Quantum Hadamard Edge Detection

Manuel Di Agostino

12 febbraio 2025

Università di Parma

Sommario

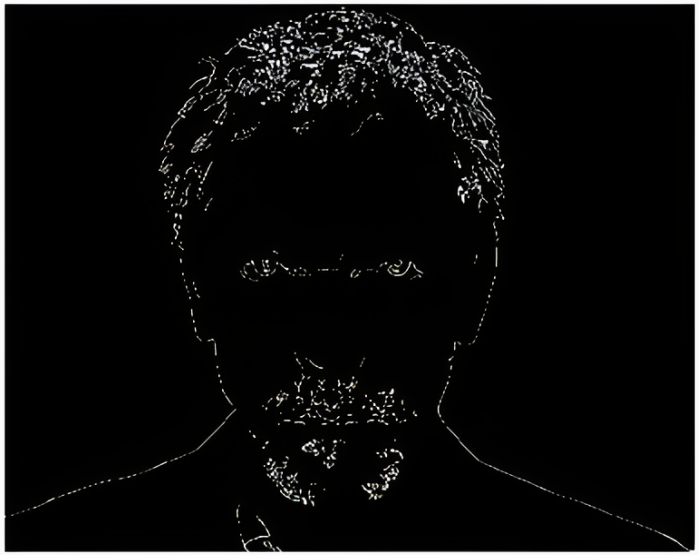
Sommario

Sommario

Edge detection

- **Goal**: identificare oggetti in un'immagine, tramite *contorni*
- Ma anche:
  - estrazione di texture, pattern
  - motion recognition
  - image restoration

Soluzioni classiche



- Lavorano sull'immagine in scala di grigi
- Approssimano il gradiente dell'intensità
- Utilizzano operatori convolutivi (*kernel*)

Filtri di Sobel

- Fu introdotto negli anni '60 [1]
- Applica due kernel di convoluzione

$$\mathbf{G_x} = \underbrace{\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}}_{\text{direzione orizzontale}}, \quad \mathbf{G_y} = \underbrace{\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}_{\text{direzione verticale}}$$

Filtri di Sobel

- Fu introdotto negli anni '60 [1]
- Applica due kernel di convoluzione

$$\mathbf{G_x} = \underbrace{\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}}_{\text{direzione orizzontale}}, \quad \mathbf{G_y} = \underbrace{\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}}_{\text{direzione verticale}}$$

- Il nuovo valore di intensità del pixel $p_{j,k}$ è

$$\tilde{p}_{j,k} = \sqrt{G_x(p_{j,k})^2 + G_y(p_{j,k})^2}$$

Filtri di Sobel

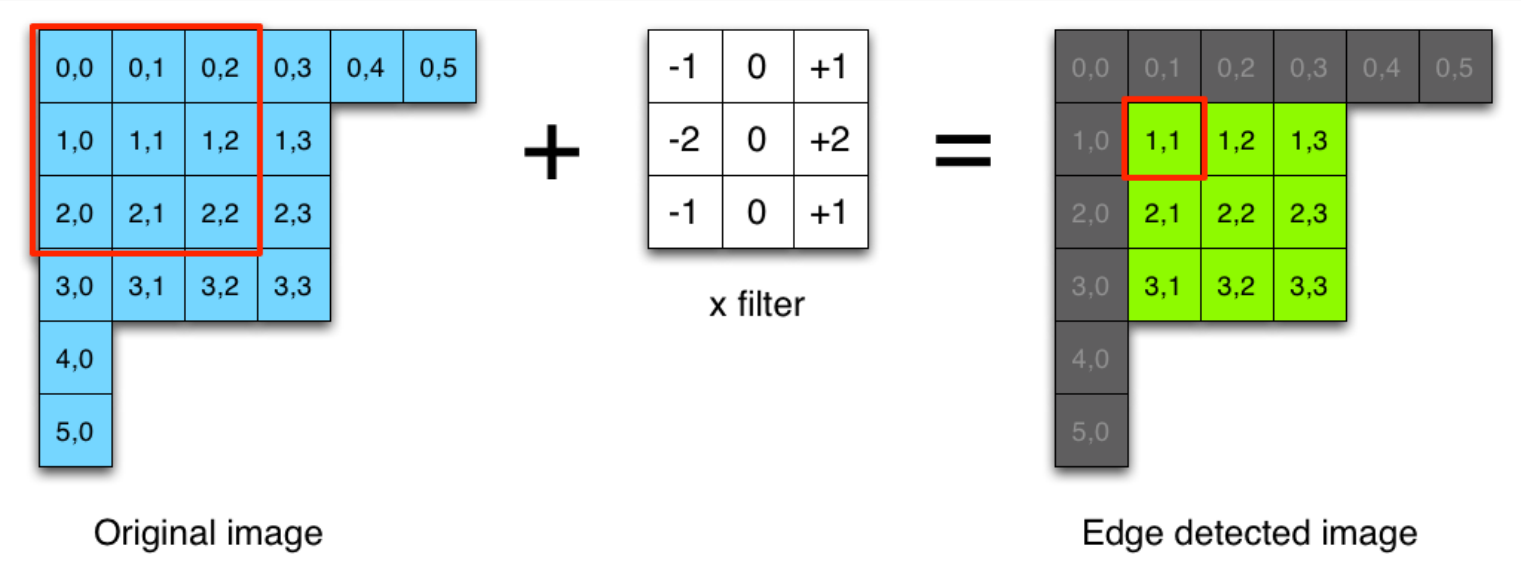


Figura 1: Applicazione orizzontale del kernel di Sobel.

- Complessità *lineare* rispetto al numero di pixel!

Sommario

Sistemi quantistici

- Unità di informazione: **qubit**
- Notazione *di Dirac*

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \alpha, \beta \in \mathbb{C}$$

- Verificano

$$|\alpha|^2 + |\beta|^2 = 1$$

Sistemi quantistici

- Unità di informazione: **qubit**
- Notazione *di Dirac*

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \alpha, \beta \in \mathbb{C}$$

- Verificano

$$|\alpha|^2 + |\beta|^2 = 1$$

- Sistemi di più qubit descritti utilizzando il *prodotto tensore*

$$|\psi_1\rangle = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad |\psi_2\rangle = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

$$|\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} a_1 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ a_2 \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{bmatrix}.$$

Sommario

Quantum gate

- L'azione di un **quantum gate** è modellata matematicamente tramite l'applicazione di una matrice *complessa unitaria* ad uno stato quantistico

$$|\psi'\rangle = U|\psi\rangle$$

Quantum gate

- L'azione di un **quantum gate** è modellata matematicamente tramite l'applicazione di una matrice *complessa unitaria* ad uno stato quantistico

$$|\psi'\rangle = U|\psi\rangle$$

- Porte rilevanti:
  - Gate *di Pauli*

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

  - Gate *di Hadamard*

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

Sommario

Quantum Image Processing

Due obiettivi principali:

- Codifica dell'immagine nei circuiti
  - Flexible Representation of Quantum Images (FRQI) [2]
  - Novel Enhanced Quantum Representation (NEQR) [3]
  - **Quantum Probability Image Encoding (QPIE)** [4]

Quantum Image Processing

Due obiettivi principali:

- Codifica dell'immagine nei circuiti
  - Flexible Representation of Quantum Images (FRQI) [2]
  - Novel Enhanced Quantum Representation (NEQR) [3]
  - **Quantum Probability Image Encoding (QPIE)** [4]
- Algoritmi di processamento
  - QSobel [5]
  - **Quantum Hadamard Edge Detection (QHED)** [4]

Quantum Probability Image Encoding

- Valori di intensità dei pixel codificati nelle ampiezze di probabilità dello stato quantistico
- Immagine di N pixel. Qubit necessari:

$$n = \lceil log_2 N \rceil$$

Quantum Probability Image Encoding

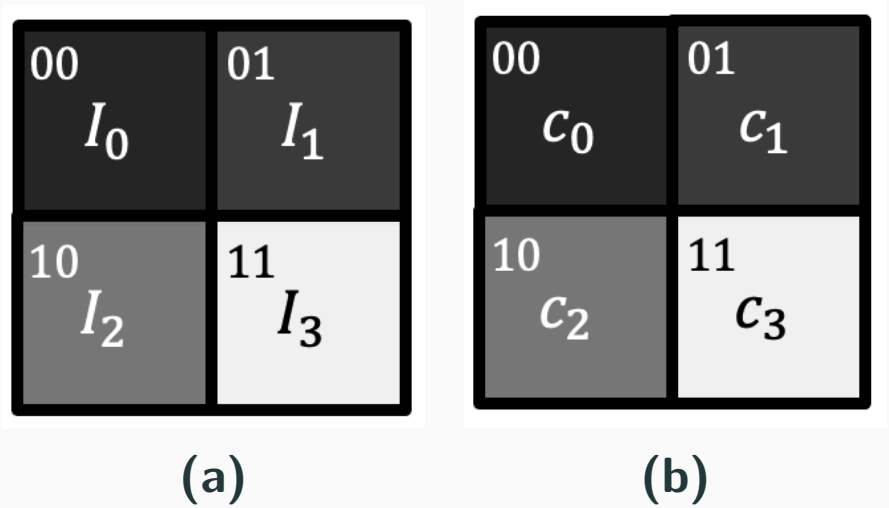


Figura 2: (2a) rappresentazione classica, (2b) rappresentazione quantistica.

- Pixel numerati utilizzando stringhe binarie
- Intensità normalizzate secondo

$$c_i = \frac{I_i}{\sqrt{\sum_k I_k^2}}$$

Quantum Probability Image Encoding

- Nell'esempio (2b) si ottiene

$$|\mathrm{Img}\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Quantum Probability Image Encoding

- Nell'esempio (2b) si ottiene

$$|\text{Img}\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- Generalizzando per n qubit

$$|\text{Img}\rangle = \sum_{i=1}^{2^n} c_i|i\rangle = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix}$$

Quantum Hadamard Edge Detection i

- Fa uso della QPIE
- **Idea di base**: utilizzare il *gate di Hadamard* sul qubit q_0

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

Quantum Hadamard Edge Detection ii

- Applicando questa trasformazione a $|\text{Img}\rangle$ si ottiene lo stato

$$(I_{2^{n-1}} \otimes H_0) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{bmatrix}$$

  che esplicita il gradiente di coppie (*pari*) di pixel adiacenti
- Questo è riconducibile a

$$\frac{1}{\sqrt{2}}(|\text{sum}\rangle \otimes |0\rangle + |\text{dif}\rangle \otimes |1\rangle)$$

- **Punto chiave**: misurare il circuito a condizione che q_0 sia nello stato $|1\rangle$

QHED: variazione i

- È possibile calcolare in una sola passata il gradiente di coppie pari e dispari di pixel
- **Idea:** qubit aggiuntivo inizializzato in $|+\rangle$

$$|\text{Img}\rangle \otimes \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 \\ c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_{N-1} \end{bmatrix}$$

QHED: variazione ii

- Applicando uno *shift* allo stato iniziale

$$D_{2^{n+1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 1 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \end{bmatrix}$$

QHED: variazione iii

- Si ottiene lo stato

$$D_{2^{n+1}} \cdot \begin{bmatrix} c_0 \\ c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_{N-1} \\ c_0 \end{bmatrix}$$

QHED: variazione iv

- Infine

$$(I_{2^n} \otimes H_a) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_{N-1} \\ c_0 \end{bmatrix} = \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_1 + c_2 \\ c_1 - c_2 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-1} + c_0 \\ c_{N-1} - c_0 \end{bmatrix}$$

Sommario

Sommario

Normalizzazione dell'immagine

- Normalizzazione dell'immagine

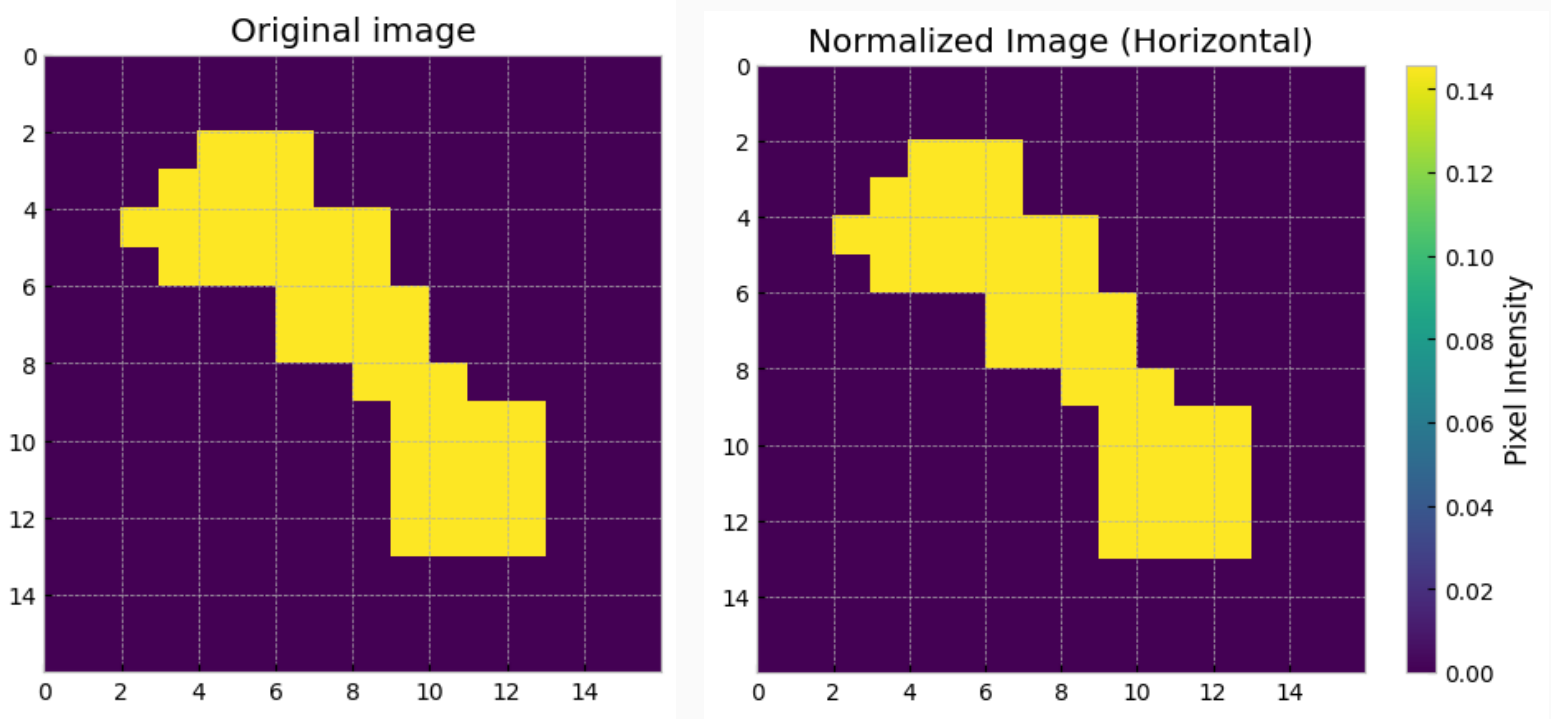
```
1  def amplitude_encode(img_data):
2      # Calculate the RMS value
3      rms = np.sqrt(np.sum(np.sum(img_data**2, axis=1)))
4      # Create normalized image
5      image_norm = []
6      for arr in img_data:
7      for ele in arr:
8      image_norm.append(ele / rms)
9      # Return the normalized image as a numpy array
10     return np.array(image_norm)
11
12 # Get the amplitude ancoded pixel values
13 image_norm_h = amplitude_encode(image)
```

Il circuito

- Strutturazione

```
from qiskit import *
from qiskit import transpile
from qiskit_aer import Aer

# Create the circuit for horizontal scan
qc_h = QuantumCircuit(total_qb)
qc_h.initialize(image_norm_h, range(1, total_qb))
qc_h.h(0)
qc_h.unitary(D2n_1, range(total_qb))
qc_h.h(0)
```

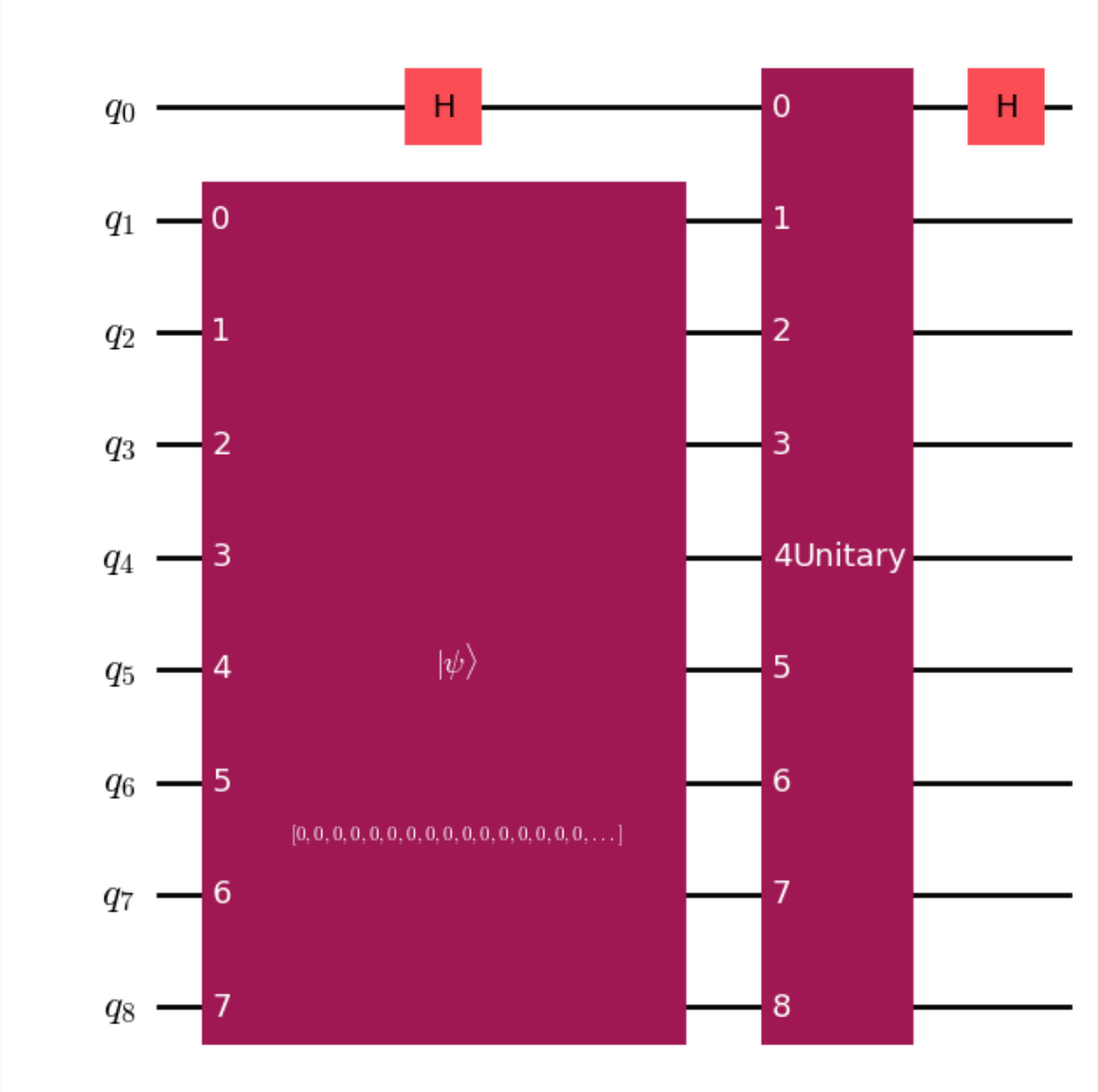
- Immagine di esempio 16x16 in input



(a) Immagine originale **(b)** Immagine normalizzata

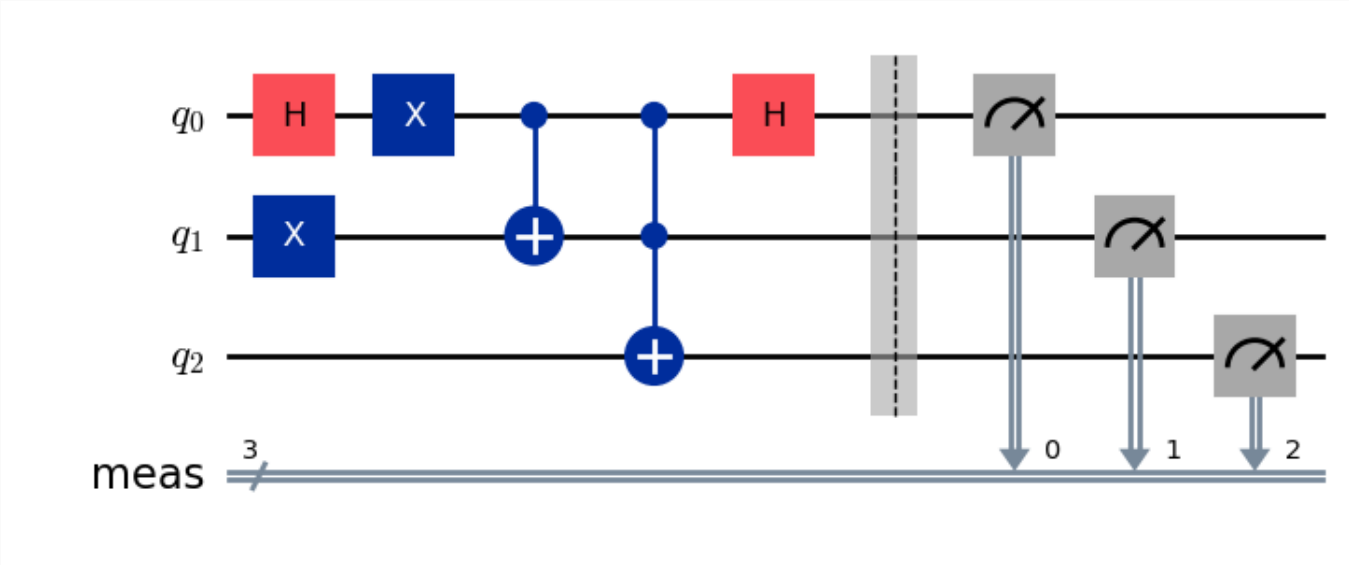
Figura 3: Confronto tra l'immagine originale (3a) e l'immagine normalizzata (3b).

- Circuito risultante



Codifica tramite gate

- Per un'immagine 2x2



Sommario

La gestione degli errori

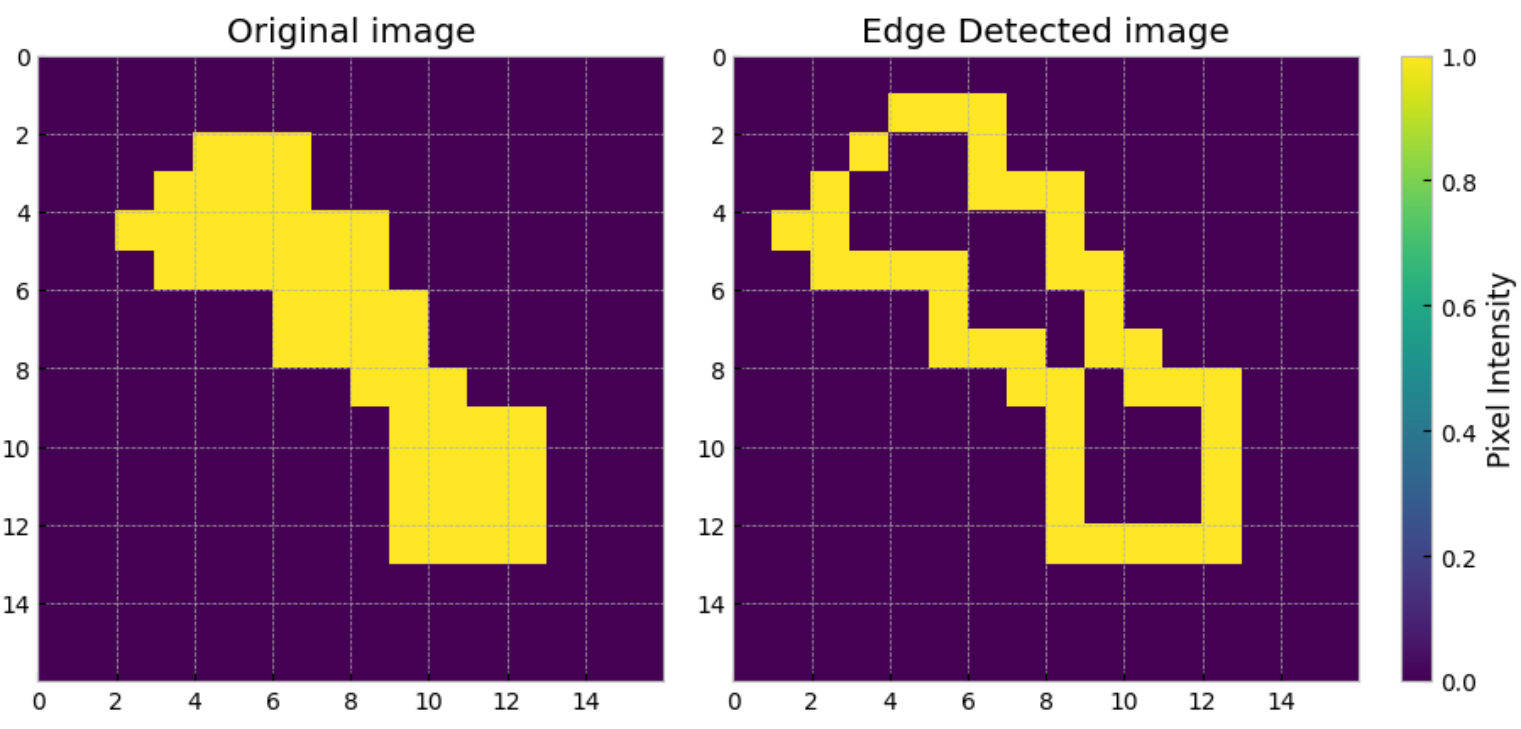
- È stata utilizzata la classe `NoiseModel` di Qiskit
- Dati estrapolati dal backend reale `ibm_kyiv`

```
from qiskit_aer import AerSimulator
from qiskit_aer.noise import NoiseModel

from qiskit_ibm_runtime import QiskitRuntimeService
service = QiskitRuntimeService(channel="ibm_quantum", token="XXX")

backend = service.backend("ibm_kyiv")
from qiskit_aer.noise import (NoiseModel, QuantumError, ReadoutError,
    pauli_error, depolarizing_error, thermal_relaxation_error)

noise_model = NoiseModel.from_backend(backend)
# Get coupling map from backend
coupling_map = backend.configuration().coupling_map
# Get basis gates from noise model
basis_gates = noise_model.basis_gates
```

Sommario

Esecuzione ideale

- Inizialmente esecuzione simulata in ambiente ideale, tramite `statevector_simulator`

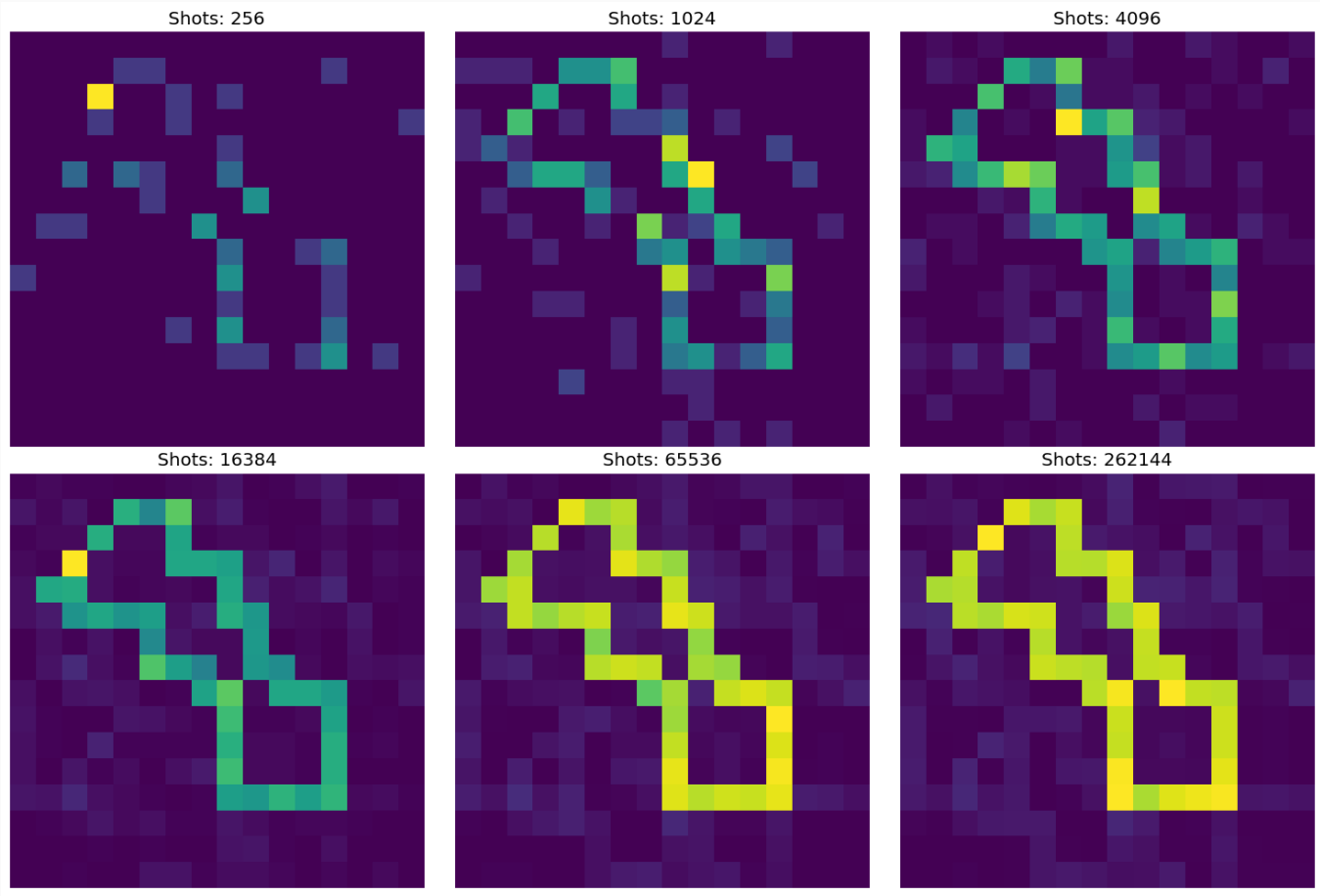


Esecuzione con rumore

- Successivamente è stato aggiunto il modello di rumore

Esecuzione con rumore

- Successivamente è stato aggiunto il modello di rumore
- Generale miglioramento all'aumentare del numero di *shots*



Il *transpiling*

- *Transpiling* oneroso, anche per immagini 4x4
- Per un'immagine 2x2, profondità del circuito risultante pari a 64

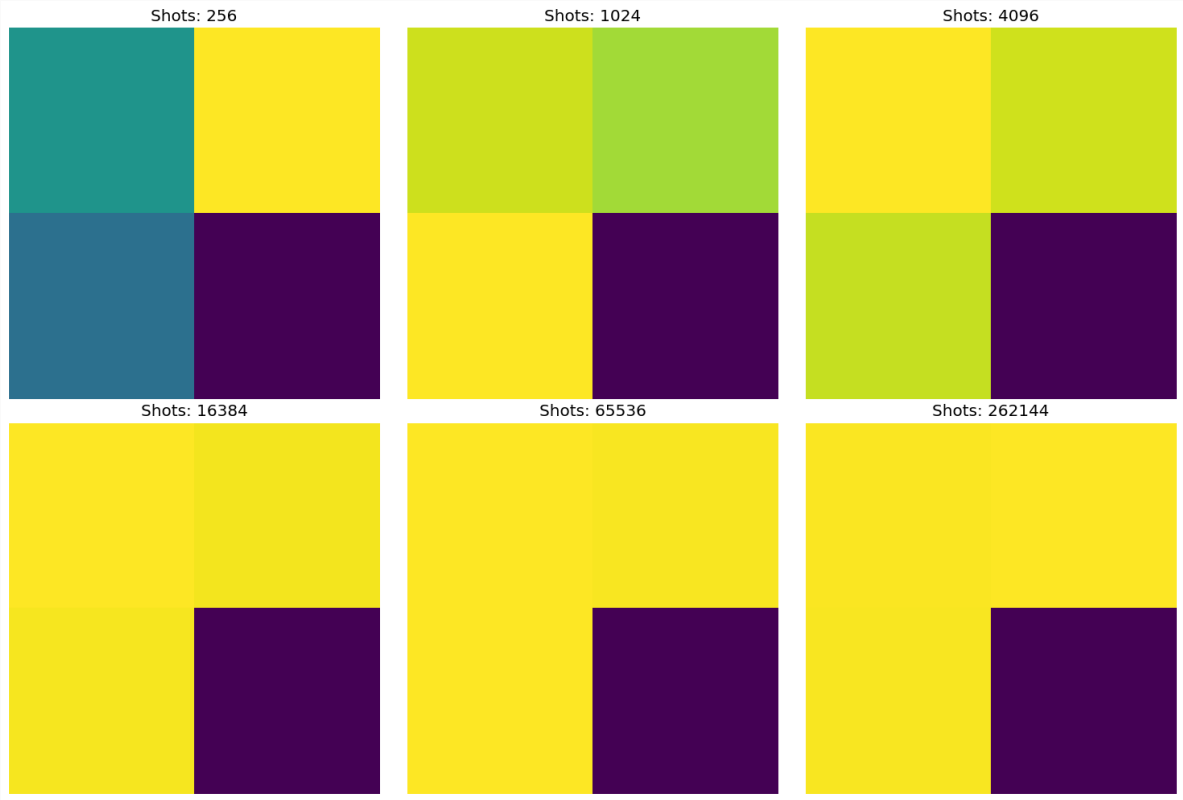


Figura 4: Simulazione con rumore con variazione sul numero di shots, immagine 2x2

Complessità spaziale e computazionale

Sia $N = 2^n, n \in \mathbb{N}$ la dimensione dell'immagine.

Algoritmo	**Costo Spaziale**	**Costo Computazionale**[1]
Classici	$\mathcal{O}(N = 2^n)$	$\mathcal{O}(N = 2^n)$
QSobel	$\mathcal{O}(2n + 1)$	$\mathcal{O}(n^2)$
QHED	$\mathcal{O}(n + 1)$	$\mathcal{O}(1)$

Tabella 1: Confronto tra il costo spaziale e computazionale di diversi algoritmi

[1] In riferimento al solo calcolo dei gradienti.

Bibliografia i

[1] Irwin Sobel e Gary Feldman. «An Isotropic 3x3 Image Gradient Operator». In: (1968). Presented at the Stanford Artificial Intelligence Laboratory (SAIL). URL: https://www.researchgate.net/publication/281104656_An_Isotropic_3x3_Image_Gradient_Operator.

[2] Phuc Q. Le, Fangyan Dong e Kaoru Hirota. «A flexible representation of quantum images for polynomial preparation, image compression, and processing operations». In: *Quantum Information Processing* 10.1 (2011), pp. 63–84. DOI: 10.1007/s11128-010-0177-y. URL: https://doi.org/10.1007/s11128-010-0177-y.

Bibliografia ii

[3] Yi Zhang et al. «NEQR: a novel enhanced quantum representation of digital images». In: *Quantum Information Processing* 12.8 (2013), pp. 2833–2860. DOI: 10.1007/s11128-013-0567-z. URL: https://doi.org/10.1007/s11128-013-0567-z.

[4] Xi-Wei Yao et al. «Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment». In: *Physical Review X* 7.3 (2017), p. 031041. DOI: 10.1103/PhysRevX.7.031041. URL: https://doi.org/10.1103/PhysRevX.7.031041.

[5] Yi Zhang et al. «QSobel: A novel quantum image edge extraction algorithm». In: *Science China Information Sciences* 57.11 (2014), pp. 1–9. DOI: 10.1007/s11432-014-5158-9. URL: https://doi.org/10.1007/s11432-014-5158-9.