Quantum Edge Detection Demo

Paul Kassebaum

January 21, 2019

1 Quantum Edge Detection

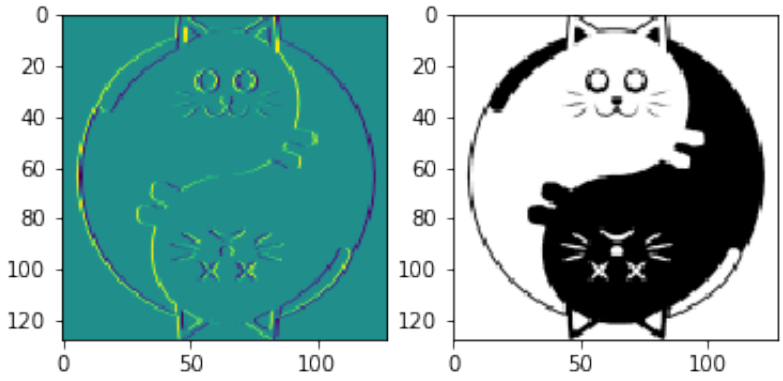
This demonstration shows how to use Qiskit™ to perform edge detection in images with the quantum Hadamard edge detection algorithm, which completes the task with just one single-qubit operation, independent of the size of the image, illustrating the potential of quantum image processing for highly efficient image and video processing.

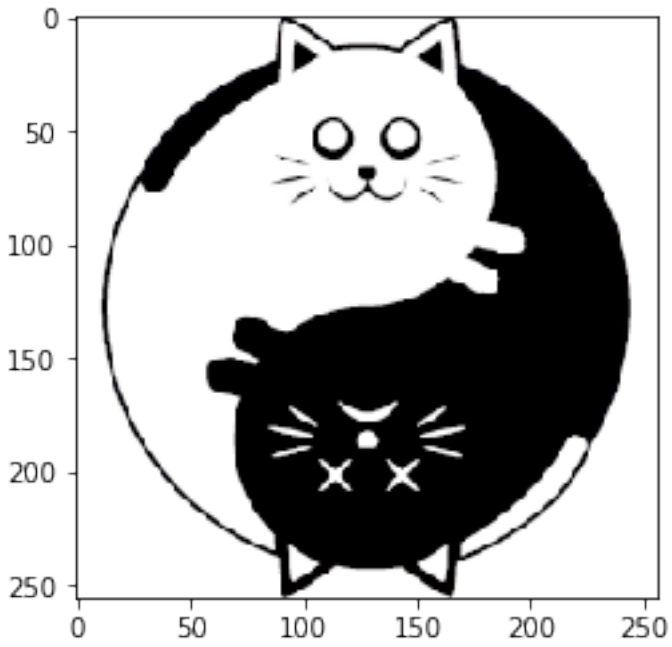
Read the example image in and display it.

In [1]:
```
import numpy as np
from numpy import pi
import matplotlib.pyplot as plt
import matplotlib.image  as mpimg
```

In [2]:
```
filename = './schrodin_yang.png'
im = mpimg.imread(filename)
fig, ax = plt.subplots()
ax.imshow(im)
```

Out[2]:
```
<matplotlib.image.AxesImage at 0x11a75aa90>
```



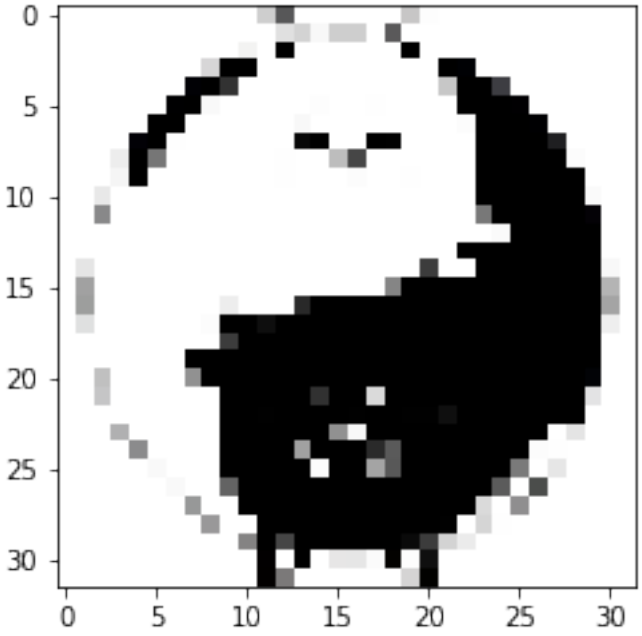


If necessary, shrink the image to make the rest of the demonstration run faster.

```
In [3]: from skimage.transform import resize
```

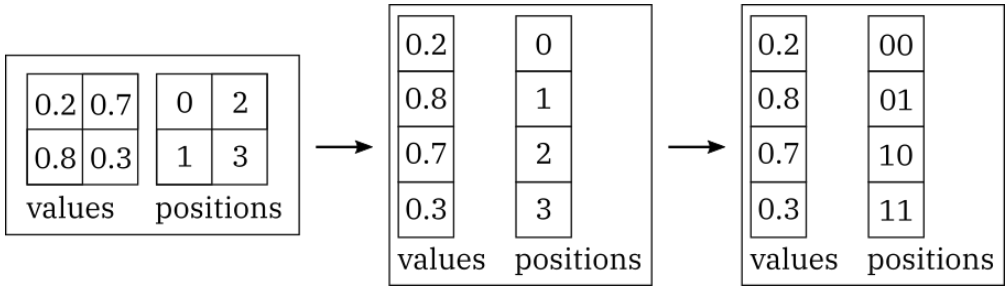
```
In [4]: n_pixels = 2**5
        im = resize(im, (n_pixels, n_pixels))
        fig, ax = plt.subplots()
        ax.imshow(im)
```

```
Out[4]: <matplotlib.image.AxesImage at 0x81c0fcba8>
```



1.1 Representing the Image as a State Vector

Represent the image as a quantum state as follows



The image begins as a matrix of values at certain pixel positions. Unravel the matrices to form vectors. Then translate the pixel positions from decimal to binary.

The image can then be represented as the quantum state

$$|\text{image}\rangle = \frac{0.2|00\rangle + 0.8|01\rangle + 0.7|10\rangle + 0.3|11\rangle}{\sqrt{0.2^2 + 0.8^2 + 0.7^2 + 0.3^2}},$$

or in general, $|\text{image}\rangle = \sum_k \alpha_k |k\rangle$, where the basis state $|k\rangle$ encodes the position of each pixel and the amplitude α_k encodes the pixel value.

Get the first color channel of the image and `ravel` it to form a 1-D vector.

```
In [5]: data = im[:,:,0].ravel()
```

Use Qiskit Aqua to encode the image as a quantum state.

```
In [6]: n_qubits    = np.int_(np.log2(len(data)))

        from qiskit_aqua.components.initial_states import Custom
        init_state = Custom(n_qubits, state_vector=data)
```

Get the quantum circuit `circ` that creates the image representation and its quantum register `qr` containing qubits. The circuit's register begins in the zero state $|000\ldots000\rangle$ (as many zeros as there are qubits) that get modified by quantum gates to end up in the state that represents the image, $|\text{image}\rangle$.

```
In [7]: circ = init_state.construct_circuit('circuit')
        qr   = circ.qregs
        #circ.draw()
```

1.2 How the Quantum Hadamard Edge Detection Algorithm Works

The Hadamard gate has the following effect on the zero and one basis states:
$H|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}$
$H|1\rangle \rightarrow (|0\rangle - |1\rangle)/\sqrt{2}$
Consider a four pixel image and what happens when a Hadamard gate is applied to the last qubit. We'll subscript the Hadamard operator to tell which qubit it's acting on.

$$|\text{image}\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

$$|\text{image}\rangle = \alpha_{00}|0\rangle|0\rangle + \alpha_{01}|0\rangle|1\rangle + \alpha_{10}|1\rangle|0\rangle + \alpha_{11}|1\rangle|1\rangle$$

$$\sqrt{2}H_0|\text{image}\rangle = \alpha_{00}|0\rangle(|0\rangle + |1\rangle) + \alpha_{01}|0\rangle(|0\rangle - |1\rangle) + \alpha_{10}|1\rangle(|0\rangle + |1\rangle) + \alpha_{11}|1\rangle(|0\rangle - |1\rangle)$$

$$\sqrt{2}H_0|\text{image}\rangle = \alpha_{00}(|00\rangle + |01\rangle) + \alpha_{01}(|00\rangle - |01\rangle) + \alpha_{10}(|10\rangle + |11\rangle) + \alpha_{11}(|10\rangle - |11\rangle)$$

$$\sqrt{2}H_0|\text{image}\rangle = (\alpha_{00} + \alpha_{01})|00\rangle + (\alpha_{00} - \alpha_{01})|01\rangle + (\alpha_{10} + \alpha_{11})|10\rangle + (\alpha_{10} - \alpha_{11})|11\rangle$$

$$\sqrt{2}H_0|\text{image}\rangle = (\alpha_{00} + \alpha_{01})|00\rangle + (\alpha_{00} - \alpha_{01})|01\rangle + (\alpha_{10} + \alpha_{11})|10\rangle + (\alpha_{10} - \alpha_{11})|11\rangle$$

The terms with the difference of neighboring pixel values tell us information about the image's edges. Here's why. Consider the following row of pixels called p and the difference of its neighboring pixels called Δp:

$$p = [0, 0, 0, 1, 1, 1, 0, 0, 0]$$

$$\Delta p = [p(1) - p(0), p(2) - p(1), \ldots] = [0, 0, 1, 0, 0, -1, 0, 0]$$

The neighboring differences Δp take on non-zero values where there are changes (edges) in the original image p.

If we measure the state of the first qubit, and the result is 1, then the state will have collapsed into the final state proportional to

$$|\text{final state}\rangle = (\alpha_{00} - \alpha_{01})|01\rangle + (\alpha_{10} - \alpha_{11})|11\rangle$$

which holds just the edge information we're interested in.

1.3 Implementing the Algorithm

Beginning with the circuit `circ` that generates the image's state vector representation, apply the Hadamard gate to the first qubit.

```
In [8]: circ.h(qr[0][0])
```

```
Out[8]: <qiskit.extensions.standard.h.HGate at 0x1c21a15048>
```

Simulate the circuit using the `StatevectorSimulator` and read the resulting state vector.

```
In [9]: from qiskit import BasicAer, execute
```

```
In [10]: simulator = BasicAer.get_backend('statevector_simulator')
         result = execute(circ, simulator).result()
         final_state_vector = result.get_statevector(circ)
```

1.4 Decode the State Vector Back into an Image

```
In [11]: edge = np.real(final_state_vector)
         n_rows    = int(np.sqrt(len(edge)))
         edge = edge.reshape(n_rows, -1)
```

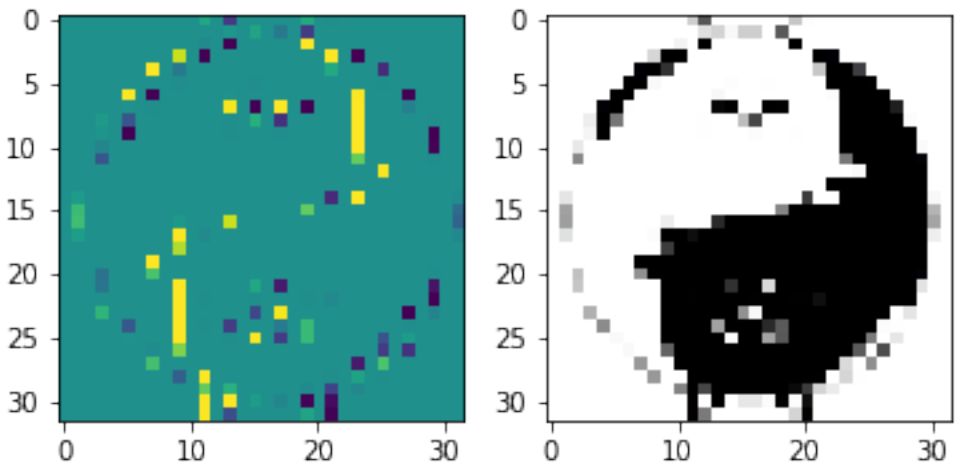
The edges are indicated by the basis states where the first qubit is 1. After decoding the 1-D state vector back into a 2-D image, these basis states are the 2-D image's even columns. To retain only these columns, zero out the odd columns.

```
In [12]: edge[:,::2] = 0
```

Display the edges and the original image for comparison.

```
In [13]: fig, ax = plt.subplots(1,2)
         ax[0].imshow(edge)
         ax[1].imshow(im)
```

```
Out[13]: <matplotlib.image.AxesImage at 0x1c21897b00>
```



Notice we've only found the edges in every other column of the image. A modification of the QHED algorithm can find the edges in all columns with a single circuit, as described in Yao, Xi-Wei et al., Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment, Phys. Rev. X 7, 031041, (2017).