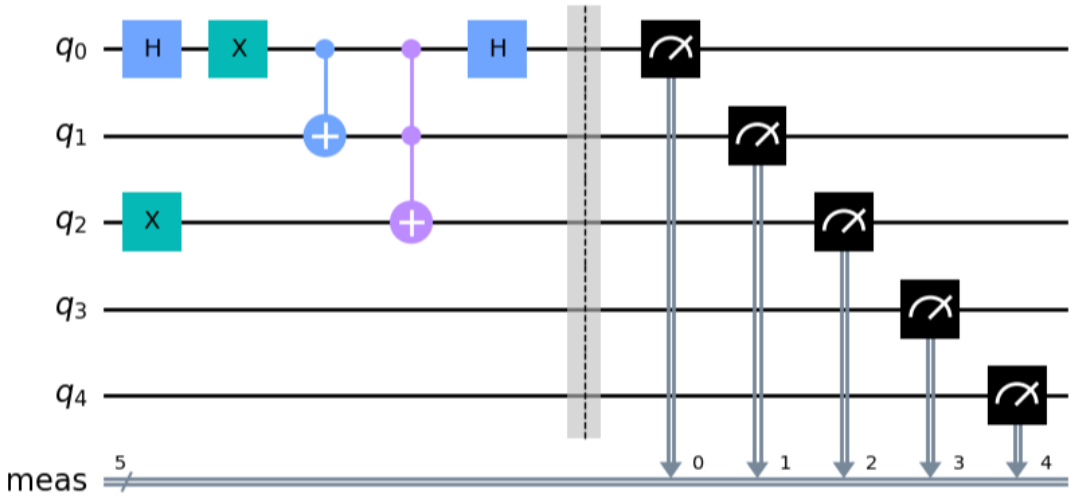
EXAMPLE 2

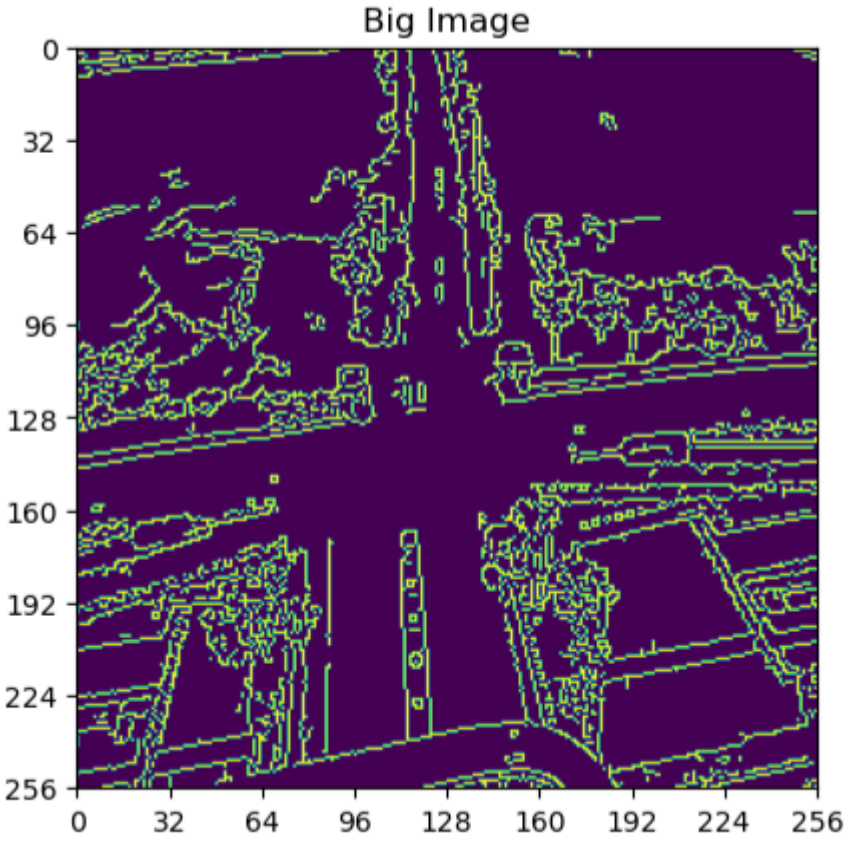


In [29]:
```
# Initialize some global variable for number of qubits
data_qb = 4
anc_qb = 1
total_qb = data_qb + anc_qb
# Initialize the amplitude permutation unitary
D2n_1 = np.roll(np.identity(2**total_qb), 1, axis=1)
```

In [30]:
```
# Create the circuit for horizontal scan
qc_small_h = QuantumCircuit(total_qb)
qc_small_h.x(1)
qc_small_h.h(0)
# Decrement gate - START
qc_small_h.x(0)
qc_small_h.cx(0, 1)
qc_small_h.ccx(0, 1, 2)
# Decrement gate - END
qc_small_h.h(0)
qc_small_h.measure_all()
display(qc_small_h.draw('mpl'))
# Create the circuit for vertical scan
qc_small_v = QuantumCircuit(total_qb)
qc_small_v.x(2)
qc_small_v.h(0)
# Decrement gate - START
qc_small_v.x(0)
qc_small_v.cx(0, 1)
qc_small_v.ccx(0, 1, 2)
# Decrement gate - END
qc_small_v.h(0)
qc_small_v.measure_all()
display(qc_small_v.draw('mpl'))
# Combine both circuits into a single list
circ_list = [qc_small_h, qc_small_v]
```

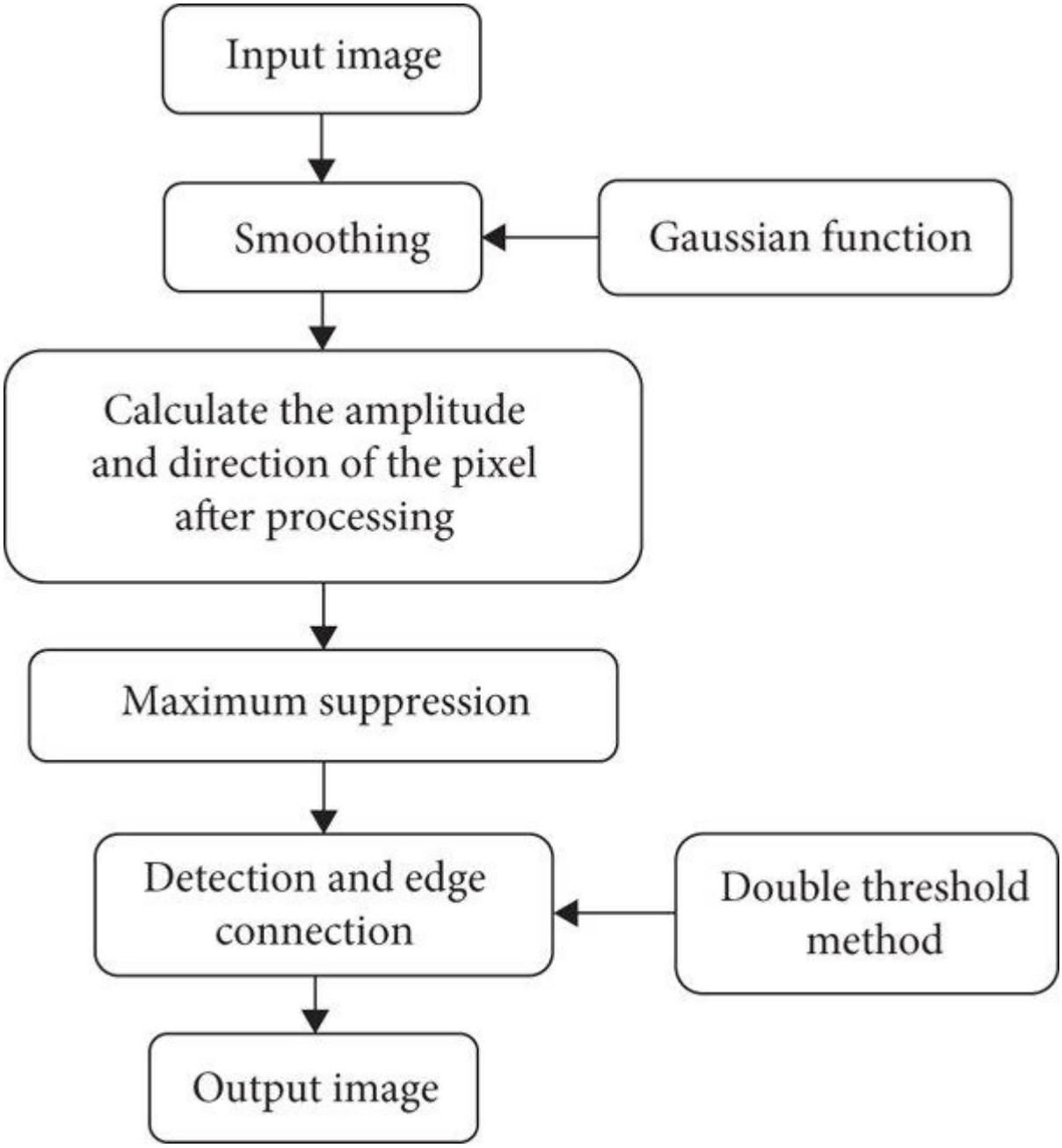




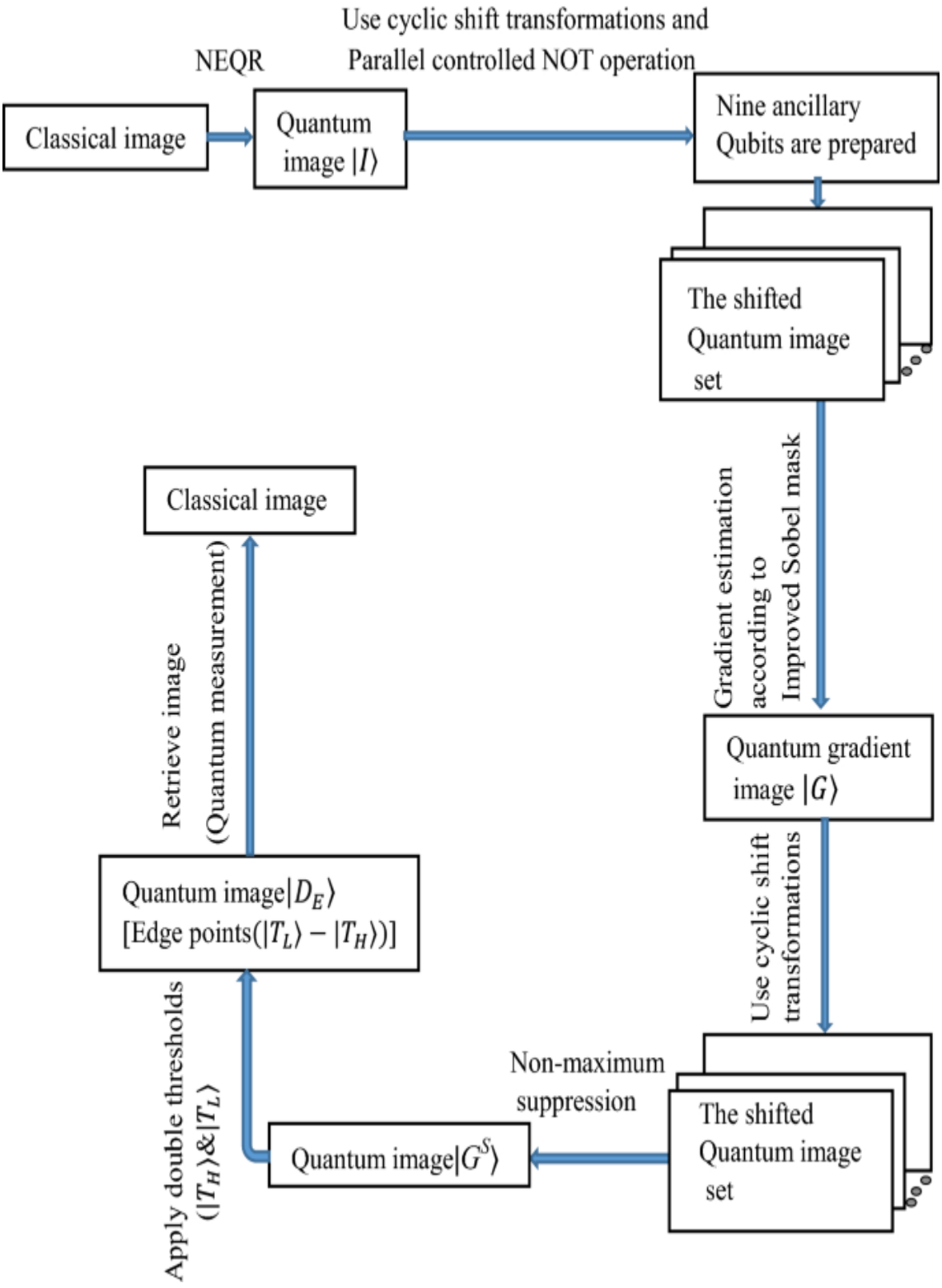


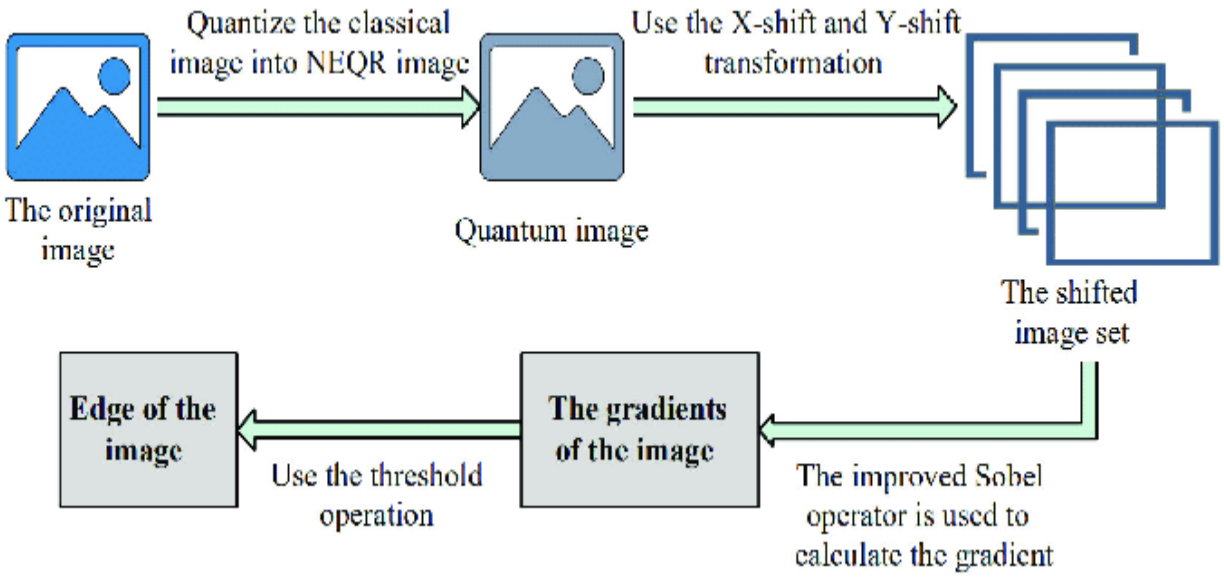
FL0W DIAGRAM

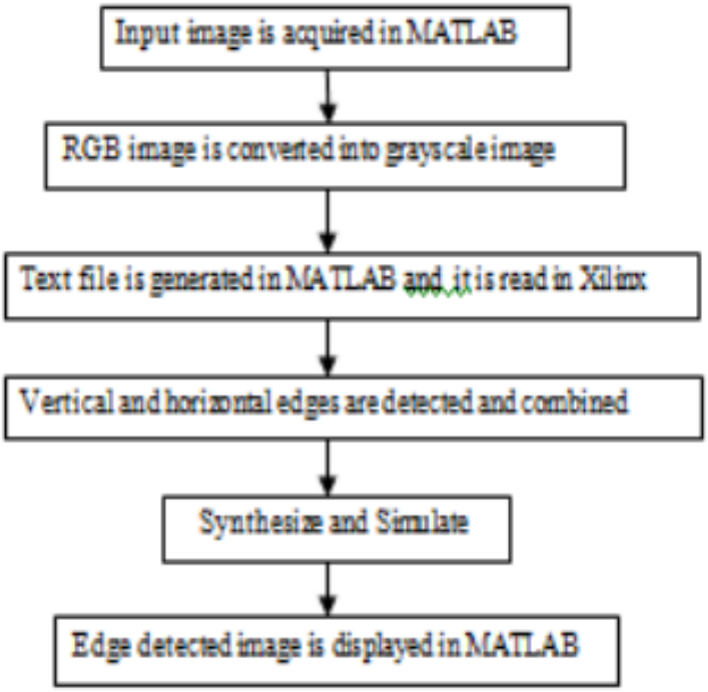
1. TRADITIONAL CANNY EDGE DETECTION



2. QUANTUM EDGE DETECTION


Use cyclic shift transformations and
Parallel controlled NOT operation
NEQR
Classical image
Quantum image $|I\rangle$
Nine ancillary Qubits are prepared
The shifted Quantum image set
Gradient estimation according to Improved Sobel mask
Quantum gradient image $|G\rangle$
Use cyclic shift transformations
The shifted Quantum image set
Non-maximum suppression
Quantum image$|G^S\rangle$
Apply double thresholds $(|T_H\rangle \& |T_L\rangle$
Quantum image$|D_E\rangle$ [Edge points$(|T_L\rangle - |T_H\rangle)$]
Retrieve image (Quantum measurement)
Classical image


Quantize the classical image into NEQR image
Use the X-shift and Y-shift transformation
The original image
Quantum image
The shifted image set
Edge of the image
The gradients of the image
Use the threshold operation
The improved Sobel operator is used to calculate the gradient


Input image is acquired in MATLAB
RGB image is converted into grayscale image
Text file is generated in MATLAB and it is read in Xilinx
Vertical and horizontal edges are detected and combined
Synthesize and Simulate
Edge detected image is displayed in MATLAB