

UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

IQ01 - RAPPORT DE PROJET

Réalisation d'une application quantique : traitement d'image quantique

Étudiants :

Louis Soto
Alexandre Lepicier
Jamil Bouhdada
Alexandre Constantin

Responsable :
Ahmed Lounis

Résumé

Dans ce rapport, nous faisons un état de l'art d'abord sur les méthodes de réalisation d'ordinateur quantique, et ensuite sur la représentation d'image quantique et la détection de contour quantique. Nous revenons sur les principaux défis vis-à-vis de la fabrication d'un ordinateur quantique et nous présentons les trois technologies les plus prometteuses à ce jour : les supraconducteurs, les anyons et les ions piégés. Pour la partie image, nous exposons trois méthodes de représentation d'images différentes : Flexible Representation of Quantum Images (FRQI), Novel Enhanced Quantum Representation (NEQR) et Quantum Probability Image Encoding (QPIE). Enfin nous présentons un algorithme quantique de détection de contour, Quantum Hadamard Edge Detection (QHED), avant de l'implémenter et de présenter ses résultats.

Table des matières

1 Introduction à l'ordinateur quantique	3
2 Ordinateur quantique	3
2.1 Les principaux défis des systèmes quantiques physiques	3
2.2 Réalisation de qubits physiques	5
3 Traitement d'image et informatique quantique	9
3.1 Traitement d'image classique	9
3.1.1 Représentation d'image	9
3.1.2 Edge detection/Edge filter	10
3.2 Représentation d'image en informatique quantique	11
3.2.1 Technique de représentation flexible d'images quantiques (FRQI : Flexible Representation of Quantum Images)[5]	11
3.2.2 Technique de représentation nouvelle quantique améliorée (Novel Enhanced Quantum Representation (NEQR) for Digital Images	13
3.2.3 La technique du codage d'image quantique par probabilité quantique (QPIE : the Quantum Probability Image Encoding)	14
3.3 Détection de contours en informatique quantique	16
3.3.1 Principe de l'algorithme	17
3.3.2 Variation de l'algorithme QHED (Utilisation d'un qubit auxiliaire)	18
3.3.3 Circuit Quantique	19
3.3.4 Analyse de la complexité temporelle et spatiale de QHED	20
4 Application et tests	22
4.1 Technique de représentation flexible des images quantiques (FRQI)	22
4.2 Application des nouvelles techniques de représentation quantique améliorée (NEQR)	24
4.3 Utilisations des QImR pour effectuer la détection des contours à l'aide de l'algorithme Quantum Hadamard Edge Detection(QHED)	26
4.3.1 Implémentation et application de l'algorithme QHED sur une image simplifiée	26
4.3.2 Utilisation de l'algorithme QHED implémenté sur une image réelle	28
4.3.3 Conclusion et Reflexion sur nos applications concernant QHED	29
5 Conclusion	30

1 Introduction à l'ordinateur quantique

Au cours de l'UV IQ01 nous nous intéressons en profondeur aux qubits logiques qui nous permettent de mettre au point des circuits et des algorithmes quantiques en tirant parti des propriétés de superposition et d'intrication quantique. Toute cette théorie autour de l'informatique quantique nécessite néanmoins une implémentation physique de qubits ainsi qu'un moyen d'interagir avec leurs états, ceci constituerait l'ordinateur quantique. Il faudrait mettre au point une technologie qui possède un comportement quantique et qui soit adaptée aux besoins informatiques imposés par la théorie. De nombreuses technologies manifestent un comportement quantique, mais à ce jour aucune ne correspond totalement à l'informatique quantique. Les méthodes qui sont en train d'être développées pour implémenter des qubits physiques sont confrontées à des problèmes de stabilité, de décohérence, de tolérance aux fautes et du passage à grande échelle. Dans cette première partie, nous passerons en revue différentes technologies permettant d'implémenter des qubits physiques et nous nous intéresserons à leurs limites actuelles.

2 Ordinateur quantique

2.1 Les principaux défis des systèmes quantiques physiques

Comme il a été énoncé précédemment, les technologies actuelles permettant de matérialiser des qubits physiques se heurtent à des soucis de stabilité, de tolérance aux fautes et de passage à grande échelle. Nous allons voir plus en détail ce que ces termes signifient avant de passer aux différents systèmes physiques.

Décohérence et Stabilité :

Quand un algorithme quantique est élaboré, il présume que les qubits manipulés sont "parfaits" et qu'ils se comportent exactement comme prévu. Mais en réalité, les interactions possibles avec les autres particules du milieu peuvent perturber les états de nos qubits, ce qui affecte la stabilité de l'information stockée dans nos registres de qubits. Il est possible d'améliorer cette stabilité en utilisant des codes correcteurs d'erreurs, il en résulte qu'un qubit physique n'est pas équivalent à un qubit logique, et que plusieurs qubits physiques sont nécessaires pour former un unique qubit logique.

Ce phénomène de perturbation des états quantiques s'appelle la décohérence quantique. En théorie si un système quantique est parfaitement isolé, il reste cohérent indéfiniment. Mais ceci est rarement le cas (notamment pendant la mesure) et la réduction des fonctions d'ondes des qubits mène à la disparition des états quantiques et donc à une perte d'information. Cette décohérence est irréversible et augmente avec le temps, car il y a de plus en plus d'interaction avec l'environnement. Opérer à très basse température pour limiter au maximum le mouvement des particules est un moyen de ralentir cette décohérence inévitable. La conséquence majeure de la décohérence étant : les qubits n'ont pas une durée de vie infinie, contrairement aux bits classiques. On peut ainsi introduire la notion de durée de vie d'un qubit, c'est-à-dire le temps qu'un qubit reste cohérent avant d'avoir subi une perte trop grande d'information et ainsi être inexploitable. Cette durée est aussi appelée "temps de décohérence", et son ordre de grandeur varie de la nanoseconde à la seconde en fonction de la technologie utilisée pour le qubit en question.

Finalement, puisque nous ne pouvons pas éviter la décohérence, ou bien que nous ne savons pas encore comment faire pour l'éviter, les portes quantiques qui opèrent sur nos qubits doivent donc prendre en compte cette durée de vie. Leurs actions sur nos qubits doivent être assez rapide pour que nous puissions exploiter nos qubits et faire du calcul.

Initialisation des qubits :

Un autre souci dans la matérialisation des qubits est l'initialisation. Dans les frameworks d'informatique quantique comme qiskit, il est facile de tenir pour acquis l'initialisation parfaite des qubits dans l'état que l'on souhaite. Mais ceci n'est pas aussi simple à réaliser physiquement, il est difficile d'initialiser des qubits rapidement (relativement à la durée de vie des qubits) et avec une haute exactitude. Ceci accroît les erreurs et l'incertitude des circuits quantiques.

Jeu universel de portes quantiques :

Il existe une infinité d'opérations quantiques possibles et il est alors invraisemblable de toutes les implémenter dans notre ordinateur quantique. Il est plus judicieux de trouver un sous-ensemble fini de ces opérations permettant de toutes les reconstruire avec une suite finie d'opérations de ce sous-ensemble, c'est ce qu'on appelle un "jeu universel" de portes quantiques. Ceci est irréalisable techniquement, car le nombre de portes à reconstruire est infini et nous nous limitons à l'utilisation d'un nombre fini de portes. Pour pallier ce problème, on s'autorise à approximer chaque opération par une suite finie de portes dans notre jeu universel. Grâce à cela, il suffit de trouver un nombre réduit de portes qui vont nous permettre, en les combinant, de faire n'importe quelle opération que l'on souhaite sur nos qubits.

Un jeu universel de portes quantiques connu est le jeu Clifford+T, qui regroupe les portes CNOT, Hadamard, S et T. Ces quatre portes sont suffisantes pour reconstruire n'importe quelle autre porte quantique, même les portes à plusieurs qubits. Plus précisément, on reconstruit avec ces quatre portes une approximation, avec un degré de précision arbitraire, de n'importe quelle autre porte. Par conséquent, il est suffisant dans un ordinateur quantique de pouvoir faire uniquement ces 4 opérations sur les qubits. En fonction de la technologie utilisée pour implémenter les qubits, il faut aussi mettre au point un moyen d'interagir avec eux et de modifier leurs états, ce qui équivaut à leur appliquer des portes quantiques. Pour qu'un ordinateur quantique soit fonctionnel il doit pouvoir appliquer physiquement un jeu universel de portes quantiques sur ses qubits.

Mesure des qubits :

Pour exploiter les qubits que nous matérialisons et en faire un réel ordinateur quantique, il faut pouvoir mesurer l'état de nos qubits après les avoir manipulé avec des portes quantiques. Mais en fonction de quelle technologie est utilisée pour faire des qubits, ils sont plus ou moins difficile à lire et il faut aussi prendre en compte le temps de décohérence. La mesure requiert que le système ne soit pas isolé donc il y a un compromis à trouver entre la décohérence et la facilité à mesurer.

Tolérance aux fautes, codes de corrections d'erreurs :

On a vu précédemment que les réalisations physiques de qubits sont soumises à la décohérence et de ce fait l'information quantique possède beaucoup d'erreurs. Pour contrer cela il faut que notre ordinateur quantique soit un système tolérant aux fautes, c'est-à-dire qu'il puisse continuer de fonctionner avec une certaine qualité malgré l'occurrence d'erreurs. Pour cela il est possible de mettre en place des codes de correction d'erreurs qui ont pour but de conserver l'intégrité dans notre information quantique. La correction d'erreur sur des bits classiques se base largement sur la redondance, essentiellement stocker plusieurs fois l'information. Cependant, il est impossible fondamentalement

de cloner un qubit, et en conséquence de copier de l'information stockée quantiquement. Il faut donc une autre approche pour des codes de correction d'erreurs quantiques. Un exemple connu est le code de Shor dont le principe est de propager l'information d'un qubit sur plusieurs qubits fortement intriqués.

Passage à l'échelle (scalabilité) :

La dernière propriété que nous souhaitons avoir avec les technologies qui nous permettent de matérialiser des qubits est la facilité du passage à l'échelle. On veut pouvoir faire grandir la taille du système, dans notre cas cela veut dire augmenter le nombre de qubits, en gardant une architecture similaire et sans perte de performance ou de stabilité.

2.2 Réalisation de qubits physiques

Il existe une multitude de méthodes différentes pour réaliser des qubits physiques, elles sont toutes au stade de recherche et il n'est pas possible de savoir laquelle sera retenue puisqu'elles possèdent chacune leurs particularités, avantages et inconvénients. Dans cette partie nous allons vous présenter les quatre technologies les plus prometteuses et les plus développées dans l'état de l'art actuel.

Supraconducteurs (transmons) :

Les circuits supraconducteurs sont un moyen de former un ordinateur quantique, c'est une technologie qui est développée par des entreprises notables comme Google, IBM ou Intel. Cette méthode exploite l'effet Josephson, qui se manifeste lorsqu'une fine couche d'isolant sépare deux matériaux supraconducteurs (appelée jonction Josephson). Les jonctions de Josephson se manipulent et se comportent comme des objets quantiques, avec elles on peut matérialiser des qubits de trois manières différentes. Des qubits de charge (transmon), de flux (SQUID), et de phase, nous allons nous intéresser plus en profondeur dans le qubit de type transmon qui est à ce jour le plus abouti.

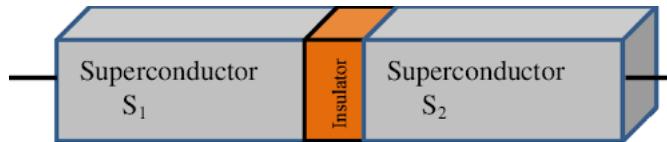


FIGURE 1 – Jonction Josephson

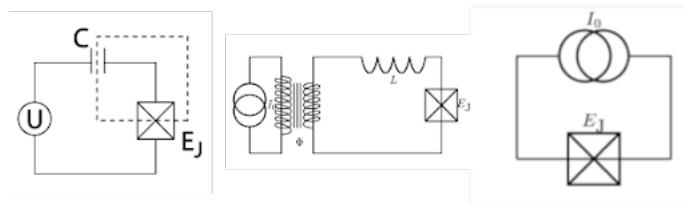


FIGURE 2 – Architecture des qubits de charge, de flux, et de phase

Il faut agencer les jonctions de Josephson d'une façon particulière pour former un circuit sur une puce pour matérialiser des qubits. Sur cette puce, il est possible de lier les qubits entre eux et donc d'avoir la possibilité de les intriquer. Cette puce est ensuite placée au fond d'une cuve avec au-dessus un système complexe de tuyaux permettant de refroidir la puce jusqu'à 0.02 Kelvin

grâce à de l'hélium. Cette basse température est nécessaire d'une part pour observer le phénomène de supraconductivité dans le circuit, mais aussi car la chaleur est un élément qui induit beaucoup d'erreurs dans les qubits. L'agitation thermique des particules environnantes facilite la décohérence des qubits et introduit des erreurs que l'on veut éviter à tout prix.

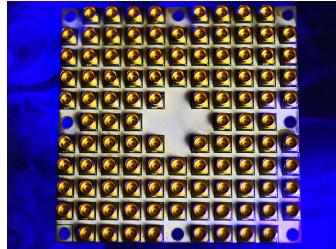


FIGURE 3 – Puce quantique d'Intel



FIGURE 4 – Ordinateur quantique d'IBM à base de supraconducteurs, cuve de refroidissement

Après avoir matérialisé nos qubits, il faut aussi pouvoir interagir avec eux, c'est-à-dire appliquer des portes quantiques sur eux et modifier leurs états, les intriquer et les mesurer. Avec les supraconducteurs et les jonctions de Josephson, cette communication se fait en envoyant des micro-ondes sur les qubits. Ces micro-ondes ont des fréquences, des durées et des formes très spécifiques et elles agissent sur les états quantiques des qubits physiques. Comme nous connaissons les actions que font nos portes quantiques sur les états des qubits logiques, nous pouvons faire correspondre à une porte quantique une micro-onde qui effectue le changement d'état voulu au qubit physique. L'application d'une porte quantique se fait donc en envoyant sur les qubits la micro-onde qui correspond à cette porte.

Pour ce qui est de la mesure des qubits de type transmon, on couple les qubits avec des résonateurs de lecture qui vont nous permettre de faire la mesure. Il y a un décalage de la fréquence du résonateur qui dépend de l'état du qubit, ainsi en lisant cette fréquence nous pouvons en déduire l'état du qubit.

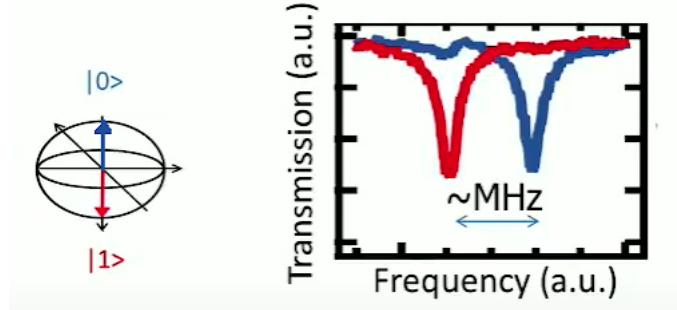


FIGURE 5 – Différence de fréquence du résonateur en fonction de l'état $|0\rangle$ ou $|1\rangle$ du qubit mesuré

Les jonctions de Josephson sont de bons candidats pour construire un ordinateur quantique pour plusieurs raisons :

- les temps de cohérence de leurs qubits sont relativement longs
- les procédés de fabrication sont connus grâce à l'industrie de la micro/nano-électronique
- technologie qui passe bien à l'échelle

Ordinateur quantique topologique (anyons) :

Microsoft a une autre approche que les circuits supraconducteurs pour réaliser des qubits, ils utilisent des quasi-particules nommées anyons pour matérialiser ce qu'on appelle des qubits "topologiques". L'idée principale est de tirer parti des propriétés de la topologie pour s'assurer que l'information stockée quantiquement reste intègre malgré l'interaction avec les autres particules environnantes. Le bruit et les petites perturbations du milieu qui d'habitude produisent des erreurs sur nos qubits ici ne perturbent pas les propriétés topologiques, autrement dit les qubits topologiques sont résistants aux erreurs causées par la décohérence. On arriverait alors à avoir des qubits beaucoup plus stables et robustes par rapport à la décohérence, et donc de réduire considérablement le nombre d'erreurs sur nos qubits. Ce faible taux d'erreur facilite aussi le passage à l'échelle pour cette technologie.

Le rôle des portes quantiques est endossé par ce qu'on appelle des "tresses", qui permettent de modifier l'état des qubits topologiques et de les intriquer. Pour ce qui est de la mesure, on y parvient par ce qu'on appelle la fusion d'anyons.

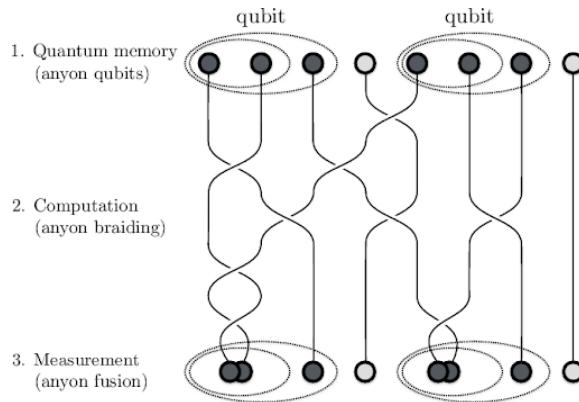


FIGURE 6 – Processus de calcul quantique avec des qubits topologiques

Ions piégés :

Il est possible de matérialiser des qubits en utilisant des ions, leur état fondamental représentant l'état $|0\rangle$ et leur état excité l'état $|1\rangle$ du qubit. Pour cela on maintient en suspension un ion (ou plusieurs) grâce à des champs magnétiques dans une cavité vide et refroidie. Le vide dans cette cavité est important puisqu'on veut éviter que des particules perturbent l'ion qui joue le rôle de notre qubit. Ayant ces ions en suspension dans une cavité, c'est avec des lasers que nous pouvons les exciter et changer leur état quantique. En appliquant des lasers sur ces particules, il est possible d'initialiser un qubit, d'appliquer une porte quantique, d'intriquer des qubits, et aussi de faire une mesure.

Les qubits à base d'ions piégés ont une bonne durée de vie et c'est sur cette technologie que les opérations fondamentales du calcul quantique ont les meilleures performances à ce jour. Cependant, l'inconvénient majeur des ions piégés réside dans le passage à l'échelle, dans une cavité nous sommes limité par le nombre d'ions que l'on peut stocker et il faut alors développer un moyen d'interconnecter plusieurs pièges à ions entre eux pour pouvoir augmenter le nombre de qubits.

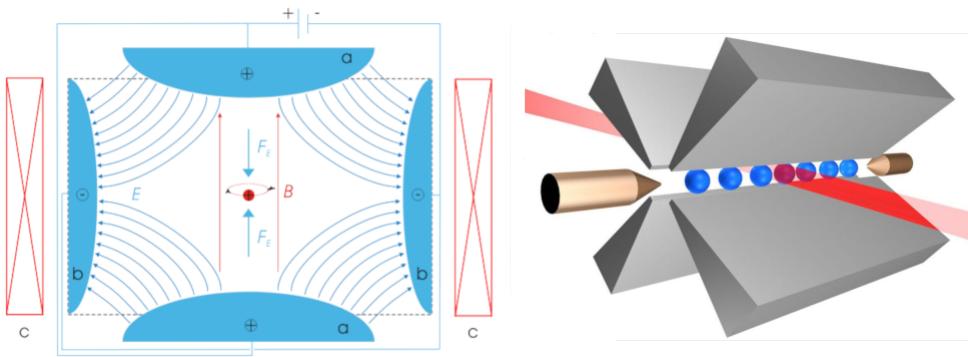


FIGURE 7 – Schémas d'un piège à ions

3 Traitement d'image et informatique quantique

3.1 Traitement d'image classique

Un des concepts les plus importants du traitement de l'image et de la vision en informatique, en particulier dans les domaines de la détection et de l'extraction de caractéristiques, est la capacité de détecter les bords des objets présents dans une image. Ce problème est appelé « Edge Detection » illustré par la figure 8.



FIGURE 8 – Résulat d'un algorithme d'edge detection

3.1.1 Représentation d'image

En informatique, une image est représentée par une fonction $f(x, y)$, où x et y représentent les coordonnées d'un pixel dans l'image. En général on représente l'image dans une matrice de colonne X (nombre de pixels horizontal) et de ligne Y (nombre de pixels vertical), aussi appelé résolution horizontale et verticale. La valeur de la fonction $f(x, y)$ en chaque point indexé par une ligne et une colonne est appelée valeur de gris ou intensité de l'image. Généralement, la valeur du pixel est la valeur d'intensité de l'image à ce point. Dans le cas d'une image en couleur, il est nécessaire de stocker 3 valeurs pour l'intensité de chaque couleur primaire. Le nombre de bit nécessaire par pixel est appelé « bit depth ». Dans notre cas ,on ne s'intéressera que aux images en noirs et blanc, ce qui est en général suffisant dans des problèmes de détection de contour.

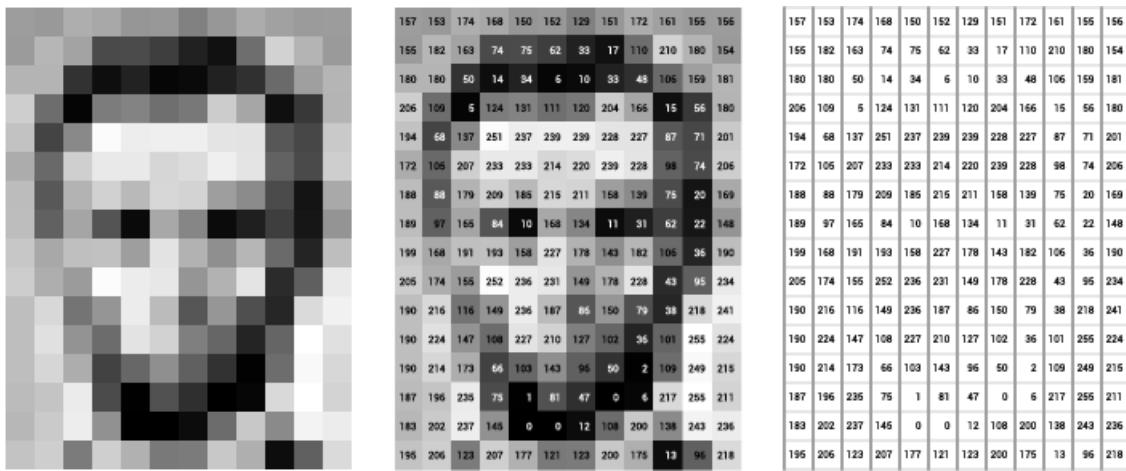


FIGURE 9 – Image et sa représentation matricielle

Il existe également de nombreuses techniques d'encodage permettant de stocker les informations

présentes dans l'image dans des formats moins lourds. Dans notre cas, il ne nous semble pas pertinent d'approfondir le sujet car les images sont toujours décodées avant d'être analysées. On expliquera dans la partie suivante les techniques de base de détection de contour en informatique classique.

3.1.2 Edge detection/Edge filter

Premièrement, il est important de définir ce que l'on entend par « bord » (« Edge »). Un bord dans une image est un changement local significatif de l'intensité de l'image, généralement associée à une discontinuité dans l'intensité de l'image ou dans la dérivée première de l'intensité de l'image. Ces changements significatifs dans les valeurs de gris d'une image peuvent être détectés en utilisant le gradient. Dans notre cas, le gradient est un vecteur dont les composants mesurent la rapidité avec laquelle les valeurs des pixels changent avec la distance dans les directions x et y . Il est possible d'approcher la valeur du gradient de manière numérique de la manière suivante : On pose G_x et G_y tels que :

$$G_x \simeq f[i, j + 1] - f[i, j] \quad \text{et} \quad G_y \simeq f[i, j] - f[i + 1, j]$$

On peut calculer G_x et G_y par le biais des matrice de convolution suivante :

$$G_x = \begin{bmatrix} 1 & -1 \end{bmatrix} \quad \text{et} \quad G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Cependant, en utilisant les approximations ci-dessus, G_x est en fait l'approximation du gradient au point interpolé ($[i, j + 1/2]$ et G_y à $[i + 1/2, j]$). Pour récupérer le gradient en chaque pixel (en non entre 2 pixels comme dans l'approximation précédente), on utilisera des matrice de convolution 3x3 qui sont définis ci-dessous :

Filtre de Sobel :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{et} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

avec $G = \sqrt{G_x^2 + G_y^2}$

Filtre de Prewitt :

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{et} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

avec $G = \sqrt{G_x^2 + G_y^2}$

Exemple de calcul avec le filtre Sobel :

$$\begin{bmatrix} 100 & 100 & 100 & 100 \\ 110 & 110 & 110 & 110 \\ 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 \end{bmatrix} \rightarrow \begin{bmatrix} 222 & 255 & 255 & 222 \\ 121 & 121 & 121 & 121 \\ 112 & 109 & 109 & 112 \\ 121 & 121 & 121 & 121 \\ 30 & 0 & 0 & 30 \end{bmatrix}$$

3.2 Représentation d'image en informatique quantique

3.2.1 Technique de représentation flexible d'images quantiques (FRQI : Flexible Representation of Quantum Images)[5]

Dans cette technique on représente une image grâce à l'état quantique suivant :

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (\cos \theta_i |0\rangle + \sin \theta_i |1\rangle) \otimes |i\rangle \quad (1.1)$$

$$\theta_i \in \left[0, \frac{\pi}{2}\right], i = 0, 1, \dots, 2^{2n} - 1 \quad (1.2)$$

Où les informations quand à l'intensité des pixels est stocké dans les θ et leurs positions dans $|i\rangle$

Exemple pour une image 2x2 :

$\theta_0, 00\rangle$	$\theta_1, 01\rangle$
$\theta_2, 10\rangle$	$\theta_3, 11\rangle$

Équivalent à l'état :

$$\begin{aligned} |I\rangle = \frac{1}{2} [& (\cos \theta_0 |0\rangle + \sin \theta_0 |1\rangle) \otimes |00\rangle \\ & + (\cos \theta_1 |0\rangle + \sin \theta_1 |1\rangle) \otimes |01\rangle \\ & + (\cos \theta_2 |0\rangle + \sin \theta_2 |1\rangle) \otimes |10\rangle \\ & + (\cos \theta_3 |0\rangle + \sin \theta_3 |1\rangle) \otimes |11\rangle] \end{aligned} \quad (1.3)$$

Si θ vaut 0, le pixel sera noir et si θ vaut $\pi/2$, le pixel sera blanc.

Construction de l'état $|I(\theta)\rangle$:

On doit commencer par mettre tous les qubits, sauf 1, dans un état de superposition. Pour cela on utilise on applique Hadmard sur chacun des qubits :

$$|H\rangle = \frac{1}{2^n} |0\rangle \otimes \sum_{i=0}^{2^{2n}-1} |i\rangle = \mathcal{H}(|0\rangle^{\otimes 2n+1}) \quad (2.1)$$

On utilise ensuite des matrices de rotation contrôlées sur le dernier qubit (celui qui n'est pas en superposition) pour stocker les informations sur l'intensité de chaque pixel. On obtient alors l'état $|I(\theta)\rangle$.

$$\mathcal{R}|H\rangle = \left(\prod_{i=0}^{2^{2n}-1} R_i \right) |H\rangle = |I(\theta)\rangle \quad (2.2)$$

Avec :

$$R_i = \left(I \otimes \sum_{j=0, j \neq i}^{2^{2n}-1} |j\rangle\langle j| \right) + R_y(2\theta_i) \otimes |i\rangle\langle i| \quad (2.3)$$

et

$$R_y(2\theta_i) = \begin{pmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{pmatrix} \quad (2.4)$$

Enfin, on utilisera l'opérateur X pour se déplacer dans l'ensemble des coordonées. A chaque coordonné, on applique la Rotation contrôlée $Ry(2\theta)$ sur le dernier qubit, avec $\theta = 0$ si l'intensité est minimal et $\theta = \pi/2$ pour une intensité maximal. Pour toute les valeur intermédiaire, il suffit de scale l'intensité entre ces 2 valeurs. Par exemple une intensité de 0.5 (si 1 est l'intensité maximal et 0 minimal) correspondra à $\theta = \pi/4$

Exemple d'implémentation de la représentation d'une image 2x2 (ici 4 pixel noire) :

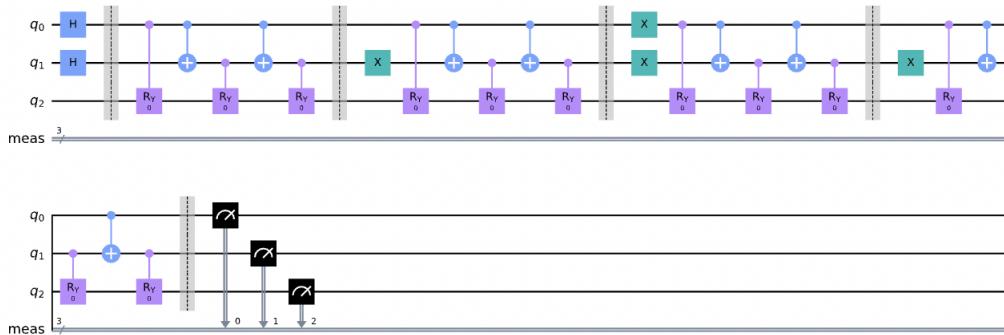


FIGURE 10 – Circuit encodant une image 2x2

Mesure et récupération de l'image :

Comme présenté précédemment, l'image est encodé à travers l'état suivant :

$$\begin{aligned} |I\rangle = \frac{1}{2} [& (\cos \theta_0 |0\rangle + \sin \theta_0 |1\rangle) \otimes |00\rangle \\ & + (\cos \theta_1 |0\rangle + \sin \theta_1 |1\rangle) \otimes |01\rangle \\ & + (\cos \theta_2 |0\rangle + \sin \theta_2 |1\rangle) \otimes |10\rangle \\ & + (\cos \theta_3 |0\rangle + \sin \theta_3 |1\rangle) \otimes |11\rangle] \end{aligned}$$

Chaque pixel est associé à une valeur potentielle des $n-1$ qubits (exemple : 00, pour le premier pixel , 01 pour le deuxième etc.) On remarque que pour pour chacune de ces valeurs le derniers

qubit est dans l'état : $\cos \theta |0\rangle + \sin \theta |1\rangle$. Comme expliqué précédemment, $\theta = \pi/2$ correspond à une intensité maximal, un $\theta = 0$ à une intensité minimal. Si on reproduit un grand nombre de fois les mesures, nous serons capable, en regardant la proportion d'état ayant comme valeur 1 et 0 au premier qubit d'estimer θ . En effet si la totalité des états correspondant à un pixel à son premier qubit à 1, alors $\theta = \pi/2$ (car $\sin \pi/2 = 1$ et $\cos \pi/2 = 0$), ou encore si l'on observe autant d'état commençant par 0 et 1 alors on peut déduire que $\theta = \pi/4$ (car $\sin \pi/4 = \cos \pi/4$). On peut donc, en suivant ce raisonnement, retrouver la valeur de tout les θ (et donc de l'intensité), quelconque, pour chacun des pixels.

Pour une image 2x2 avec les intensités suivantes (de 0 à 100)

0	50
100	100

On observe, pour 10000 simulations :

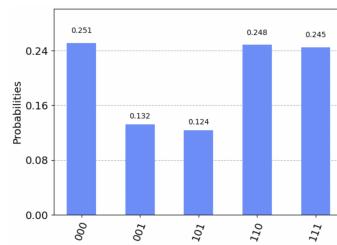


FIGURE 11 – Mesure du circuit

On implémentera cette algorithme pour des images de taille n dans la dernière partie et le testera sur différentes images.

3.2.2 Technique de représentation nouvelle quantique améliorée (Novel Enhanced Quantum Representation (NEQR) for Digital Images

La nouvelle représentation quantique améliorée est une autre des premières formes de représentation d'images quantiques. Il est similaire à la représentation flexible des images quantiques (FRQI), notamment par le fait qu'il utilise une superposition normalisée pour stocker des pixels dans une image. La limitation de FRQI est qu'il utilise un qubit pour stocker les informations en niveaux de gris du pixel, ce qui empêche d'effectuer des transformations d'image complexes. NEQR a été créé pour améliorer le FRQI en tirant parti de l'état de base d'une séquence de qubits pour stocker la valeur en niveaux de gris de l'image.

Par ailleurs, Le NEQR offre plus d'avantages que FRQI, notamment :

- Accélération quadratique de la complexité temporelle pour préparer l'image quantique NEQR.
- Taux de compression d'image optimal jusqu'à $1,5\times$.
- Précision de l'image élevée après la mesure, par opposition à probabiliste comme FRQI.
- Une couleur complexe et de nombreuses autres opérations peuvent être réalisées.

Voyons voir en pratique comment nous encodons une telle images avec avec le NEQR. Comme précisé précédemment, une image en noir et blanc est défini par des pixels de coordonnées x et y ainsi que qu'une intensité comprise entre 0 et 255. Pour le cas d'une image 2x2, on peut avoir l'encodage suivant avec deux qubits :

<i>position</i>	<i>binary string</i>	<i>grayscale intensity</i>
$ 00\rangle$	$ 00000000\rangle$	0 – Black
$ 01\rangle$	$ 01100100\rangle$	100 – Dark shade
$ 10\rangle$	$ 11001000\rangle$	200 – Light shade
$ 11\rangle$	$ 11111111\rangle$	255 – White

Ainsi, nous avons une fonction $f(X, Y)$ avec X et Y la position en ligne colonne du pixel qui encode la couleur sous forme de string binaire. Ainsi chaque pixel peut être représenté selon l'équation suivante.

$$\{(Y, X) = C_{YX}^0, C_{YX}^1, \dots, C_{YX}^{q-2}, C_{YX}^{q-1} \in [0, 1], f(Y, X) \in [0, 2^{q-1}] \quad (1)$$

La généralisation d'une image pour plusieurs pixels est donc la suivante :

$$\{(1, 0) = \overline{C_{10}^0}, C_{10}^1, C_{10}^2, \overline{C_{10}^3}, \overline{C_{10}^4}, C_{10}^5, \overline{C_{10}^6}, \overline{C_{10}^7} = 01100100 = 100 \quad (2)$$

Enfin pour la représentation de l'intensité nous obtenons la formule complexe suivante :

$$|I\rangle = \frac{1}{2^n} \sum_{X=0}^{2^{2n-1}} \sum_{Y=0}^{2^{2n-1}} |\{(X, Y)\}|XY\rangle = \frac{1}{2^n} \sum_{X=0}^{2^{2n-1}} \sum_{Y=0}^{2^{2n-1}} |\otimes_{i=0}^{q-1}\rangle |C_{XY}^i\rangle |XY\rangle \quad (3)$$

Pour revenir à notre cas d'une image 2x2, la formule suivante donne la formule suivante :

$$\Omega_{XY}|0\rangle^{\otimes q} = \frac{1}{\sqrt{2}}(|00000000\rangle|00\rangle + |01100100\rangle|01\rangle + |11001000\rangle|10\rangle + |11111111\rangle|11\rangle) \quad (4)$$

Avec $\Omega_{XY}|0\rangle$, l'opérateur quantique qui se charge d'associer la valeur à la position X, Y . Concernant le circuit associé à un encodage 2x2 il est représenté par la figure ci-dessous :

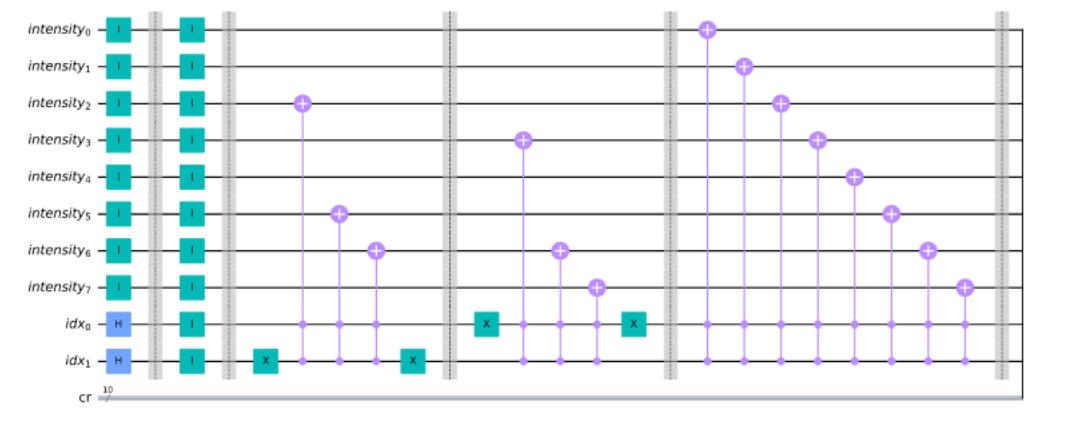


FIGURE 12 – Représentation du circuit NEQR pour une image 2x2

Nous verrons dans la dernière partie, une application et le résultat avec la NEQR.

3.2.3 La technique du codage d'image quantique par probabilité quantique (QPIE : the Quantum Probability Image Encoding)

La représentation QPIE est proche de celles présentées précédemment car elle utilise les amplitudes de probabilité d'un état quantique pour stocker les valeurs des pixels d'une image classique.

Ainsi, si nous avons n qubits, nous avons accès à un maximum de 2^n états en superposition. Dans QPIE, nous tirons parti de ce fait pour concevoir un schéma de codage efficace et robuste pour les images en noir et blanc (B&W) ou RVB et réduire de façon exponentielle la mémoire requise pour stocker les données. Cela signifie que pour stocker une image de 4 pixels, nous n'avons besoin que de 2 qubits ; pour une image de 8 pixels, nous avons besoin de 3 qubits, et ainsi de suite. En général, le nombre de qubits (n) pour une image de N pixels est calculée comme suit :

$$n = \lceil \log_2 N \rceil \quad (5)$$

Processus de passage d'une image à l'état classique à une image à l'état quantique via QPIE. Prenons un exemple d'image avec quatre pixels qui est disposé en 2D comme suit :

00 I_0	01 I_1
10 I_2	11 I_3

FIGURE 13 – représentation classique d'une image à 4 pixels

Ici, le vecteur I_0, I_1, I_2, I_3 ou $(I_{00}, I_{01}, I_{10}, I_{11})$ en représentation binaire des indices de sous-script) représente les intensités de couleur (en couleur 8 bits B&W) de différents pixels (00 , 01 , 10 , 11) représentés sous forme de matrice 2D pour former une image classique 2×2 . L'image peut être représentée en termes d'intensité de ses pixels comme suit :

$$I = (I_{yx})_{N_1 \times N_2} \quad (6)$$

Par conséquent, (6) représente une image bidimensionnelle composée de $N_1 \times N_2$ pixels, où I_{yx} est l'intensité du pixel à la position (x, y) dans l'image 2D en partant des axes de coordonnées du coin supérieur gauche de l'image.

Nous devons maintenant représenter ces intensités de pixel comme les amplitudes de probabilité d'un état quantique particulier. Pour ce faire, les intensités des pixels doivent être normalisées de manière à ce que la somme des carrés de toutes les amplitudes de probabilité soit égale à 1. Pour chaque c_i correspondant à un I_{yx} respectif, la normalisation peut être effectuée comme suit :

$$c_i = \frac{I_{yx}}{\sqrt{\sum I_{yx}^2}} \quad (7)$$

Après la normalisation, l'image quantique se présente comme suit :

00 c_0	01 c_1
10 c_2	11 c_3

FIGURE 14 – représentation quantique d'une image à 4 pixels via la méthode QPIE

Enfin, en affectant les valeurs normalisées de la couleur de chaque pixel P_i à l'état quantique respectif $|i\rangle$, on peut écrire l'état de l'image $|img\rangle$ comme suit

$$|Img\rangle = c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle \quad (8)$$

Ou en généralisant pour n -qubits, on a :

$$|Img\rangle = \sum_{i=0}^{2^n-1} c_i|i\rangle \quad (9)$$

Un tel état peut être préparé très efficacement en utilisant quelques portes de rotation et de CNOT comme on a pu le voir en cours ou dans les références [3] et [7].

Voici un exemple de circuit avec l'histogramme associé pour l'image de 4 pixels avec des valeurs de pixels B&W : (0, 128, 192, 255). Voir figure 15

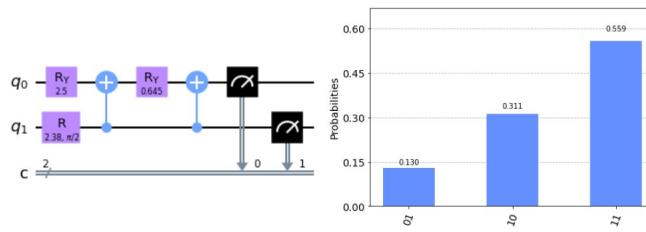


FIGURE 15 – Exemple de circuit avec l'histogramme associé pour l'image de 4 pixels

Cet exemple fut réalisé à l'aide de qiskit et vous pourrez trouver l'implémentation sur notre [git](#)

3.3 Détection de contours en informatique quantique

La détection des contours est un élément essentiel de toute procédure d'extraction de caractéristiques d'image. Le processus de détection des bords est largement utilisé dans les algorithmes classiques modernes de traitement d'images (comme vu précédemment) pour extraire la structure des objets/caractéristiques représentés dans une image. Le traitement d'image quantique étant un domaine émergent, il est très intrigant et permet d'obtenir une accélération exponentielle (comme mentionné dans l'article de Ruan et al. [6]), dans certains cas, par rapport au traitement d'image classique.

Bien que la détection des contours soit assez efficace dans le traitement d'image classique, elle devient très lente pour les grandes images en raison de la résolution énorme de ces images et du calcul pixel par pixel qui est nécessaire pour la plupart des algorithmes classiques de détection des contours. D'autre part, nous venons de voir dans la sous partie précédente comment on peut convertir des images classiques en images quantiques en utilisant les représentations quantiques d'images aussi appelé QImR :

- La technique de la représentation flexible d'images quantiques (FRQI : Flexible Representation of Quantum Images)
- La technique de la représentation quantique améliorée (NEQR : Novel Enhanced Quantum Representation)
- La technique du codage d'image quantique par probabilité quantique (QPIE : the Quantum Probability Image Encoding)

Ici nous allons nous intéresser à l'extension de l'utilisation de ces QImRs et plus particulièrement QPIE pour effectuer de la détection de contours à l'aide de l'algorithme quantique de détection de bords basé sur l'utilisation de la porte d'Hadamard (QHED algorithme : Quantum Hadamard Edge Detection)[8]).

3.3.1 Principe de l'algorithme

On rappelle que la porte d'Hadamard : H ou H-gate opère de la façon suivante sur l'état du qubit :

$$|0\rangle \rightarrow \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \quad (10)$$

$$|1\rangle \rightarrow \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \quad (11)$$

L'algorithme QHED généralise cette action de la porte H et l'utilise pour la détection des bords d'une image. Supposons que nous ayons une image de n pixels. Les pixels de l'image peuvent être numérotés en utilisant des chaînes de bits binaires de la forme $|b_{n-1}b_{n-2}b_{n-3}...b_1b_0\rangle$ où $b_i \in \{0, 1\}$. Pour deux pixels voisins, les chaînes de bits peuvent être écrites sous la forme $|b_{n-1}b_{n-2}...b_10\rangle$ et $|b_{n-1}b_{n-2}...b_11\rangle$, c'est-à-dire que seul le bit le moins significatif (LSB : least significant bit) est différent pour les deux. Les valeurs d'intensité des pixels correspondants (normalisées) peuvent être écrites respectivement comme $c_{b_{n-1}b_{n-2}...b_10}$ et $c_{b_{n-1}b_{n-2}...b_11}$. Pour simplifier la notation, nous utiliserons la représentation décimale des chaînes de bits. Ainsi, les valeurs des pixels peuvent être écrites comme c_i et c_{i+1} en représentation décimale. Maintenant, si nous appliquons la porte H au LSB d'un registre quantique de taille arbitraire, nous pouvons représenter l'unité résultante comme suit :

$$I_{2^{n-1}} \otimes H_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix} \quad (12)$$

En appliquant cet opérateur unitaire à un registre quantique contenant des valeurs de pixels codées à l'aide de la représentation QPIE : $|\text{Img}\rangle = \sum_{i=0}^{N-1} c_i|i\rangle$ comme indiqué dans l'éq : 5 on a :

$$(I_{2^{n-1}} \otimes H_0) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix} \rightarrow \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \end{bmatrix} \quad (13)$$

A partir de la matrice résultante ci-dessus (13), il est clairement visible que nous avons maintenant accès au gradient entre les intensités des pixels voisins sous la forme de $(c_i - c_{i+1})$ où, i est pair. En mesurant le circuit conditionné à ce que le LSB soit dans l'état $|1\rangle$, nous pouvons obtenir les gradients par analyse statistique. Ce processus aboutit à la détection des frontières horizontales entre

les paires de pixels pairs (0 & 1 , 2 & 3 , et ainsi de suite). Pour détecter les frontières horizontales entre les paires de pixels impairs (1 & 2 , 3 & 4 , etc.), nous pouvons effectuer une permutation d'amplitude sur le registre quantique pour convertir le vecteur d'amplitude $(c_0, c_1, c_2, \dots, c_{N-1})^T$ en $(c_1, c_2, c_3, \dots, c_{N-1}, c_0)^T$, puis appliquer la porte H et mesurer le registre quantique à condition que LSB soit $|1\rangle$. Cependant, nous pouvons le rendre plus efficace en termes de ressources en utilisant un qubit auxiliaire supplémentaire :

3.3.2 Variation de l'algorithme QHED (Utilisation d'un qubit auxiliaire)

Comme nous l'avons vu dans la sous-section précédente, nous avons toujours un registre quantique avec n qubits ($n = \lceil \log_2 N \rceil$) pour coder l'image à N pixels. Cependant, dans ce cas, nous ajoutons un qubit auxiliaire supplémentaire au registre que nous pouvons utiliser pour étendre l'algorithme QHED et effectuer des calculs sur les paires de pixels pairs et impairs simultanément. Comme la dernière fois, nous nous initialisons à l'état $|Img\rangle = (c_0, c_1, c_2, \dots, c_{N-2}, c_{N-1})^T$. Cependant, la porte H est maintenant appliquée au qubit auxiliaire qui est initialisé à l'état $|0\rangle$. Cela produit un état d'image redondant à $(n + 1)$ qubits qui peut être représenté comme :

$$|Img\rangle \otimes \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} c_0 \\ c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ \vdots \\ c_{N-2} \\ c_{N-2} \\ c_{N-1} \\ c_{N-1} \end{bmatrix} \quad (14)$$

Maintenant, puisque nous obtenons les amplitudes de probabilité redondantes obtenues dans l'état résultant dans (14) , nous pouvons définir un opérateur de permutation d'amplitude comme suit pour transformer les amplitudes en une structure qui facilitera le calcul des gradients d'image plus loin :

$$D_{2^{n+1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \quad (15)$$

L'opérateur unitaire ci-dessus correspond à une porte de décrémentation. Par conséquent, nous pouvons efficacement décomposer cette unité en un ensemble de rotations X à contrôle unique ou multiple sur un registre de qubits multiples, comme l'ont montré Fijany et Williams dans [1] et Gidney dans [2]. En appliquant l'opérateur unitaire de décrémentation : $D_{2^{n+1}}$ ci-dessus à l'état d'image redondant, nous pouvons transformer l'état $(c_0, c_0, c_1, c_1, c_2, c_2, \dots, c_{N-2}, c_{N-2}, c_{N-1}, c_{N-1})^T$ en le nouvel état d'image redondant $(c_0, c_1, c_1, c_2, c_2, c_3, \dots, c_{N-2}, c_{N-1}, c_{N-1}, c_0)^T$. Maintenant, si nous

appliquons à nouveau la porte H au qubit auxiliaire, nous obtenons les gradients pour les paires de pixels pairs et impairs en même temps, comme suit :

$$(I_{2^n} \otimes H) \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_1 \\ c_2 \\ c_2 \\ c_3 \\ \vdots \\ c_{N-2} \\ c_{N-1} \\ c_{N-1} \\ c_0 \end{bmatrix} \rightarrow \begin{bmatrix} c_0 + c_1 \\ c_0 - c_1 \\ c_1 + c_2 \\ c_1 - c_2 \\ c_2 + c_3 \\ c_2 - c_3 \\ \vdots \\ c_{N-2} + c_{N-1} \\ c_{N-2} - c_{N-1} \\ c_{N-1} + c_0 \\ c_{N-1} - c_0 \end{bmatrix} \quad (16)$$

Enfin, en mesurant à de multiple reprise cet état à condition que le qubit auxiliaire (le LSB dans notre cas) soit dans l'état $|1\rangle$, on obtiendra les valeurs de gradient horizontal résultantes (les amplitudes $C_i - C_{i+1}$) pour toutes les paires possibles de qubits adjacents. En effet, comme nous savons que le LSB dans une chaîne de bits (représentant une ligne du vecteur d'état résultant dans 16) est 1 seulement pour les lignes impairs, nous pouvons facilement prendre les amplitudes correspondant aux états impairs du vecteur d'état pour former notre image à bords détectés et rejeter tous les états pairs.

Tout le processus ci-dessus fournit un balayage horizontal de l'image entière dont les bords sont détectés uniquement dans la direction horizontale. Pour obtenir l'image à bords détectés par balayage vertical, nous prenons la transposition de la matrice de l'image et suivons à nouveau le même processus pour obtenir un balayage vertical. Ces balayages : horizontal et vertical sont ensuite superposés l'un sur l'autre à l'aide d'un post-traitement classique pour créer l'image complète des bords détectés.

3.3.3 Circuit Quantique

Prenons un échantillon d'image 44, aplati et représenté comme le vecteur :

$$(0, 0.9, 0, 0, 0.5, 0.6, 0.3, 0, 0, 0.2, 0.7, 0.8, 0, 0, 1, 0)$$

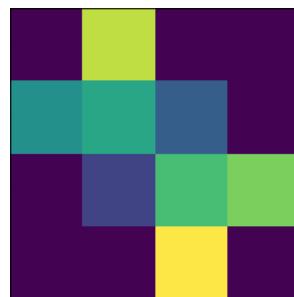


FIGURE 16 – échantillon d'image 4x4

Le circuit quantique QHED pour l'image ici peut être généralisé comme suit :

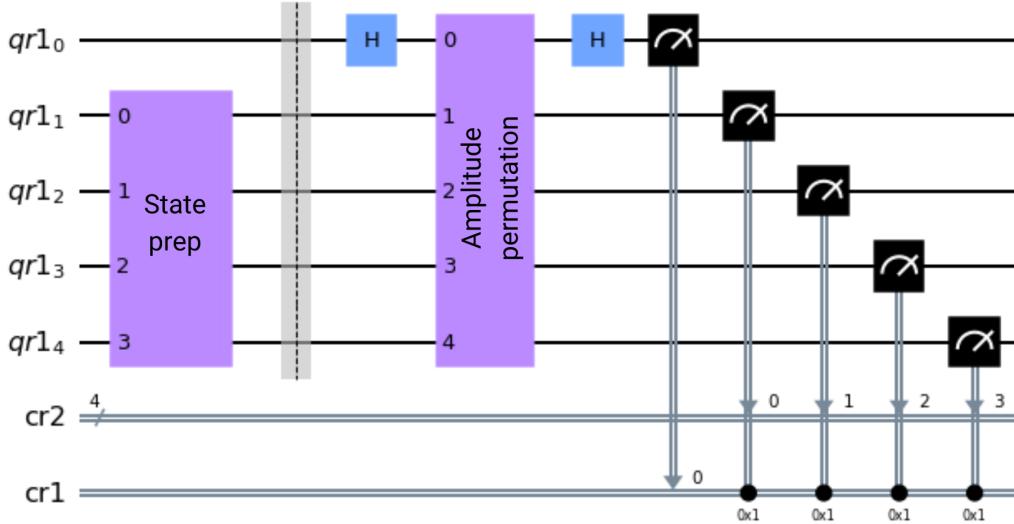


FIGURE 17 – circuit quantique correspondant à l’application de QHED sur l’image de la figure précédente pour l’obtention d’un scan horizontal

Précision : Les mesures des qubits de données ($qr_{11}, qr_{12}, qr_{13}$, et qr_{14}) dépendent du résultat de la mesure du qubit auxiliaire (qr_{10}) qui est dans l’état $|1\rangle$.

3.3.4 Analyse de la complexité temporelle et spatiale de QHED

Nous avons brièvement fait des recherches sur la complexité temporelle des algorithmes classiques de détection des contours et avons déterminé qu’elle est de l’ordre de $O(2^n)$ (image à n éléments/pixels) dans le pire des cas et de $O(mn \cdot \log(mn))$ (image $m \times n$ pixels) pour certaines des techniques classiques de détection des contours les plus améliorées [4].

D’autre part, en ce qui concerne les techniques quantiques de détection des bords, l’algorithme QSobel est beaucoup plus rapide et utilise la représentation d’image FRQI pour coder une image de $(N \times N)$ pixels ($N = 2^n$, dans un système à n -qubits) [10]. Cependant, la représentation d’image FRQI a un processus de préparation d’état complexe ($[O(n) + O(\log^2 n)]$ de profondeur de circuit dans le pire des cas) et nécessite plus de qubits ($[1 + 2N]$ -qubits) pour stocker les données d’image [5], ce qui est une ressource limitée dans le matériel actuel. De plus, QSobel souffre également de problèmes d’implémentation efficace de certaines sous-routines intermédiaires (comme COPY et la fonction boîte noire pour le calcul du gradient) au sein de l’algorithme [8].

L’algorithme QHED qui est utilisé ici, possède un schéma de codage d’image plus efficace en termes d’espace (QPIE) qui utilise le codage d’amplitude conduisant à une diminution exponentielle du nombre de qubits utilisés (seulement $(n = \lceil \log_2 N \rceil)$ -qubits). Cependant, la complexité temporelle de l’étape de préparation de l’état pour le codage d’images à l’aide de QPIE est légèrement supérieure à $O(n^2)$ [9], par rapport à FRQI. De plus, l’implémentation la plus efficace de la porte de décrémentation a une profondeur de circuit de $O[\text{poly}(n)]$. Mais, comme QHED utilise intelligemment la propriété de la porte d’Hadamard, nous sommes en mesure d’obtenir une complexité temporelle de $O(1)$ pour la procédure de détection des bords (sans inclure la préparation des états et la permutation des amplitudes). Cette complexité est bien inférieure à la complexité $O(n^2)$ requise

pour l'algorithme QSobel. Par conséquent, l'algorithme QHED nous donne une accélération super-exponentielle par rapport aux algorithmes classiques et une accélération polynomiale par rapport à l'algorithme QSobel.

Précision : un autre aspect sur lequel nous devrions nous concentrer pour faire fonctionner cet algorithme quantique est le nombre de mesures à effectuer pour obtenir une précision considérable pour l'algorithme. En général, pour un circuit à n qubits, il faut $O(2^n)$ mesures pour obtenir une bonne précision des probabilités de sortie. Cependant, si le but est simplement de découvrir des motifs spécifiques dans l'image, nous pouvons effectuer la mesure d'une seule observable locale avec un nombre de mesures de l'ordre de $O(n^2)$ [8]. Toutefois, cette limitation n'est pas le fait de l'algorithme, mais une caractéristique de la nature quantique inhérente du système sur lequel l'algorithme est exécuté. Par conséquent, les algorithmes quantiques et classiques de détection des contours ne sont pas directement comparables sur la seule base des limites de complexité temporelle.

4 Application et tests

4.1 Technique de représentation flexible des images quantiques (FRQI)

On implémentera l'algorithme pour créer le circuit FRQI représentant une image de taille $2^n \times 2^n$ ainsi que l'algorithme permettant de restituer l'image à partir de X mesure de ce circuit.

Création du circuit :

1. On commence par créer un circuit de taille $2n + 1$.
 2. On applique l'opérateur d'Hadamard sur les $2n$ premiers qubits.
 3. On associe chaque coordonné de la matrice d'intensité de l'image à encodé à un état possible des $2n$ premiers qubits.

Exemple pour $n = 1$: $(0, 0) : '00'$, $(0, 1) : '01'$, $(1, 0) : '10'$, $(1, 1) : '11'$

On applique des opérateurs X sur les $2n$ premiers qubit, pour itérer dans les différents états correspondant aux coordonnées.

On applique ensuite l'opérateur de rotation contrôlé correspondant à l'intensité de chaque pixel scale entre 0 et $\pi/2$: dans l'exemple suivant, 0000 = (0,0) avec une intensité de 0, 0001 = (0,1) avec une intensité de $\pi/4$ et 0010 = (0,2) avec une intensité de $\pi/2$

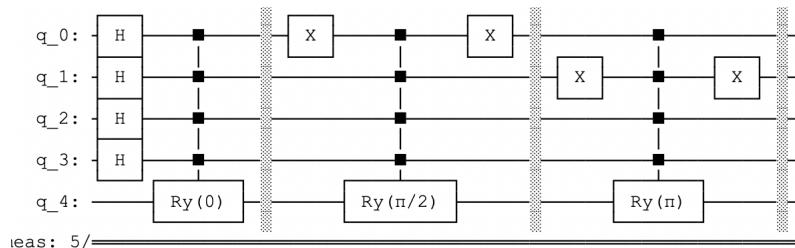


FIGURE 18 – Circuit

4. On mesure l'ensemble des qubits

Récuperation de l'image :

Pour récupérer l'image, on simule X fois (pour $n = 1$, 1000, pour $n = 2$, 10000, pour $n = 3$, 100000, pour $n = 4$, 1000000). On récupère le nombre de fois que chaque état a été mesuré. Les $2n$ premiers qubits permettent d'identifier chaque pixel. On récupère l'intensité de chaque pixel, en ajoutant le nombre d'observation si le dernier qubit est égal à 1, et en soustrayant si le dernier qubit est égal à 0. On scale ensuite ces valeurs dans l'intervalle dans lequel l'image est initialement encodé. On peut voir une illustration dans l'exemple suivant qui s'agit d'une image à encoder :

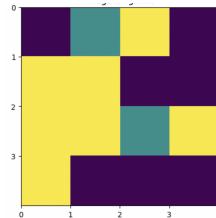


FIGURE 19 – Image

Circuit coorespondant :

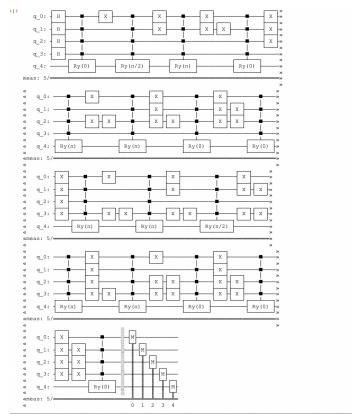


FIGURE 20 – Circuit

Mesure :

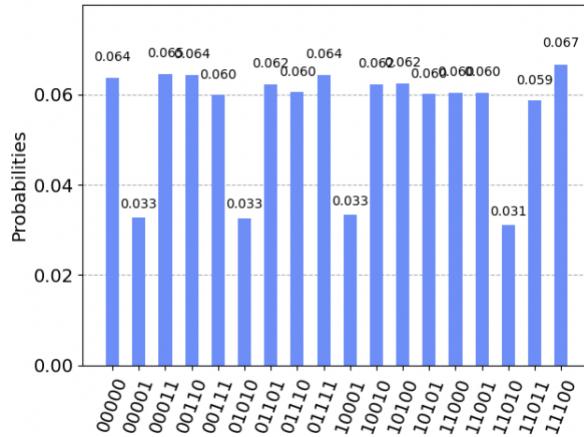


FIGURE 21 – Histogramme

Image récupérée :

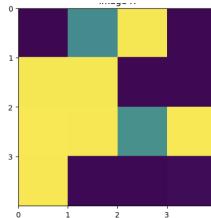


FIGURE 22 – Image récupérée

Des tests ont été effectués pour des images 16x16 (voir figures suivantes), au-delà de cela, le temps de compilation commence à devenir très important.

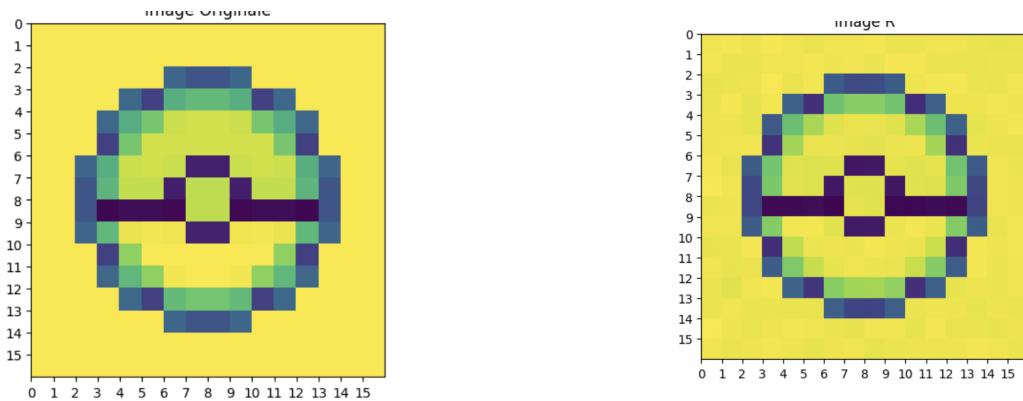


FIGURE 23 – Représentation FRQI d'une image 16x16

Pour plus d'information, se référer au code.

4.2 Application des nouvelles techniques de représentation quantique améliorée (NEQR)

Nous suivons la pipeline suivante :

1. Préparation de l'image :

Cette phase consiste simplement à préparer l'image afin d'y effectuer par la suite les traitements nécessaires. Je prends une photo aléatoire puis je la transforme en noir et blanc. Enfin je découpe l'image de sorte à avoir un carré et enfin je réduis la résolution à une image de 32x32 car nous avons trop de points pour simuler.

2. Application du FRQI :

Nous créons notre circuit quantique et l'initialisons en combinant à la fois le circuit de position de pixel avec sa valeur d'intensité de pixel respective. En suivant le process expliqué dans le tutoriel sur Qiskit, nous utilisons des CNOT afin de contrôler les qubits.

3. Premiers résultats :

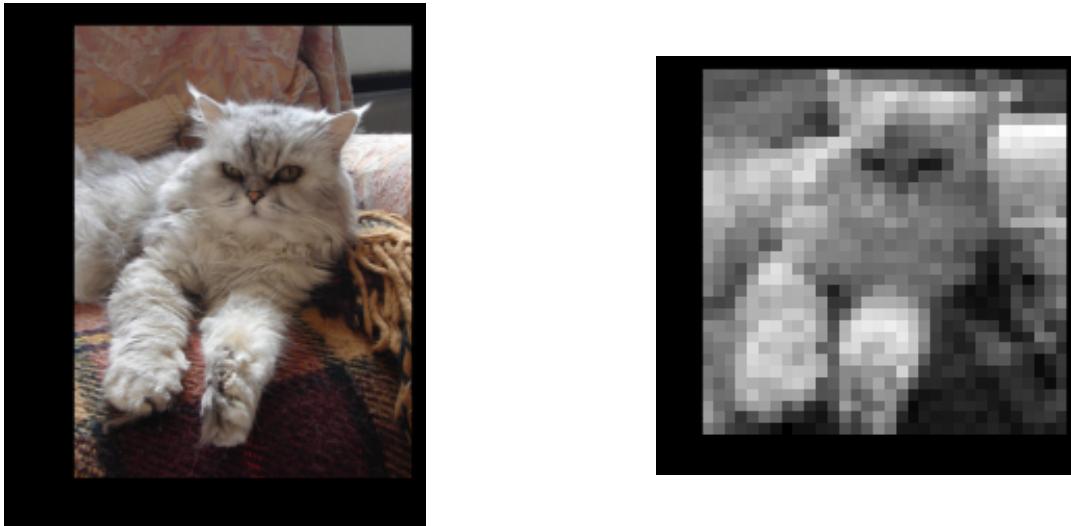


FIGURE 24 – Représentation NEQR d'une image 32x32

Nous évaluons également l'ensemble des états. Cependant ayant beaucoup de représentation pour une image 32x32 celle-ci sur la figure est saturé est donc les résultats ne sont pas clairement visible. Plus de détails sont disponible dans le git du projet.

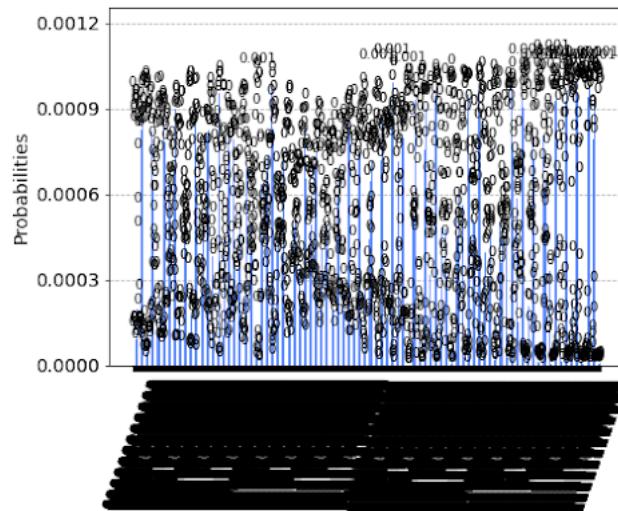


FIGURE 25 – Résultat final

4.3 Utilisations des QImR pour effectuer la détection des contours à l'aide de l'algorithme Quantum Hadamard Edge Detection(QHED)

4.3.1 Implémentation et application de l'algorithme QHED sur une image simplifiée

Dans cette partie, l'objectif est d'implémenter l'algorithme QHED au travers de 2 démonstrations en nous appuyant sur les parties 3.2.3 et 3.3 de la présentation.

1. Initialisation : Implémentation de l'image simplifiée ([voir code](#))

Dans un premier temps émettons les hypothèses suivant vis à vis de l'image sur laquelle nous allons appliquer QHED : Si nous supposons que l'image consiste en une collection de valeurs de pixels représentée sous forme de matrice numpy en python. Et si de plus, les pixels de cette images soient représentés par des valeurs binaires pour plus de simplicité, tel que $I_{jk} \in \{0, 1\}$, et donc il n'y a pas de valeurs à virgule flottante pour les intensités des pixels.

Cela énoncé, il est alors facile d'implémenter une telle image en python :

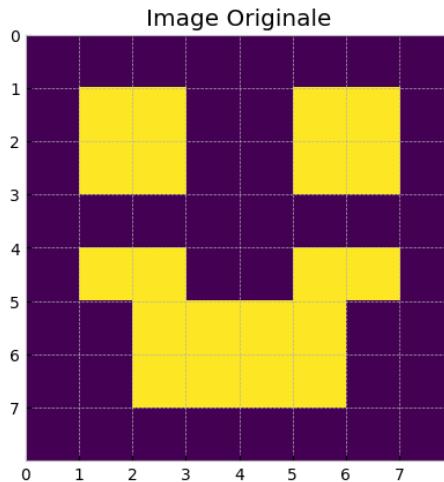


FIGURE 26 – Image Originale

2. Transformation de notre image classique en une image quantique via la méthode QPIE

Maintenant que nous avons défini notre image à tester, nous pouvons utiliser l'équation : [7](#) pour coder les intensités des pixels comme des amplitudes de probabilité des différents états du système via la méthode QPIE : ([voir code](#)) Nous obtenons alors deux images quantiques codées en amplitude différentes. La première `image_norm_h` est pour le balayage horizontal de l'image et la seconde `image_norm_v` est pour le balayage vertical de l'image.

3. Mise en place de l'algorithme QHED

Après cela, nous pouvons débuter le traitement de l'image en initialisant le nombre de qubits nécessaires et l'opérateur de permutation d'amplitude D_{2n+1} ([voir code](#)). Nous pouvons ainsi passer à présent à la fabrication du circuit quantique correspondant. Comme notre image représente maintenant essentiellement des amplitudes de différents états quantiques, nous pouvons utiliser directement la méthode `initialize()` pour effectuer la préparation de l'état.

Après cela, nous ajoutons une porte d'Hadamard au qubit auxiliaire, puis D_{2n+1} , et enfin, à nouveau, une porte d'Hadamard au qubit auxiliaire ([voir code](#)).

précision : l'ensemble du circuit est répété une fois de plus pour le balayage vertical de l'image.
Nous obtenons pour le balayage horizontal et pour le balayage verticale les résultats suivants :



FIGURE 27 – Résultat du balayage horizontal à gauche et résultat du balayage vertical à droite

4. Simulation : ([voir code](#))

Enfin, nous pouvons simuler les circuits en utilisant le simulateur de vecteur d'état et nous obtenons le vecteur d'état du système comme sortie.

D'après l'équation : [16](#) , nous pouvons clairement voir que nous devons considérer seulement les états où le qubit auxiliaire (qubit-0 ou LSB dans notre cas) donne une sortie de mesure de $|1\rangle$. Comme nous savons que le LSB dans une chaîne de bits (représentant une ligne du vecteur d'état final résultant dans [16](#)) est 1 seulement pour les lignes impaires, nous pouvons facilement prendre les amplitudes correspondant aux états impairs du vecteur d'état pour former notre image à bords détectés et rejeter tous les états pairs.

Le code implémenté pour cette tâche effectue un post-traitement classique pour s'assurer que nous obtenons les meilleurs résultats lorsque nous traçons notre image.

Après avoir filtré les états requis à partir du vecteur d'état brut, nous pouvons réorganiser le tableau 1D des amplitudes en une matrice 2D pour obtenir nos balayages (scans) : le scan horizontal et le scan vertical, comme suit :

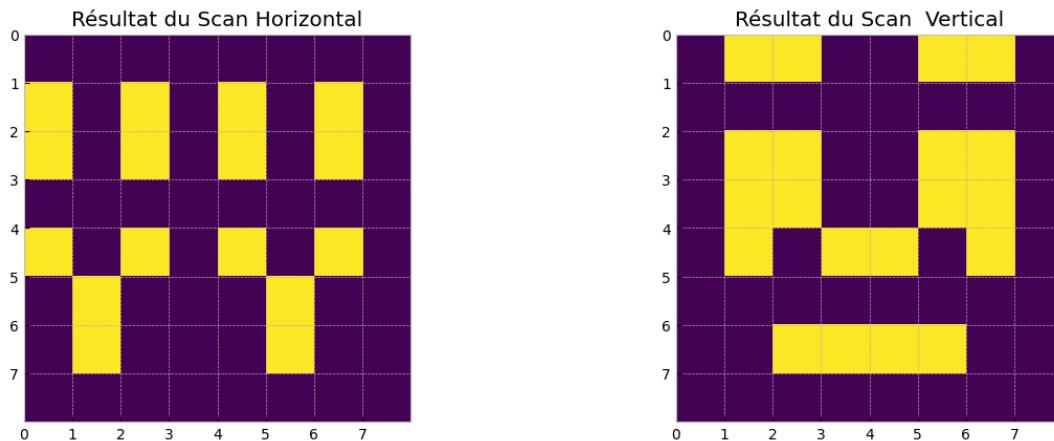


FIGURE 28 – Résultat du scan horizontal à gauche et résultat du scan vertical à droite

Enfin, nous combinons les balayages horizontaux et verticaux pour obtenir l'image complète de détection des bords et nous obtenons :

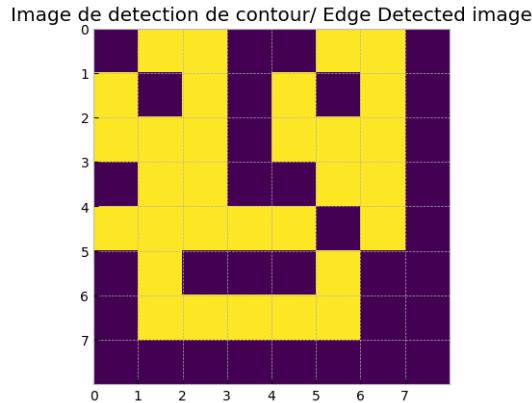


FIGURE 29 – Résultat final

4.3.2 Utilisation de l'algorithme QHED implémenté sur une image réelle

Étant donné que nous savons à présent comment réaliser la détection de contour sur une image simplifiée 8x8 px, passons à présent à quelque chose de plus concret en travaillant sur une image réelle 256x256px. Pour cela :

1. Nous récupérons une image que nous re-dimensionnons à la taille 256x256 px
2. Nous convertissons la composante RGB de l'image en image N&B, sous forme de tableau numpy (uint8)
3. Nous décomposons l'image en 64 images de tailles 32x32 px.
4. Pour chacune des images obtenues :
 - (a) Nous appliquons le protocole de détection de contour quantique réalisé précédemment
 - (b) Une fois l'image traité nous ajoutons à l'image "résultat" : `final_image()` qui représentera le résultat du traitement complet de l'image originale
5. Affichage de l'image "résultat".

Notre [programme](#) suit ces étapes et est entièrement commenté. Pour plus de compréhension nous vous invitons à le lire. Voici des exemples de ce que peut donner notre programme :

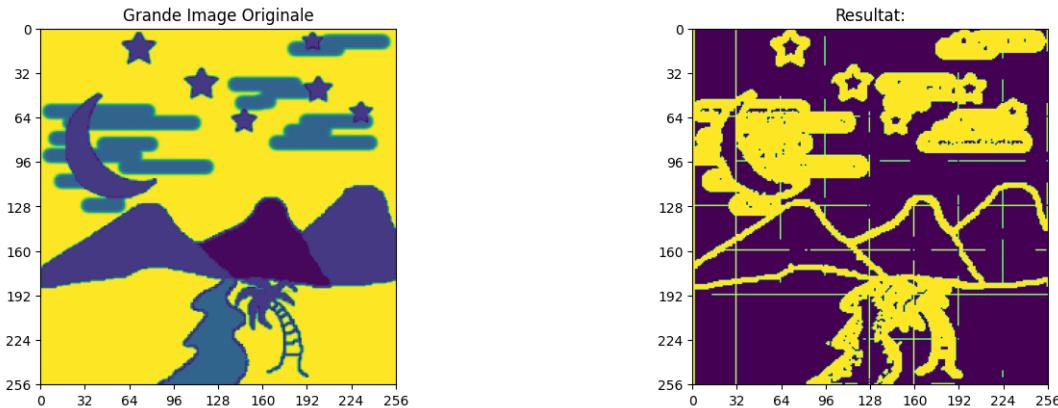


FIGURE 30 – Détection de contour d'une image de paysage

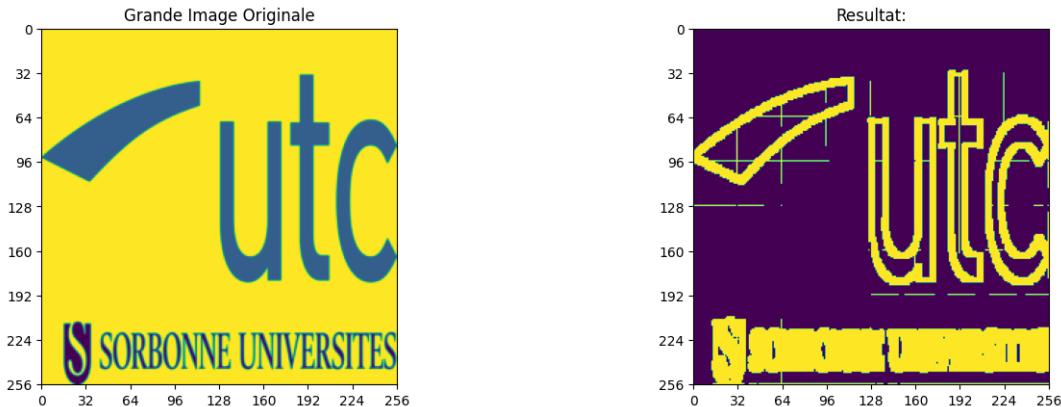


FIGURE 31 – Detection de contour du logo de l'UTC

4.3.3 Conclusion et Reflexion sur nos applications concernant QHED

L’implémentation de l’algorithme QHED était vraiment prenante et passionnante, et nous a permis d’obtenir des résultats très intéressants pour la détection d’image. Néanmoins notre travail à ses limites. Tout d’abord, nous n’avons pas vraiment pu tester nos algorithmes sur des machines réelles d’IBM. En effet, étant donné que l’exécution sur du matériel réel entraîne des erreurs dues au bruit, nous étions limité à faire des tests sûrs ($2+1$) qubits pour le traitement d’images à 4 pixels et nos résultats n’étaient pas du tout concluants. Ensuite pour notre dernière application sur images "réelles", ces dernières sont limitées à 256×256 pixels et prennent déjà un temps considérable à traiter via les modules de simulation quantiques. Ensuite, le traitement de ces images consistant à diviser le travail en traitement de 16 sous-images de 32×32 pixels, les images avec trop de détail deviennent impossibles à traiter et donne des mauvais résultat (on peut notamment le constater avec l’image du logo de l’UTC). Néanmoins nous restons vraiment contents de notre travail car lors des premiers essayés nous étions loin des résultats présentés qui nous paraissent déjà très satisfaisants.

5 Conclusion

Après avoir introduit le fonctionnement d'un ordinateur quantique et établi comment ce dernier fonctionne aujourd'hui au niveau physique, nous avons pu présenter et développer une des applications de l'informatique quantique : le traitement d'images quantiques et plus particulièrement la détection es bords d'image quantique. Pour présenter un tel traitement, nous avons d'abord expliqué mais aussi présenté l'application pratique de différentes méthodes de représentation d'image classique en image quantique. Nous avons vu que ces représentations : FRQI, NEQR et QPIE permettent un traitement quantique d'images (à la base classique). Une fois cela bien établie nous avons pu passer à l'explication de la détection de contour d'image quantique consistant principalement à exploiter la représentation QPIE pour appliquer un algorithme quantique d'"edge detection" : QHED Algoritm. Cet algorithme est basé sur la généralisation de l'action de la porte d'Hadamard sur le circuit quantique d'une image représenté quantiquement pour la détection des bords de cette image. Nous avons vu que cet algorithme offre en théorie une accélération exponentielle dans la détection de contour d'image, par rapport à celle d'image classique. Après implémentation, nous avons été globalement très satisfaits de nos résultats d'application et de tests de cet algorithme sur nos machines. Néanmoins comme nous l'avons vu, notre travail a certaines limites et ne peut fonctionner correctement sur une machine quantique réelle. Cela s'explique finalement par ce qui fut dit en première partie mentionnant qu'aujourd'hui les problèmes de stabilité, de décohérence, de tolérance aux fautes et du passage à grande échelle sur machine réelle sont encore trop importants pour permettre une application stable et concrète de l'algorithme quantique tel que le QHED Algorithm.

Références

- [1] Amir FIJANY et Colin P. WILLIAMS. "Quantum Wavelet Transforms: Fast Algorithms and Complete Circuits". In : *arXiv* (sept. 1998). eprint : [quant-ph/9809004](https://arxiv.org/abs/quant-ph/9809004). URL : <https://arxiv.org/abs/quant-ph/9809004v1>.
- [2] Craig GIDNEY. *Constructing Large Increment Gates*. [Online; accessed 22. Dec. 2021]. Août 2020. URL : <https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html>.
- [3] Lov GROVER et Terry RUDOLPH. "Creating superpositions that correspond to efficiently integrable probability distributions". In : *arXiv* (août 2002). eprint : [quant-ph/0208112](https://arxiv.org/abs/quant-ph/0208112). URL : <https://arxiv.org/abs/quant-ph/0208112v1>.
- [4] S. K. KATIYAR et P. V. ARUN. "Comparative analysis of common edge detection techniques in context of object extraction". In : *arXiv* (fév. 2014). eprint : [1405.6132](https://arxiv.org/abs/1405.6132). URL : <https://arxiv.org/abs/1405.6132v1>.
- [5] Phuc Q. LE, Fangyan DONG et Kaoru HIROTA. "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations". In : *Quantum Inf. Process.* 10.1 (fév. 2011), p. 63-84. ISSN : 1573-1332. DOI : [10.1007/s11128-010-0177-y](https://doi.org/10.1007/s11128-010-0177-y).
- [6] Yue RUAN, Xiling XUE et Yuanxia SHEN. "Quantum Image Processing: Opportunities and Challenges". In : *Math. Prob. Eng.* 2021 (jan. 2021), p. 6671613. ISSN : 1024-123X. DOI : [10.1155/2021/6671613](https://doi.org/10.1155/2021/6671613).
- [7] Andrei N. SOKLAKOV et Ruediger SCHACK. "Efficient state preparation for a register of quantum bits". In : *arXiv* (août 2004). eprint : [quant-ph/0408045](https://arxiv.org/abs/quant-ph/0408045). URL : <https://arxiv.org/abs/quant-ph/0408045v2>.
- [8] Xi-Wei YAO et al. "Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment". In : *arXiv* (jan. 2018). DOI : [10.1103/PhysRevX.7.031041](https://doi.org/10.1103/PhysRevX.7.031041). eprint : [1801.01465](https://arxiv.org/abs/1801.01465).
- [9] Xiao-Ming ZHANG, Man-Hong YUNG et Xiao YUAN. "Low-depth Quantum State Preparation". In : *arXiv* (fév. 2021). eprint : [2102.07533](https://arxiv.org/abs/2102.07533). URL : <https://arxiv.org/abs/2102.07533v3>.
- [10] Yi ZHANG, Kai LU et YingHui GAO. "QSobel: A novel quantum image edge extraction algorithm". In : *Sci. China Inf. Sci.* 58.1 (jan. 2015), p. 1-13. ISSN : 1869-1919. DOI : [10.1007/s11432-014-5158-9](https://doi.org/10.1007/s11432-014-5158-9).