Université Hassan II de Casablanca
Faculté des Sciences et Techniques
-Mohammedia-

Département Informatique

Mémoire du Mini Projet du Module Transmission des Données Multimédia (TDM)

Licence Sciences et Techniques

Option : Informatique, Réseaux et Multimedia (IRM)

IRM : Une Nouvelle Approche de Compression d'Images

Réalisé Par :
Salma **AIT LAKBIR**
Ibtissam **ER-RACHIDI**
Latifa **JAKIR**
Khalil **MELLOUK**
Issa **SANGARÉ**

Encadrant :
Pr. Abdellah **ADIB**

Année universitaire 2023-2024

Table des matières

Remerciements

Ce n'est pas par tradition que cette page figure au préambule de ce rapport, mais c'est plutôt un devoir moral et une reconnaissance sincère qui nous poussent à le faire.

Nous serions en effet ingrats si nous n'exprimions pas notre reconnaissance et notre gratitude à tous ceux qui ont facilité notre tâche et nous ont permis de mener à bien ce travail.

Nous tenons tout particulièrement à exprimer toute notre gratitude et nos vifs remerciements à toute l'équipe qui œuvre continuellement pour nous assurer une formation de haut niveau dans les conditions les plus adéquates.

Et particulièrement à notre professeur et encadrant, **Mr Abdellah ADIB**, pour son aide, son expérience et son savoir-faire.

Nous le sollicitons de croire en notre respectueuse estime et notre sincère reconnaissance pour ses conseils qui nous ont été très précieux et indispensables.

Nous vous remercions chaleureusement.

Table des figures

Liste des sigles et acronymes

JPEG	Joint Photographic Experts Group
GIF	Graphics Interchange Format
BMP	Bitmap
TIFF	Tagged Image File Format
LZW	Lempel-Ziv-Welch
PNG	Portable Network Graphics
APP	Application Specific Protocol
SOI	Start of Image
YCbCr	Luminance-Chrominance Color Space
DCT	Discrete Cosine Transform
RLE	Run-Length Encoding
GUI	Graphical User Interface

Introduction Générale

Les images ont pris une place importante dans notre quotidien avec l'avènement des technologies numériques. Leur abondance pose des défis majeurs en termes de gestion des données, de bande passante des réseaux et de capacités de stockage.

La compression d'images émerge comme une solution cruciale pour résoudre ces défis. En réduisant la taille des fichiers image, elle permet d'économiser de l'espace de stockage et d'améliorer l'efficacité de la transmission des données tout en préservant la qualité visuelle.

Deux principaux types de compression d'images existent : la compression sans perte et la compression avec perte, chacune adaptée à des besoins spécifiques en matière de qualité et d'utilisation des données.

Ce mini-projet se concentre sur l'étude et la mise en œuvre d'algorithmes de compression d'images en utilisant Python. L'objectif est de simuler le codage et le décodage statistique, puis de tester les performances de l'algorithme de compression sur différentes images.

Une innovation est proposée dans le domaine de la compression d'images avec l'introduction d'un nouveau format nommé IRM, combinant diverses méthodes de compression statistique. Une interface utilisateur graphique est également développée pour compresser des images sous ce nouveau format, avec une évaluation qualitative de la méthode proposée.

Ce rapport explore les différentes facettes de la compression d'images, combinant une approche théorique avec une mise en pratique à travers le développement de fonctions sous Python et la conception d'une interface utilisateur innovante.

Présentation du Problème

1 Contexte

Avec la prolifération des images dans notre quotidien, leur gestion devient un défi majeur. La taille des fichiers d'image peut entraîner des problèmes de stockage, de bande passante réseau et de traitement informatique. Pour résoudre ces problèmes, la compression d'image est essentielle.

2 Objectif

L'objectif est de réduire la taille des fichiers d'image tout en préservant la qualité visuelle, ou en acceptant une perte de qualité acceptable dans le cas de la compression avec perte. Cette réduction de taille facilite le stockage, la transmission et le traitement des images.

3 Conclusion

La compression d'image est une pratique essentielle pour gérer efficacement les données multimédias. Ce problème nécessite une compréhension approfondie des techniques de compression et une implémentation pratique pour évaluer leur efficacité. La proposition d'un nouveau format de compression et son évaluation démontrent un effort continu pour améliorer les méthodes existantes.

Compression d'Images : Algorithmes, Théorie et Applications

4 Algorithmes de Compression

4.1 Codage Statique

4.1.1 RLE

4.1.1.1 Généralités et principes :

- Généralités :

La méthode de compression **RLE** (Run Length Encoding, parfois notée RLC pour Run Length Coding) est utilisée par de nombreux formats d'images (BMP, PCX, TIFF). Elle est basée sur la répétition d'éléments consécutifs.

- Principes :

Le principe de base consiste à coder un premier élément donnant le nombre de répétitions d'une valeur puis le compléter par la valeur à répéter. Ainsi, selon ce principe, la chaîne "AAAAAHHHHHHH" compressée donne "5A7H".

Le gain de compression est ainsi de (11-4)/11 soit environ 63,7%. En contrepartie, pour la chaîne "REELLEMENT", dans laquelle la redondance des caractères est faible, le résultat de la compression donne "1R2E2L1E 1M1E1N1T" ; la compression s'avère ici très coûteuse, avec un gain négatif valant (10-16)/10 soit 60% !

En réalité, la compression RLE est régie par des règles particulières permettant de compresser lorsque cela est nécessaire et de laisser la chaîne telle quelle lorsque la compression induit un gaspillage. Ces règles sont les suivantes :

- Lorsque trois éléments ou plus se répètent consécutivement, alors la méthode de compression RLE est utilisée.

- Sinon, un caractère de contrôle (00) est inséré, suivi du nombre d'éléments de la chaîne non compressée puis de cette dernière.

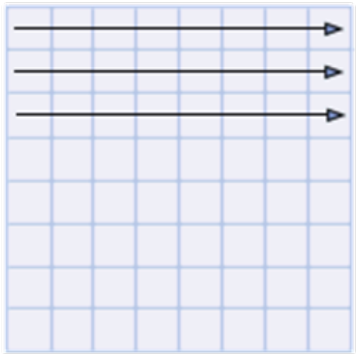

FIGURE 1 – Lecture Ligne

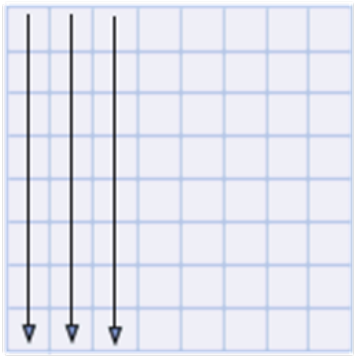

FIGURE 2 – Lecture Colonne

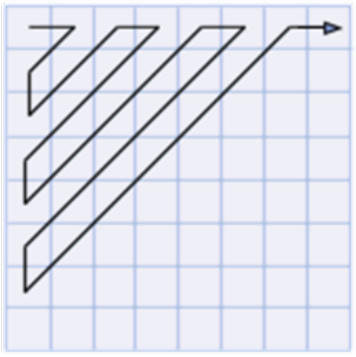

FIGURE 3 – Lecture Zig-Zag

Ainsi, la compression RLE n'a de sens que pour les données possédant de nombreux éléments consécutifs redondants, notamment les images possédant de larges parties uniformes. Cette méthode a toutefois l'avantage d'être peu difficile à mettre en œuvre. Il existe des variantes dans lesquelles l'image est encodée par pavés de points, selon des lignes, ou même en zigzag.

4.1.1.2 Applications :

Domaine d'application du code RLE :

Images Bitmap :

Les images bitmap, en particulier celles qui présentent de grandes zones de couleur uniforme ou des motifs répétitifs, peuvent bénéficier de la compression RLE. Les formats d'image comme BMP peuvent utiliser RLE pour compresser les données.

Images en noir et blanc :

Les images monochromes ou en noir et blanc, telles que les icônes et les images simples, sont souvent compressées à l'aide de RLE.

Images avec des zones de couleur uniforme :

Les images avec de grandes zones de couleur uniforme, comme les captures d'écran ou les images générées par ordinateur, sont également des candidats idéaux pour la compression RLE.

Images médicales :

Les images médicales telles que les rayons X, les IRM et les scans CT peuvent être compressées avec RLE, en particulier lorsque la qualité de l'image peut être légèrement réduite sans perte significative d'informations diagnostiques.

Vidéos :

Bien que moins courant que dans le cas des images, le RLE peut être utilisé dans la compression de certains types de vidéos, en particulier pour les formats où les images clés peuvent être compressées individuellement.

Animation GIF :

Les fichiers GIF animés utilisent souvent une forme de compression RLE pour stocker les images clés qui composent chaque image de l'animation.

Données graphiques dans les jeux vidéo :

Dans le développement de jeux vidéo, le RLE peut être utilisé pour compresser les textures et autres données graphiques afin de réduire l'espace de stockage nécessaire et d'accélérer le chargement des ressources.

4.1.2 Huffman

4.1.2.1 Généralités et principes :

- Généralités :

David Huffman a proposé en 1952 une méthode statistique qui permet d'attribuer un mot de code binaire aux différents symboles à compresser (pixels ou caractères par exemple). La longueur de chaque mot de code n'est pas identique pour tous les symboles : les symboles les plus fréquents (qui apparaissent le plus souvent) sont codés avec de petits mots de code, tandis que les symboles les plus rares reçoivent de plus longs codes binaires. On parle de codage à longueur variable (en anglais VLC pour variable code length) préfixé pour désigner ce type de codage car aucun code n'est le préfixe d'un autre. Ainsi, la suite finale de mots codés à longueurs variables sera en moyenne plus petite qu'avec un codage de taille constante.

- Principe :

Le codeur de Huffman crée un arbre ordonné à partir de tous les symboles et de leur fréquence d'apparition. Les branches sont construites récursivement en partant des symboles les moins fréquents.

La construction de l'arbre se fait en ordonnant dans un premier temps les symboles par fréquence d'apparition. successivement les deux symboles de plus faible fréquence d'apparition sont retirés de la liste et rattachés à un noeud dont le poids vaut la somme des fréquences des deux symboles. Le symbole de plus faible poids est affecté à la branche 1, l'autre à la branche 0 et ainsi de suite en considérant chaque noeud formé comme un nouveau symbole, jusqu'à obtenir un seul noeud parent appelé racine.

Le code de chaque chaque symbole correspond à la suite des codes le long du chemin allant de ce caractère à la racine. Ainsi, plus le symbole est "profond" dans l'arbre, plus le mot de code sera long.

Soit la phrase suivante : "*COMMENT_CA_MARCHE*". Voici les fréquences d'apparition des lettres.

Lettre	M	A	C	E	_	H	O	N	T
Fréquence	3	2	2	2	2	1	1	1	1

TABLE 1 – Fréquences d'apparition des lettres

Voici l'arbre correspondant :

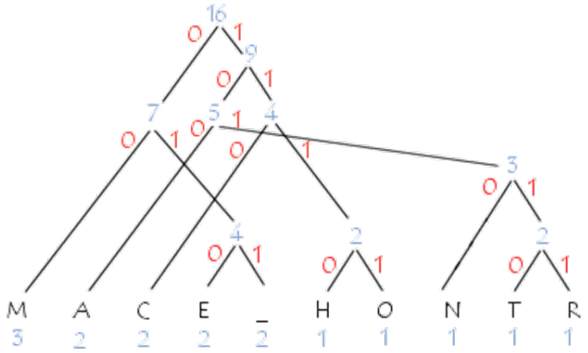

FIGURE 4 – Arbre

Les codes correspondants à chaque caractère sont tels que les codes des caractères les plus fréquents sont courts et ceux correspondant aux symboles les moins fréquents sont longs :

Lettre	Code binaire
M	00
A	100
C	110
E	010
_	011
H	1110
O	1111
N	1010
T	10110
R	10111

TABLE 2 – Tableau des lettres et de leurs codes binaires

Les compressions basées sur ce type de codage donnent de bons taux de compressions, en particulier pour les images monochromes (les fax par exemple). Il est notamment utilisé dans les recommandations T4 et T5 de l'ITU-T.

4.1.2.2 Applications :

Domaine d'application du code de Huffman :

Compression de fichiers texte :

Le codage de Huffman est très efficace pour compresser des fichiers texte, tels que des documents, des livres électroniques, des scripts, etc. Il peut réduire la taille des fichiers texte en exploitant les fréquences d'apparition des caractères.

Compression d'images sans perte :

Bien que le codage de Huffman ne soit pas aussi efficace que d'autres méthodes pour la compression d'images, il est utilisé dans certaines formes de compression sans perte, notamment dans des formats d'images tels que TIFF.

Compression audio sans perte :

Dans certains formats audio sans perte, tels que FLAC (Free Lossless Audio Codec), le codage de Huffman est utilisé pour compresser les données audio sans perte de qualité.

Compression de données de texte dans les communications réseau :

Le codage de Huffman peut être utilisé pour compresser les données de texte transmises sur des réseaux, réduisant ainsi la bande passante nécessaire pour la transmission.

Stockage de données dans les bases de données et les systèmes de fichiers :

Dans certains systèmes de stockage de données, le codage de Huffman peut être utilisé pour compresser des données textuelles ou structurées afin de réduire l'espace de stockage requis.

Compression de données de données génomiques :

Dans le domaine de la bioinformatique, le codage de Huffman est utilisé pour compresser des données de séquences génomiques en raison de la répétitivité des motifs dans ces données.

4.2 Codage dynamique (utilisant un dictionnaire)

4.2.1 LZ78 :

4.2.1.1 Généralités et principes :

- Généralités :

LZ78 inventé par Abraham Lempel et Jacob Ziv en 1978 est un algorithme de compression sans perte. Il réduit les redondances dans les données en remplaçant les éléments récurrents par des références dans un dictionnaire. Cet algorithme est largement utilisé dans des applications de compression de fichiers et transmission de données.

- Principe :

La compression LZ78 fonctionne selon le principe suivant :

- Initialement, un dictionnaire vide est créé, ne contenant que le mot vide (indice 1).

- À chaque étape, le plus long préfixe v contenu dans le dictionnaire est recherché. Le préfixe suivant $v\alpha$ n'est pas dans le dictionnaire.

- On émet le couple $(Dico[v], \alpha)$, où $Dico[v]$ est l'indice de v dans le dictionnaire.

- On ajoute $v\alpha$ dans le dictionnaire, son indice étant la taille du dictionnaire moins 1.

- On reprend le texte après $v\alpha$.

Ainsi, l'algorithme génère une séquence de couples (i, c), où i est un indice dans le dictionnaire et c un caractère.

Remarque : À la fin, si la totalité du mot est dans le dictionnaire, le mot sans la dernière lettre est choisi pour v, afin qu'il reste un caractère α à ajouter. Cela évite d'avoir à ajouter un symbole "pas de caractère".

Exemple : Pour le mot u = abaaaabaab

- la séquence de sortie sera $(1, a), (2, b), (1, a), (3, b), (3, b)$.

La décompression consiste simplement à reconstituer le dictionnaire exactement de la même manière que l'algorithme de compression.

4.2.1.2 Applications :

Domaine d'application de LZ78 :

Stockage de données :

L'algorithme LZ78 est utilisé pour compresser des fichiers et des données avant de les stocker pour économiser l'espace de stockage.

Compression des fichiers et des archives :

Il existe plusieurs logiciels de compression comme WinZip et 7-Zip qui utilisent l'algorithme LZ78.

Compression d'images et de vidéos :

L'algorithme LZ78 peut être utilisé dans certains contextes pour la compression d'images et de vidéos que d'autres algorithmes plus avancés tels que JPEG et MPEG.

Intégration dans les systèmes de reconnaissance vocale :

Les systèmes de reconnaissance vocale peuvent exploiter l'algorithme LZ78 pour comprimer les modèles acoustiques afin de réduire la taille des données nécessaires pour l'entraînement de ces systèmes.

Télédétection et traitement d'images satellitaires :

La compression à l'aide de LZ78 permet de réduire la quantité d'informations transmises, ce qui est important pour la transmission et le stockage efficaces des images satellitaires.

4.2.2 LZW

4.2.2.1 Généralités et principes :

- Généralités :

Abraham Lempel et Jakob Ziv sont les créateurs du compresseur LZ77, inventé en 1977 (d'où son nom). Ce compresseur était alors utilisé pour l'archivage (les formats ZIP, ARJ et LHA l'utilisent).

En 1978 ils créent le compresseur LZ78 spécialisé dans la compression d'images (ou tout type de fichier de type binaire).

En 1984, Terry Welch de la société Unisys le modifia pour l'utiliser dans des contrôleurs de disques durs, son initiale vint donc se rajouter à l'abréviation LZ pour donner LZW.

LZW est un algorithme très rapide aussi bien en compression qu'en décompression, basé sur la multiplicité des occurences de séquences de caractères dans la chaîne à encoder.

- Principe :

Son principe consiste à substituer des motifs par un code d'affectation (indice) en construisant au fur et à mesure un dictionnaire.

De plus, il travaille sur des bits et non sur des octets, il ne dépend donc pas de la manière de laquelle le processeur code les informations. C'est un des algorithmes les plus populaires, il est notamment utilisé dans les formats TIFF et GIF. La méthode de compression LZW ayant été brevetée par la société Unisys, c'est l'algorithme LZ77, libre de droit, qui est utilisé dans les images PNG.

- Construction du dictionnaire

Le dictionnaire est initialisé avec les 256 valeurs de la table ASCII. Le fichier à compresser est découpé en chaînes d'octets (ainsi pour des images monochromes - codées sur 1 bit - cette compression est peu efficace), chacune de ces chaînes est comparée au dictionnaire et est ajoutée si jamais elle n'y est pas présente.

- La Compression

L'algorithme parcourt le flot d'informations en le codant ; si jamais une chaîne est plus petite que le plus grand mot du dictionnaire alors elle est transmise.

- La Decompression

Lors de la décompression, l'algorithme reconstruit le dictionnaire dans le sens inverse, ce dernier n'a donc pas besoin d'être stocké.

4.2.2.2 Applications :

Domaine d'application du code LZW :

Compression de fichiers texte :

Le codage LZW est très efficace pour compresser des fichiers texte, en particulier ceux contenant des motifs répétitifs ou des séquences de caractères similaires. Il est couramment utilisé dans des formats de fichiers tels que GIF et PDF pour compresser le texte incorporé.

Compression d'images :

Le codage LZW est également utilisé dans certains formats d'images, notamment le format GIF. Il peut être utilisé pour compresser des images avec une palette de couleurs limitée, en exploitant les répétitions dans les données de couleur.

Compression de données audio

Bien que moins courant que dans les images ou les fichiers texte, le codage LZW peut également être utilisé dans certains formats de compression audio sans perte pour compresser des données audio non compressées.

Compression de données de communications réseau :

Le codage LZW peut être utilisé pour compresser des données transmises sur des réseaux, réduisant ainsi la bande passante nécessaire pour la transmission de données.

Stockage de données dans les bases de données et les systèmes de fichiers :

Dans certains systèmes de stockage de données, le codage LZW peut être utilisé pour compresser des données structurées ou des données textuelles, réduisant ainsi l'espace de stockage requis.

Compression de données de données génomiques :

Dans le domaine de la bioinformatique, le codage LZW peut être utilisé pour compresser des données de séquences génomiques en raison de la répétitivité des motifs dans ces données.

ETUDES THÉORIQUES

5 Notion de compression d'images :

5.1 Définition, généralités et principes :

5.1.1 Classification :

5.1.1.1 Compression avec pertes :

Imaginons un voyageur qui prépare ses vêtements pour un voyage. Il a la possibilité d'emporter tout le contenu de sa garde-robe, des chaussures aux chapeaux en passant par les tenues de soirée. Il en résulterait une grande quantité de bagages à transporter, ce qui ralentit le voyageur et peut coûter plus cher en transport.
Il va donc préférer se limiter à une sélection des vêtements les plus importants qu'il mettra dans une seule valise.

Tout comme notre voyageur n'avait pas besoin d'emporter l'intégralité de sa garde-robe, il est rare d'avoir besoin de voir une image entière dans sa résolution maximale et à ses dimensions les plus grandes. En général, la qualité et la taille d'une image peuvent être réduites sans que l'observateur classique ne s'en aperçoive, c'est ce qu'on appelle la compression d'image avec « perte ».

La compression d'image avec perte préserve les informations les plus essentielles de l'image sans conserver chaque pixel.

Il existe plusieurs types d'algorithmes de compression avec perte. Ils ont cependant tous en commun de supprimer des informations du fichier d'image afin de réduire le nombre d'octets composant l'image.

5.1.1.2 Compression sans pertes :

Si tous les vêtements que le voyageur souhaite emporter ne logent pas dans la valise, il peut essayer de les plier différemment et de les réorganiser pour tout faire entrer. De la même manière, la compression d'images « sans perte » utilise des algorithmes mathématiques pour réécrire un fichier image sans supprimer aucune information.

Une image traitée avec une compression sans perte doit paraître globalement identique à l'originale, mais la taille du fichier doit être beaucoup plus petite.

Si la compression sans perte peut réduire la taille des fichiers d'images jusqu'à 40 elle reste toutefois moins efficace que la compression avec perte pour réduire la taille des fichiers.

5.1.2 Applications :

5.1.2.1 JPEG

Le format JPEG est un format de fichier d'image numérique largement utilisé pour sa capacité à compresser les images avec perte, réduisant ainsi la taille du fichier sans affecter de manière significative la qualité visuelle pour l'œil humain. La compression JPEG fonctionne en supprimant les détails imperceptibles, ce qui permet d'obtenir des fichiers plus petits que les formats non compressés. Le niveau de compression est réglable, permettant de trouver un équilibre entre la qualité de l'image et la taille du fichier. Le format JPEG est compatible avec la plupart des appareils photo numériques et des logiciels d'édition d'images, ce qui en fait un choix populaire pour la photographie numérique et le partage d'images en ligne.

5.1.2.2 GIF

Le format GIF (Graphics Interchange Format) est un format d'image numérique léger et sans perte, idéal pour les petits graphiques web et les animations simples. Il prend en charge la transparence et 256 couleurs, ce qui le rend populaire pour les mèmes et les images amusantes, mais moins adapté aux photos. Largement compatible, le GIF offre un bon compromis entre la taille du fichier et la qualité d'image pour les images web non photographiques.

5.1.2.3 PNG

Le format PNG (Portable Network Graphics) est un format d'image numérique sans perte, offrant une meilleure qualité que le GIF avec un support de millions de couleurs et de la transparence. Il est idéal pour les illustrations, les logos et les images avec des arrière-plans transparents. Sa prise en charge de l'animation est limitée, mais il offre une qualité et une flexibilité supérieures aux formats GIF pour les images non photographiques.

5.1.2.4 TIFF

Le format TIFF (Tagged Image File Format) est un format d'image numérique sans perte, idéal pour les impressions haute résolution et la photographie d'art. Il offre une qualité photographique exceptionnelle, mais génère des fichiers volumineux, ce qui le rend moins pratique pour le partage en ligne. Largement utilisé par les professionnels de l'image, il offre une grande flexibilité et une compatibilité avec divers logiciels.

5.1.2.5 BMP

Le format BMP (Bitmap) est un format d'image numérique non compressé offrant une qualité d'image optimale sans perte de données. Cependant, il génère des fichiers volumineux, ce qui peut limiter le partage et le stockage. Particulièrement adapté aux analyses détaillées et aux copies d'archives, il est compatible avec de nombreux systèmes d'exploitation, bien qu'initialement développé par Microsoft pour Windows.

5.1.3 Étapes de mise en place :

5.1.3.1 Compression avec perte :

1 - Conversion de l'image en une représentation adaptée à la compression

Cela peut impliquer la conversion de l'image d'un espace colorimétrique à un autre ou la séparation de l'image en composantes (comme les composantes de luminance et de chrominance dans le cas de l'espace colorimétrique YUV).

2 - Sous-échantillonnage

Dans le cas des images couleur, cela peut impliquer la réduction de la résolution des composantes de chrominance par rapport à la composante de luminance.

3 - Transformée de l'image

Les techniques de transformation comme la transformée en cosinus discrète (DCT) sont souvent utilisées pour convertir les données spatiales de l'image en une représentation fréquentielle.

4 - Quantification

Cette étape consiste à réduire la précision de certains éléments de l'image, généralement en se basant sur la perception humaine des différences de couleur et de luminosité.

5 - Codage entropique

Les données quantifiées sont compressées en utilisant des méthodes de codage efficaces, telles que le codage Huffman ou le codage arithmétique, pour réduire davantage la taille du fichier.

5.1.3.2 Compression sans perte :

1 - Détection des redondances :

Identification des zones répétitives ou prévisibles dans l'image.

2 - Utilisation de techniques de codage sans perte :

Cela peut inclure des méthodes comme le codage Run-Length (RLE), le codage de Huffman, ou encore des algorithmes de compression plus sophistiqués comme le codage de Lempel-Ziv-Welch (LZW).

3 - Segmentation :

Dans certains cas, l'image peut être divisée en segments pour lesquels des techniques de compression spécifiques peuvent être appliquées de manière plus efficace.

4 - Codage prédictif :

Prédire les valeurs des pixels en fonction de leurs voisins et coder les différences plutôt que les valeurs absolues.

6 Formalisme des algorithmes de compression JPEG et GIF

6.1 JPEG :

6.1.1 Entête :

L'entête d'un fichier JPEG est une section cruciale qui contient des informations métadonnées essentielles pour l'image compressée. Ces métadonnées incluent généralement :

Format Marker :

Un marqueur spécifique (0xFFD8) indiquant le début d'un fichier JPEG.

Segment d'Application (APP) :

Des segments optionnels contenant des informations supplémentaires telles que des commentaires ou des données d'application spécifiques.

Définition d'Image (SOI) :

Un marqueur (0xFFE0) indiquant le début de l'image JPEG.

Paramètres d'Image :

Ces paramètres incluent des informations telles que la hauteur et la largeur de l'image, la précision des échantillons, le nombre de composants de couleur, etc.

Tables de Quantification et Tables Huffman :

Ces tables sont utilisées lors de la compression et de la décompression pour normaliser les données et optimiser la taille du fichier sans compromettre excessivement la qualité de l'image.

6.1.2 Données :

Les données JPEG consistent en l'information réelle nécessaire pour représenter l'image après compression. Elles sont généralement organisées en unités de données appelées "MCU" (Minimum Coded Unit) ou "blocs". Les étapes typiques pour générer ces données incluent celles proposées par 7.1

6.2 GIF :

6.2.1 Entête :

L'entête d'un fichier GIF contient des informations initiales sur l'image compressée. Cela comprend généralement :

Signature et Version du Format GIF :

Un marqueur (0x474946383961) indiquant le début d'un fichier GIF et la version du format GIF utilisée.

Dimensions de l'Image :

La largeur et la hauteur de l'image en pixels.

Palette de Couleurs :

Une table de couleurs limitée, souvent de taille maximale 256 couleurs, utilisée pour représenter les couleurs dans l'image.

6.2.2 Données :

Les données GIF comprennent les pixels de l'image ainsi que les informations sur la palette de couleurs utilisée. Les principales étapes de compression incluent celles proposées par 7.2.

7 Étapes des algorithmes de compression JPEG et GIF :

7.1 JPEG :

Les étapes typiques de compression JPEG comprennent :

7.1.1 Matricage :

L'opération démarre par l'extraction de la matrice de pixels de l'image RGB. Cela se fait typiquement après l'ouverture du fichier, par exemple avec Image.open(), suivie de la conversion en un tableau numpy via numpy.array(Image.open()) en Python. Les manipulations ultérieures de l'image, sous forme matricielle, agissent sur les pixels, qui représentent les coefficients des matrices de l'image.

7.1.2 Changement d'espace de couleur :

Cette étape permet une compression plus efficace tout en préservant la qualité visuelle de l'image.

7.1.3 Subdivision de l'image en blocs :

La matrice de pixels de taille L x C est découpée en blocs de 8 * 8 pixels. Cette taille facilite la manipulation et correspond à celle requise pour la matrice de quantification. Si le nombre de lignes L et/ou de colonnes C n'est pas un multiple de 8, des pixels supplémentaires sont ajoutés avec des valeurs nulles.

7.1.4 Sous-échantillonnage :

Cette étape réduit les matrices de chrominance Cr et Cb en matrices 4 * 4 en utilisant la méthode 4 :2 :0. Chaque bloc 8 * 8 de la matrice de luminance est associé à des blocs 4 * 4 de ces nouvelles matrices chrominances. Cette réduction ne s'applique qu'aux composantes de chrominance, permettant ainsi de diminuer la quantité de données tout en préservant une bonne qualité d'image.

7.1.5 Application de la DCT (Transformée en Cosinus Discrète) :

La DCT réorganise l'information pour concentrer l'énergie de l'image dans un petit nombre de coefficients, facilitant ainsi la compression tout en conservant une qualité visuelle acceptable.

7.1.6 Quantification :

La matrice résultante de la DCT est quantifiée pour chaque plan de luminance et de chrominance par une matrice spécifique prédéfinie. Cette étape est principalement responsable de la perte de données. La matrice de quantification divise la matrice de la DCT et arrondit les valeurs, conduisant à une matrice avec peu de valeurs différentes de zéro dans la partie supérieure gauche et des zéros pour le reste de la matrice. Lors de la décompression, cette matrice quantifiée est utilisée pour retrouver une approximation de l'image originale.

7.1.7 Parcours des matrices :

Le parcours en zigzag est utilisé pour maximiser les répétitions, favorisant ainsi le codage RLE.

7.1.8 Codage RLE :

Les vecteurs résultants du parcours en zigzag sont codés par RLE pour réduire la taille de l'information sans perdre de données.

7.1.9 Codage Huffman :

Le résultat du codage RLE est soumis au codage Huffman pour une compression supplémentaire.

7.2 GIF :

Les étapes typiques de compression GIF comprennent :

7.2.1 Matricage :

Cette opération est effectuée après l'ouverture du fichier, par exemple avec Image.open(), suivie de la conversion en un tableau numpy via numpy.array(Image.open()). Les manipulations ultérieures de l'image agissent sur les pixels, qui sont les coefficients des matrices représentant l'image.

7.2.2 Création de la palette de couleur :

Le k-means clustering est utilisé pour créer une palette de couleurs représentant l'image. Les centroïdes sont choisis aléatoirement parmi les pixels, puis des clusters sont déterminés pour chaque centroïde en associant les pixels au centroïde le plus proche. Ce processus est répété jusqu'à ce qu'une certaine condition de convergence soit atteinte.

7.2.3 Mappage :

Chaque pixel de l'image est associé à un élément de la palette de couleurs (un centroïde), réduisant ainsi la représentation des pixels aux index de la palette.

7.2.4 Passage de la matrice au vecteur :

La matrice est convertie en vecteur selon un parcours spécifique, comme en ligne, en colonne ou en zigzag, pour une représentation plus compacte.

7.2.5 Codage par LZW :

L'algorithme de Lempel-Ziv-Welch est utilisé pour compresser les données en se basant sur une table de code (la palette de couleurs dans ce cas). À la fin de cette étape, le fichier compressé est obtenu, préservant les informations essentielles pour la décompression.

Innovation

8 Introduction :

Dans cette démarche innovante, nous nous engageons à concevoir un nouveau format de compression d'image révolutionnaire, baptisé IRM, qui surpasse les limitations existantes des formats de compression traditionnels sur le marché. En exploitant les puissantes techniques de compression RLE, Huffman et LZW, le format IRM vise à offrir une compression efficace tout en préservant la qualité de l'image, quel que soit son type : binaire, niveau de gris ou couleur.

Grâce à une approche pratique et expérimentale, nous allons simuler les encodages et décodages pour chacune des méthodes de compression statistique, en utilisant le langage de programmation Python. Ces simulations seront ensuite testées pour réaliser des opérations de compression d'image sans perte, permettant ainsi une évaluation approfondie des performances du format IRM.

Le rendu final de ce projet comportera une interface utilisateur graphique conviviale, permettant aux utilisateurs de compresser facilement leurs images au format .IRM. Cette interface sera dotée d'outils de mesure de la qualité de compression et du taux de compression atteint, offrant ainsi aux utilisateurs une expérience optimale.

En résumé, ce projet repose sur une approche innovante qui vise à repousser les limites actuelles de la compression d'image en proposant un nouveau format, IRM, qui combine les meilleures pratiques des techniques de compression existantes tout en proposant une nouvelle en-tête pour surmonter les limitations observées.

9 Étapes de l'algorithme de compression IRM :

9.1 Prétraitement :

Commencez par extraire la ou les matrices de pixels de l'image RGB. Cette opération intervient après l'ouverture du fichier, par exemple avec `Image.open()`, suivie de sa conversion en un tableau numpy via `numpy.array(Image.open())`, en utilisant le langage de programmation Python. Tous les traitements ultérieurs de l'image sous forme matricielle agissent sur les pixels, qui sont les coefficients des matrices représentant l'image.

9.2 Changement d'espace de couleur :

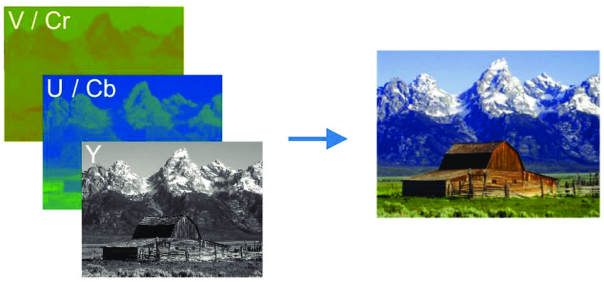

FIGURE 5 – Changement d'espace de couleur

Le changement d'espace de couleur est une étape fondamentale de la compression d'image, impliquant la conversion de l'espace colorimétrique de l'image d'origine vers un autre espace. L'espace colorimétrique YCbCr est la méthode utilisée. Cette conversion permet une meilleure représentation des caractéristiques visuelles de l'image et peut faciliter la compression tout en préservant la qualité visuelle. Le choix de l'espace colorimétrique dépend des propriétés perceptuelles de l'œil humain et des exigences spécifiques de l'application.

9.3 Création de la palette de couleurs :

La création de palettes par k-means clustering, qui est la méthode de création de palette la plus efficace pour cette méthode de compression, consiste à choisir au hasard comme centroïdes parmi les pixels (les combinaisons de couleurs [rouge, vert, bleu]), un nombre égal à celui de la palette de couleur demandée. Après cette sélection aléatoire de ces centroïdes, des clusters sont déterminés pour chaque centroïde en calculant la distance entre les valeurs des pixels de l'image et celui de tous les centroïdes, puis en associant le pixel au centroïde avec lequel la distance est la plus faible. Un cluster désigne l'ensemble des pixels associés à un centroïde pendant une itération donnée. Après avoir déterminé tous les clusters de tous les centroïdes, les centroïdes sont mis à jour en calculant la moyenne par canal (couleur) des clusters qui lui sont associés. Ensuite, ce processus est répété jusqu'à obtenir une certaine valeur de tolérance, un nombre maximal d'itérations ou une convergence des centroïdes. Ainsi, la palette de couleurs à 2^n valeurs distinctes est créée.

FIGURE 6 – Palette de couleurs

9.4 Mappage :

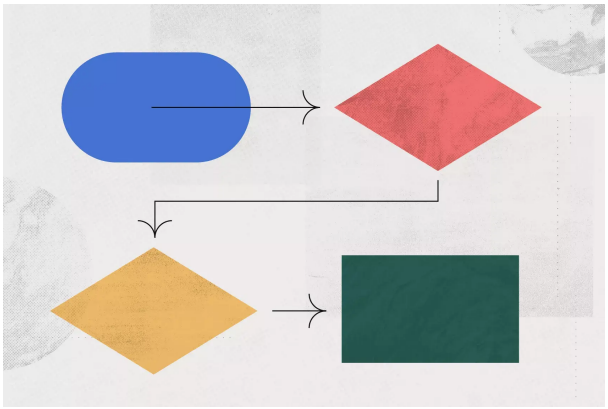

FIGURE 7 – Mappage

Chaque pixel de l'image est associé à une couleur (un élément) de la palette de couleurs (un centroïde). Ainsi, au lieu que les pixels soient représentés par les 3 composantes [rouge, vert, bleu], ils sont représentés par des indices de la palette de couleurs (les centroïdes), généralement de 0 à $2^n - 1$. Ce mappage est réalisé en calculant la distance euclidienne entre les valeurs de chaque pixel et celles de tous les éléments de la palette de couleurs. Ensuite, le pixel prend l'index de l'élément de la palette de couleurs avec lequel la distance est la plus faible.

9.5 Subdivision de l'image en blocs :

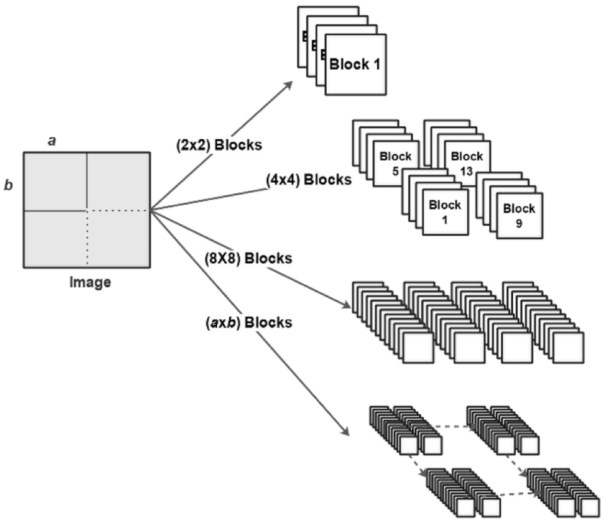

FIGURE 8 – Subdivision de l'image en blocs

La matrice (ou les matrices) de pixels de taille L x C est découpée en blocs de 8 * 8 pixels. Cette taille facilite la manipulation de l'image originale et offre une combinaison de simplicité de traitement, d'efficacité des algorithmes de compression, de compatibilité et de facilité de manipulation matricielle, ce qui en fait un choix pratique et performant pour la compression IRM. Si le nombre de lignes L et/ou de colonnes C n'est pas un multiple de 8, des pixels supplémentaires sont ajoutés avec des valeurs nulles.

9.6 Sous-échantillonnage :

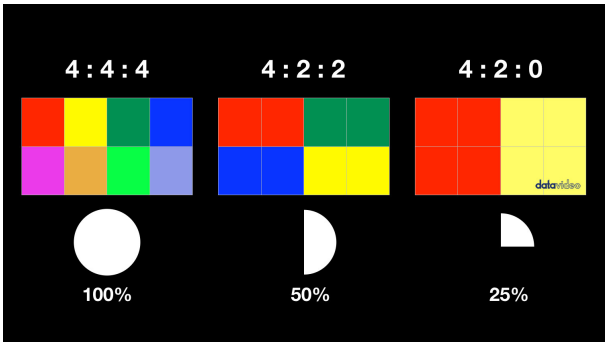

FIGURE 9 – Sous-échantillonnage

De la même manière que la méthode 4 :4 :4 propre aux composantes chrominances lors de la compression JPEG, cette étape consiste à réduire les valeurs des blocs de pixels. Cependant, ici, les blocs ne subissent aucune modification ; ils restent inchangés.

9.7 Parcours des matrices :

Pendant cette étape, nous vectorisons la matrice en utilisant un parcours en ligne, ce qui, grâce à la disposition de cette dernière, maximise les répétitions, favorisant ainsi le codage RLE.

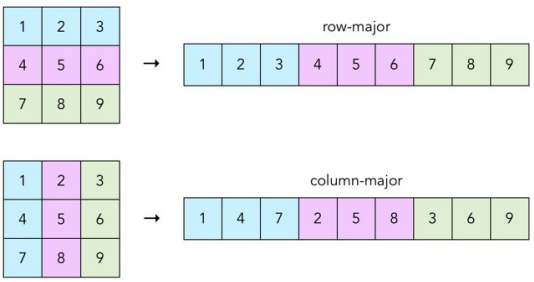

FIGURE 10 – Vectorisation

9.8 Codage RLE :

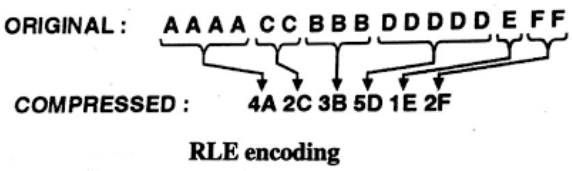

FIGURE 11 – Codage RLE

Après le parcours en zigzag, les vecteurs résultants sont codés par RLE pour réduire la taille de l'information et des données sans perdre aucune information.

9.9 Codage Huffman :

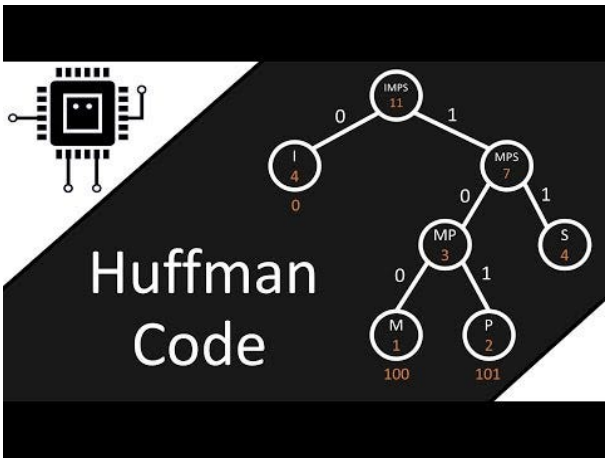

FIGURE 12 – Codage Huffman

Le codage de Huffman est appliqué au résultat du codage RLE. Il est utilisé pour représenter les symboles résultant du codage RLE de manière efficace en fonction de leur fréquence.

9.10 Codage par LZW :

Enfin, vient l'étape du codage LZW, où nous rassemblons les codes de Huffman de tous les blocs de code. Le codage par l'algorithme de Lempel–Ziv-Welch est la dernière étape du processus de compression d'image. Il se

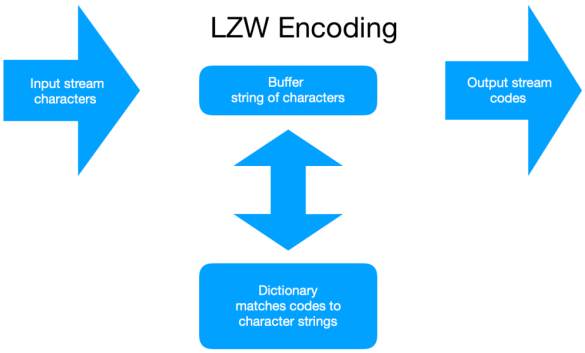

FIGURE 13 – Codage LZW

base sur l'existence d'une table de code (ici notre palette de couleurs, les centroïdes, qui représentent le dictionnaire). À la fin de cette étape, nous obtenons le fichier compressé, qui, avec les données de l'en-tête portant les informations sur le fichier et comment le décompresser, permet de reconstituer l'image initiale avec les mêmes données et la même information.

Développement de l'interface

10 Introduction :

Le développement de l'interface utilisateur est une étape essentielle dans la concrétisation de la proposition d'innovation du nouveau format de compression d'image IRM. Cette interface constitue le point d'interaction entre l'utilisateur et le système de compression, offrant une plateforme conviviale et intuitive pour réaliser efficacement la compression des images au format IRM.

L'interface utilisateur graphique (GUI) joue un rôle central dans l'expérience utilisateur, en fournissant des outils et des fonctionnalités nécessaires pour charger, visualiser, compresser et enregistrer les images. Conçue avec soin, elle doit offrir une navigation fluide, une disposition claire des éléments et une facilité d'utilisation qui permettent à l'utilisateur de tirer pleinement parti des capacités du nouveau format IRM.

Dans cette section du développement, nous explorerons les principaux aspects de la conception et de la mise en œuvre de l'interface utilisateur pour le format IRM. Nous aborderons les objectifs et les fonctionnalités clés de l'interface, les choix de conception, les technologies utilisées et les considérations ergonomiques pour garantir une expérience utilisateur optimale.

À travers cette introduction, nous jetterons les bases pour un développement réussi de l'interface utilisateur, mettant en lumière l'importance de cette composante dans la réalisation de l'innovation proposée et soulignant son rôle crucial dans la création d'une solution complète et fonctionnelle pour la compression d'image au format IRM.

11 Langage de développement :

11.1 Python :

Python est un langage de programmation de haut niveau, interprété et polyvalent, créé par Guido van Rossum et publié pour la première fois en 1991 [1]. Il se distingue par sa syntaxe claire et lisible, qui favorise la lisibilité du code et la productivité des développeurs. Python est largement utilisé dans de nombreux domaines, notamment le développement web, la science des données, l'intelligence artificielle, l'automatisation des tâches, le développement de jeux, et bien plus encore. Il est apprécié pour sa simplicité, sa polyvalence, sa vaste bibliothèque standard et sa communauté active de développeurs. Python est un langage interprété, ce qui signifie que le code source est exécuté ligne par

ligne par un interpréteur Python, ce qui le rend adapté au développement rapide et au prototypage [7].

11.1.1 Bibliothèques et modules :

11.1.1.1 Matplotlib.pyplot :

Matplotlib est une bibliothèque de visualisation de données très populaire en Python. Son module `pyplot` offre une interface conviviale pour créer une grande variété de graphiques et de tracés, tels que des diagrammes en barres, des graphiques linéaires, des histogrammes, etc. Il permet également de personnaliser les graphiques avec des étiquettes, des titres, des légendes, etc., et de les sauvegarder dans différents formats d'image [2].

11.1.1.2 module_new_irm

Ce module Python implémente un algorithme de compression d'images. Il utilise une combinaison de techniques telles que le codage Huffman pour réduire la taille des données. L'algorithme commence par convertir une image RGB en un espace colorimétrique YCrCb, puis effectue un clustering K–Means pour obtenir une palette de couleurs. Ensuite, chaque pixel de l'image est mappé à la palette de couleurs, réduisant ainsi la quantité de données à compresser. Les données résultantes sont divisées en blocs, puis soumises à un processus de compression comprenant un sous-échantillonnage et un balayage linéaire. Enfin, les données compressées sont encodées avec LZW et stockées dans un fichier. Le processus de décompression inverse les étapes de compression pour reconstituer l'image d'origine.

11.1.1.3 PyQt5 :

PyQt5 est une bibliothèque Python qui offre des liaisons Python pour le framework d'interface graphique Qt. Il permet aux développeurs Python de créer des applications GUI multiplateformes en utilisant les fonctionnalités complètes de Qt. PyQt5 comprend des modules pour la création de fenêtres, de boîtes de dialogue, de menus, ainsi que pour la gestion d'événements utilisateur et la communication avec d'autres composants de l'application [11].

11.1.1.4 Numpy :

NumPy est une bibliothèque fondamentale pour le calcul numérique en Python. Elle offre des structures de données efficaces pour la manipulation de tableaux multidimensionnels, appelés "arrays", ainsi que des fonctions pour effectuer des opérations mathématiques et statistiques avancées sur ces tableaux [3].

11.1.1.5 sklearn.cluster.KMeans :

Scikit-learn est une bibliothèque Python pour l'apprentissage automatique. Son module `cluster` contient des algorithmes de clustering, dont KMeans, qui est utilisé pour regrouper les données en clusters en utilisant l'algorithme des K-moyennes [12].

11.1.1.6 PIL.Image :

PIL (Python Imaging Library) est une bibliothèque Python pour le traitement d'images. Son module `Image` offre des fonctionnalités pour ouvrir, manipuler et enregistrer des images dans différents formats, ainsi que pour effectuer des opérations de base telles que le redimensionnement, la rotation, le recadrage, etc [4].

11.1.1.7 os :

Le module `os` de Python fournit des fonctionnalités pour interagir avec le système d'exploitation sous-jacent. Il permet de manipuler des fichiers et des répertoires, de gérer les chemins d'accès, d'exécuter des commandes système, et bien plus encore [9].

11.1.1.8 scipy.fftpack.idct :

SciPy est une bibliothèque Python pour les calculs scientifiques et techniques. Son sous-module `fftpack` fournit des fonctions pour effectuer la transformée en cosinus discrète (DCT) et son inverse (IDCT), utilisées notamment dans la compression d'image et le traitement du signal [13].

11.1.1.9 re :

Le module `re` de Python offre des fonctionnalités pour travailler avec les expressions régulières. Il permet de rechercher, de remplacer et de manipuler des chaînes de caractères en utilisant des motifs spécifiques, ce qui est utile pour le traitement de texte et l'analyse de données textuelles [10].

11.1.1.10 heapq :

Le module `heapq` de Python implémente des algorithmes basés sur les tas (heaps), notamment pour la gestion des plus petits éléments, des plus grands éléments, et pour le tri par tas. Il offre des fonctionnalités pour créer, modifier et utiliser des tas de manière efficace [8].

11.1.1.11 collections.Counter :

Le module `Counter` de Python fournit un conteneur spécialisé pour compter les occurrences des éléments dans une séquence ou un dictionnaire. Il permet de créer des histogrammes, de trouver les éléments les plus courants, et d'effectuer des opérations arithmétiques sur les compteurs, ce qui est utile pour l'analyse de données et le traitement de texte [6].

12 Outils de développement :

12.1 Jupyter Notebook :

Le Jupyter Notebook est une application web qui permet de créer des documents interactifs intégrant du texte, du code exécutable et des visualisations. Il prend en charge plusieurs langages de programmation, notamment Python, et offre une expérience de développement interactive, permettant aux utilisateurs d'explorer les données, de créer des rapports et de collaborer avec d'autres facilement. Les notebooks Jupyter peuvent être partagés et exécutés à distance, ce qui les rend très polyvalents pour la science des

données, la recherche et l'éducation [5]. En résumé, le Jupyter Notebook est un outil essentiel pour l'analyse de données interactive et la communication des résultats.

13 Interface :

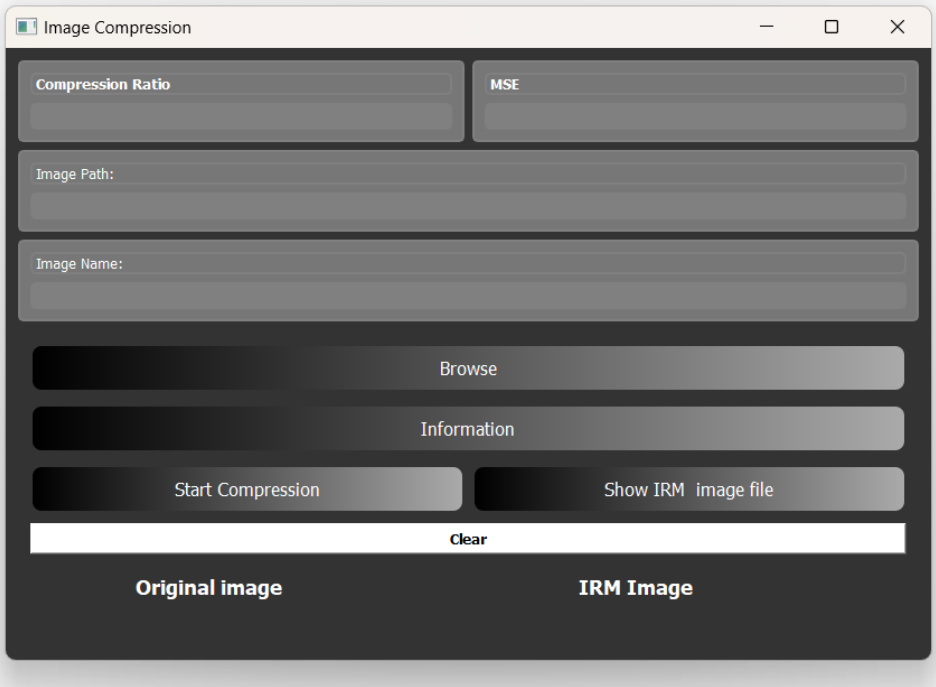

FIGURE 14 – Interface

Bouton "Browse" (Choose Button) :

Ce bouton permet à l'utilisateur de parcourir les fichiers de son système pour sélectionner une image à compresser. Une fois l'image sélectionnée, son chemin d'accès complet est affiché dans la zone de texte "Image Path" et le nom de l'image est affiché dans la zone de texte "Image Name". L'image sélectionnée est également affichée dans la fenêtre de l'application.

Bouton "Information" (Info Button) :

Ce bouton affiche des informations détaillées sur l'image sélectionnée. Les informations affichées comprennent la taille de l'image, le format, le mode de couleur, la résolution, la définition en pixels, la définition en largeur et hauteur, la taille mémoire avec compression, la taille mémoire sans compression et le taux de compression.

Bouton "Start Compression" (Compress Button) :

Ce bouton lance le processus de compression de l'image sélectionnée. Une fois la compression terminée, les informations sur le taux de compression et l'erreur quadratique moyenne (MSE) sont affichées dans les zones de texte correspondantes.

Bouton "Show IRM image file" (Decompress Button) :

Ce bouton affiche l'image compressée et décompressée dans la fenêtre de l'application. Il est activé une fois que le processus de compression est terminé et que l'image décompressée est disponible.

Bouton "Clear" (Clear Button) :

Ce bouton permet à l'utilisateur d'effacer toutes les informations affichées dans l'interface, y compris le chemin d'accès de l'image, le nom de l'image, les informations sur le taux de compression et l'erreur quadratique moyenne (MSE), ainsi que les images affichées. Cela permet à l'utilisateur de recommencer le processus avec une nouvelle image.

Les zones de texte suivantes sont utilisées pour afficher les informations sur l'image sélectionnée :

Image Path :

Affiche le chemin d'accès complet de l'image sélectionnée.

Image Name :

Affiche le nom de l'image sélectionnée.

Compression Ratio :

Affiche le taux de compression de l'image après compression.

MSE :

Affiche l'erreur quadratique moyenne entre l'image originale et l'image décompressée après compression.

13.1 Étapes :

13.1.0.1 Chercher une image :

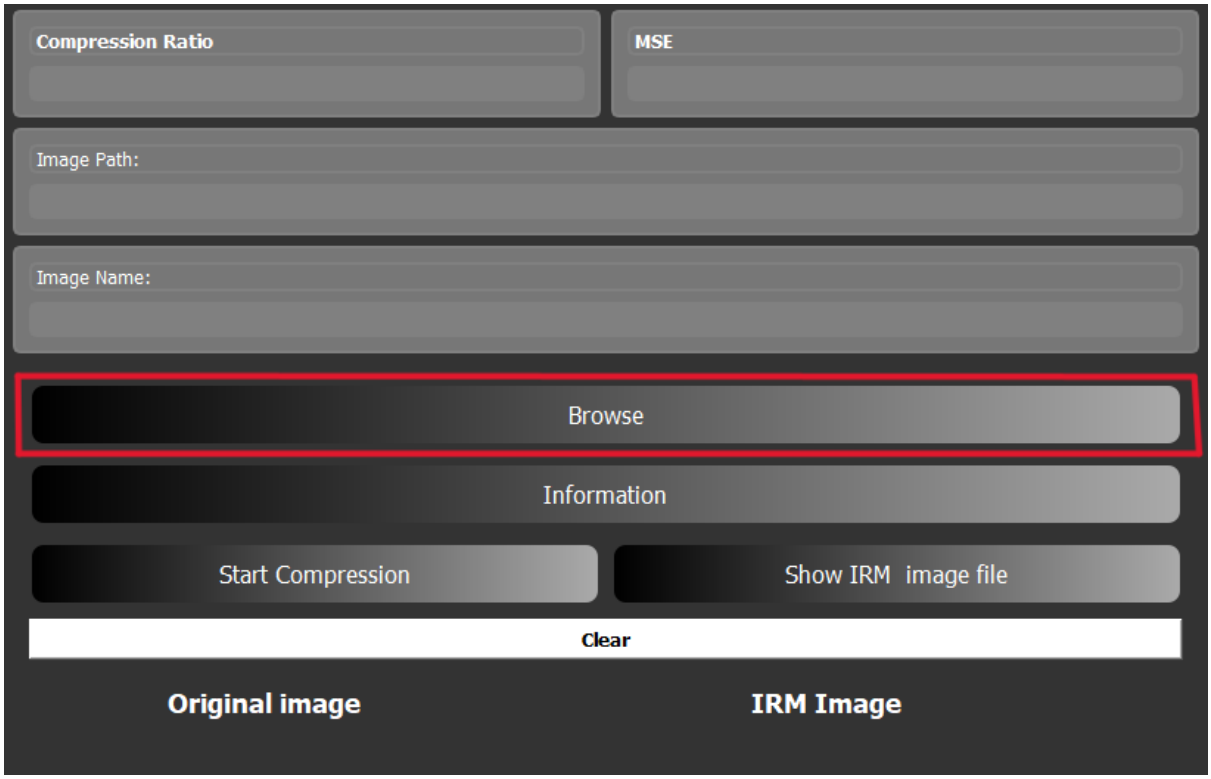

FIGURE 15 – Recherche de l'image à compresser.

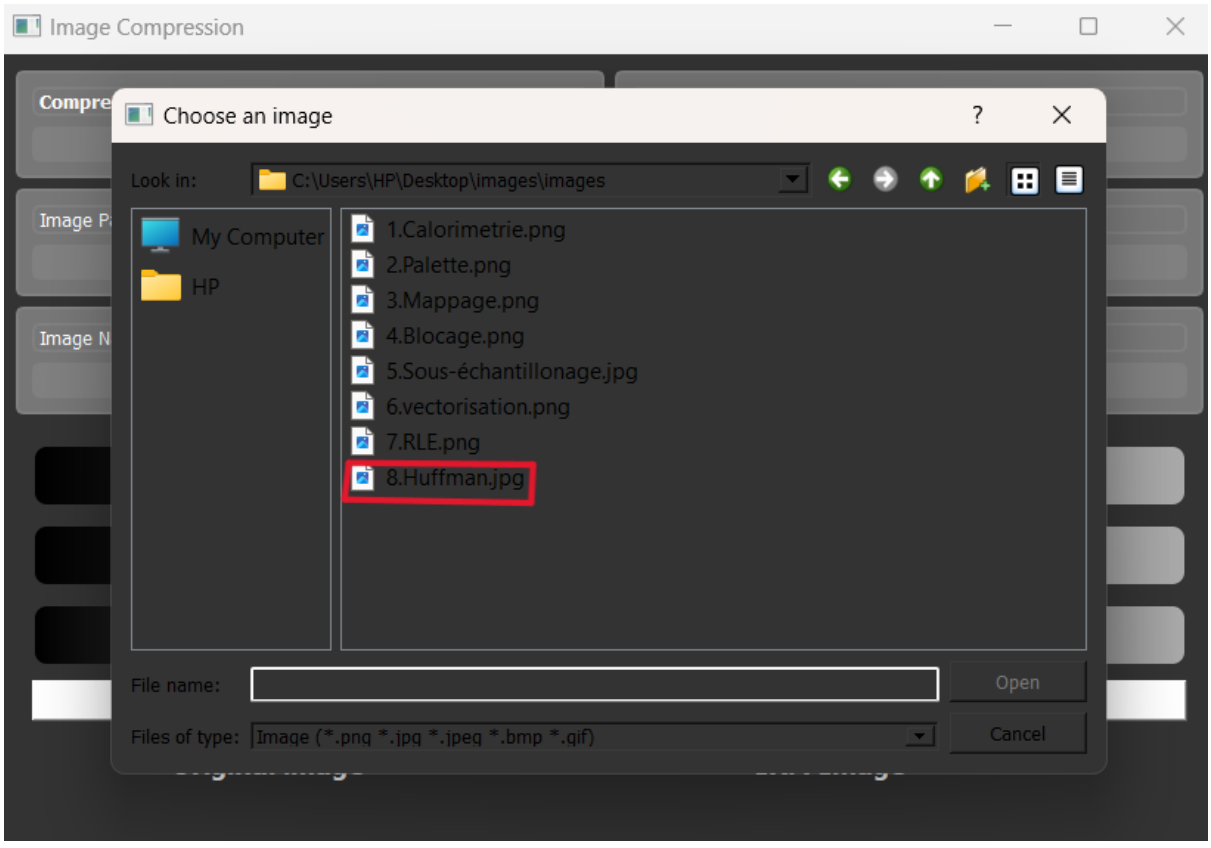

FIGURE 16 – Choisir l'image à compresser.

13.1.0.2 Information de l'image

c'est une etape optionnel

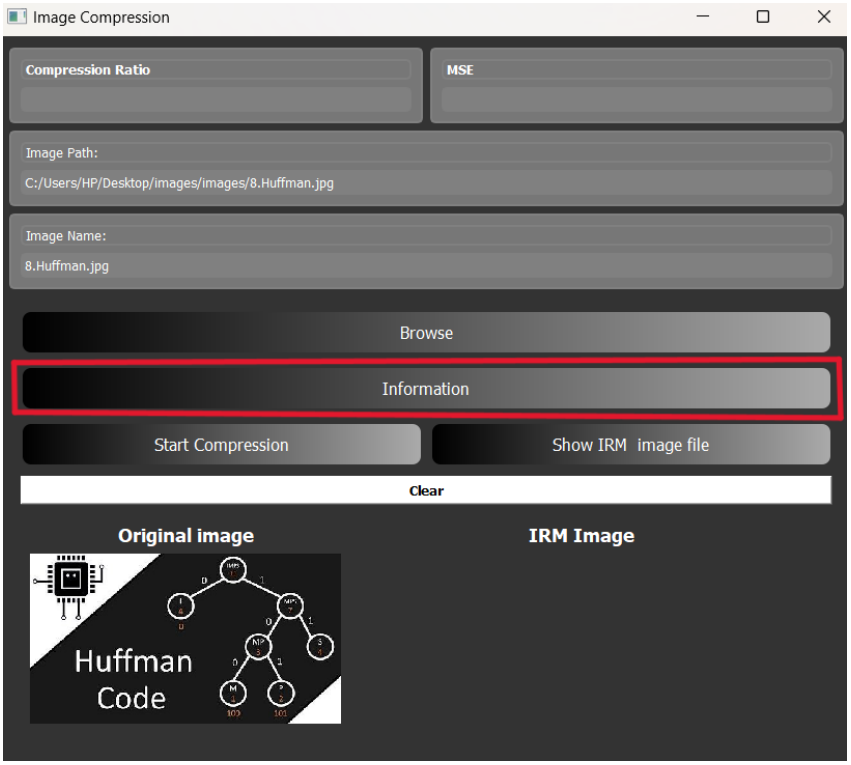

FIGURE 17 – Recherche de l'image à compresser.

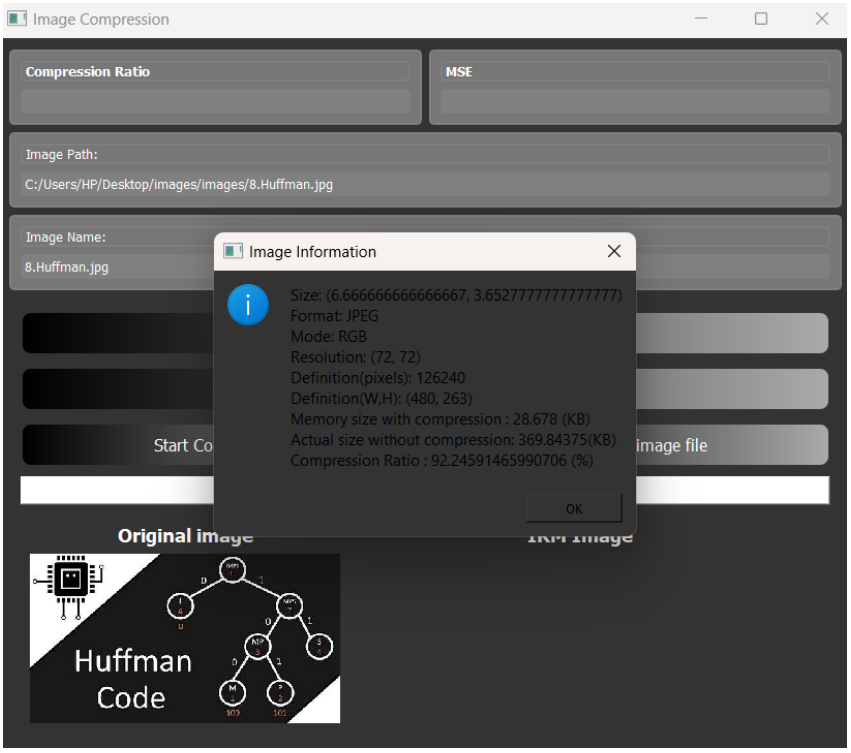

FIGURE 18 – Lecture des informations de l'image.

13.1.0.3 La compression IRM

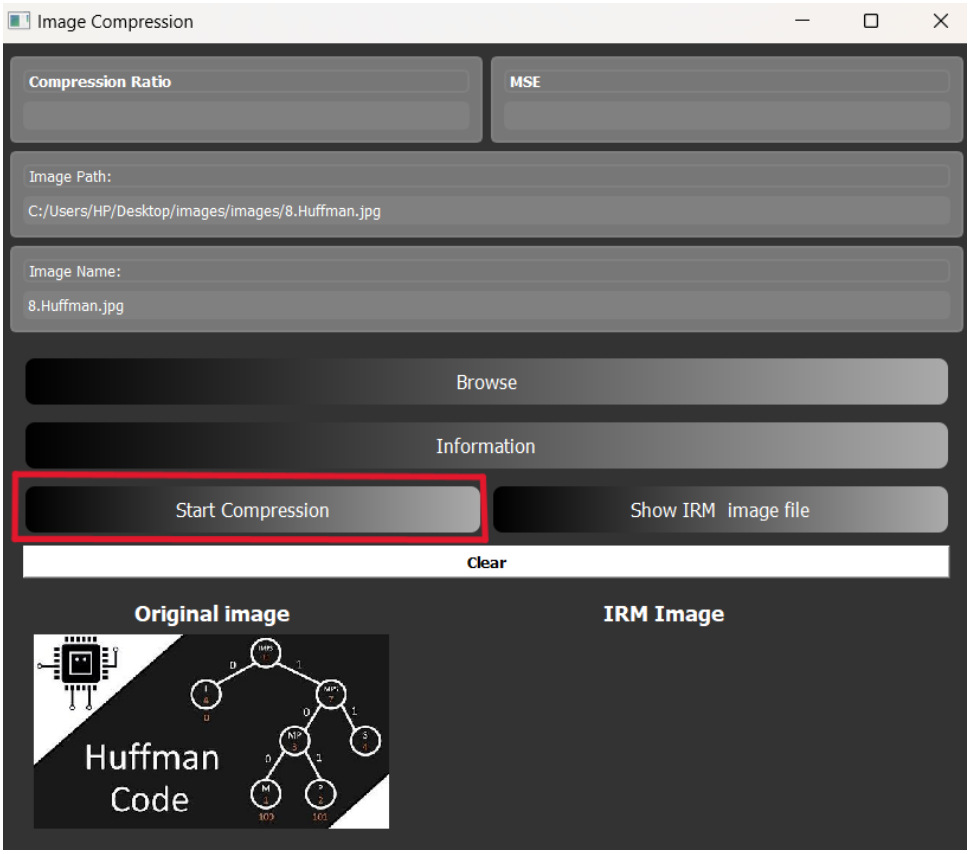

FIGURE 19 – Commencer la compression.

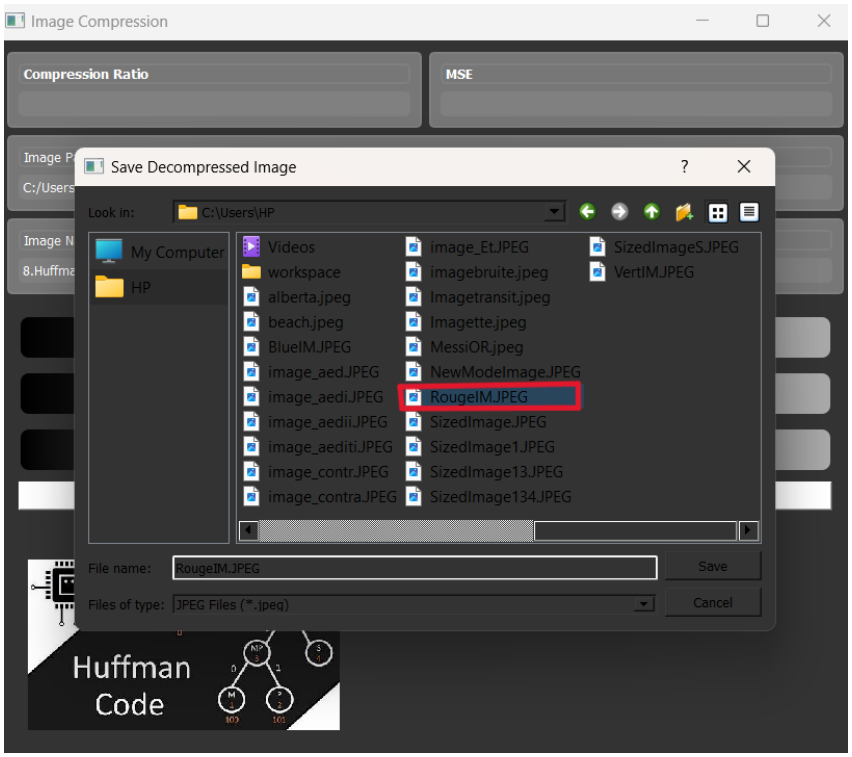

FIGURE 20 – Choisir un autre chemin pour appliquer la compression et la décompression.

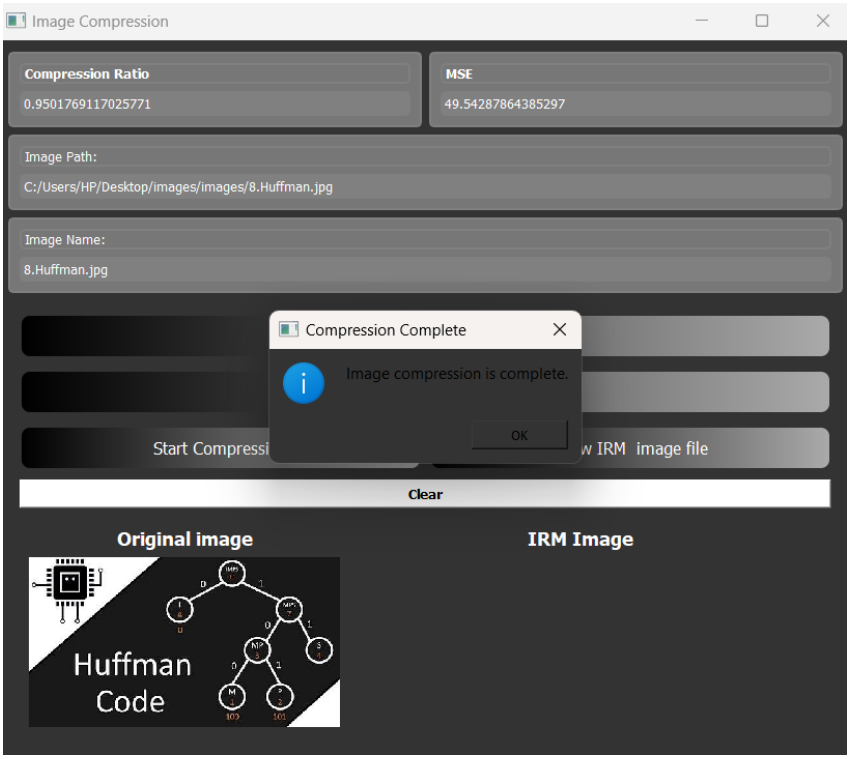

FIGURE 21 – La compression est effectuée (MSE et RLE) affichés.

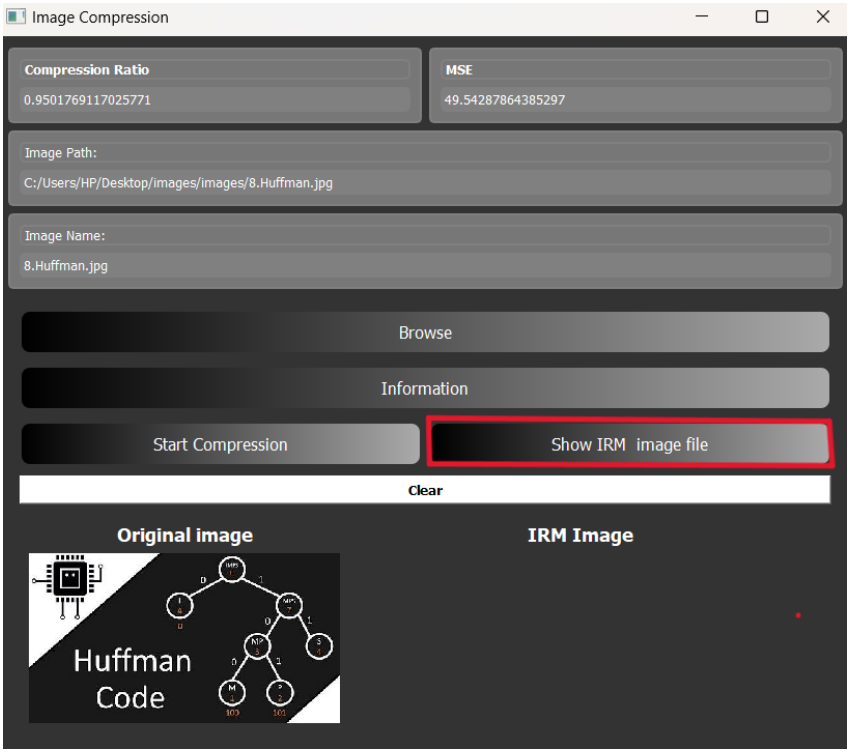

FIGURE 22 – montrer l'image IRM.

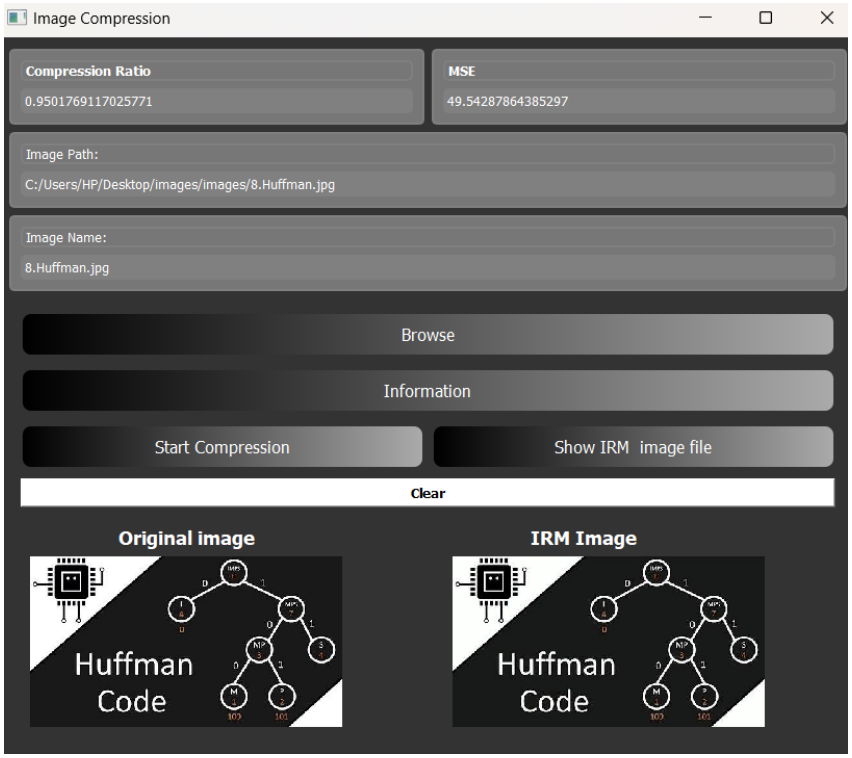

FIGURE 23 – Image IRM.

Conclusion Générale

La compression d'images est devenue un enjeu crucial dans notre société numérique où les médias visuels dominent de plus en plus notre communication et nos échanges d'information. Face à la prolifération des images et à la nécessité de les partager rapidement et efficacement, la compression d'images offre des solutions pour réduire leur taille tout en préservant une qualité visuelle acceptable.

En comprenant les principes de base de la compression d'image, qu'elle soit sans perte ou avec perte, ainsi que les différents algorithmes et techniques utilisés, il devient possible de créer des systèmes de compression efficaces adaptés à divers besoins et contraintes.

L'approche pratique de ce domaine, avec la mise en place d'algorithmes de compression et de décompression dans des langages de programmation tels que Python, permet de tester et d'évaluer ces techniques dans des conditions réelles. La comparaison des images compressées avec les originaux à l'aide de mesures telles que l'erreur quadratique moyenne (EQM) offre une évaluation quantitative de l'efficacité des algorithmes de compression.

Par ailleurs, l'innovation continue dans le domaine de la compression d'images est essentielle pour répondre aux besoins évolutifs des utilisateurs. L'introduction de nouveaux formats de compression, tels que le format IRM proposé, qui intègrent des techniques de compression avancées telles que les codages statistiques RLE, Huffman et LZW, montre la volonté d'améliorer constamment l'efficacité et la qualité de la compression d'images.

Enfin, la création d'interfaces utilisateur conviviales pour permettre aux utilisateurs de compresser facilement leurs images et d'évaluer la qualité de la compression réalisée est un aspect crucial de l'adoption et de l'utilisation efficace des techniques de compression d'images.

En conclusion, la compression d'images est un domaine en constante évolution, offrant des solutions techniques sophistiquées pour répondre aux défis posés par la prolifération des médias visuels dans notre société numérique, tout en cherchant à améliorer constamment l'efficacité, la qualité et l'accessibilité des techniques de compression pour les utilisateurs.

Bibliographie

[1] The history of python. https://www.python.org/doc/essays/blurb/.

[2] Matplotlib Development Team. *Matplotlib Documentation*, 2022. https://matplotlib.org/stable/contents.html.

[3] NumPy Developers. *NumPy Documentation*, 2022. https://numpy.org/doc/stable/.

[4] Pillow Developers. *Python Imaging Library (PIL) Documentation*, 2022. https://pillow.readthedocs.io/en/stable/.

[5] Project Jupyter. *Jupyter Documentation*, 2022. https://jupyter.org/documentation.

[6] Python Software Foundation. *Python collections Module Documentation*, 2022. https://docs.python.org/3/library/collections.html.

[7] Python Software Foundation. *Python Documentation*, 2022. https://docs.python.org/3/.

[8] Python Software Foundation. *Python heapq Module Documentation*, 2022. https://docs.python.org/3/library/heapq.html.

[9] Python Software Foundation. *Python os Module Documentation*, 2022. https://docs.python.org/3/library/os.html.

[10] Python Software Foundation. *Python re Module Documentation*, 2022. https://docs.python.org/3/library/re.html.

[11] Riverbank Computing Limited. *PyQt5 Documentation*, 2022. https://www.riverbankcomputing.com/static/Docs/PyQt5/.

[12] Scikit-learn Developers. *Scikit-learn Documentation*, 2022. https://scikit-learn.org/stable/documentation.html.

[13] SciPy Developers. *SciPy Documentation*, 2022. https://docs.scipy.org/doc/scipy/reference/.