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 **SANGARE**

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 me poussent à la 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 notre professeur et encadrant **Mr. Abdellah ADIB** pour son aide avec son expérience et savoirfaire.
Nous la prions de croire en notre respectueuse estime et notre sincère reconnaissance pour ses conseils qui nous ont été très précieux et indispensables.
Merci infiniment.

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 :

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.

4.1.1.2 Applications :

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 "1R2E2L1E1M1E1N1T" ; 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.

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.

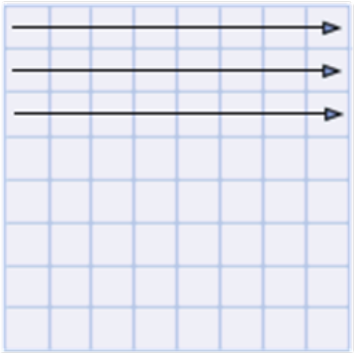

FIGURE 1 – Ligne

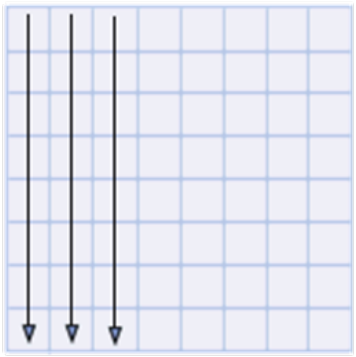

FIGURE 2 – Colonne

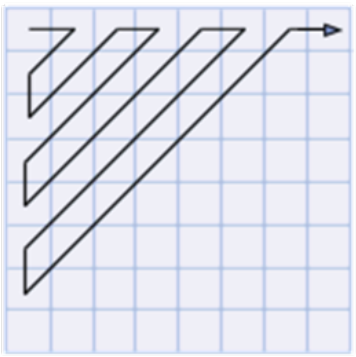

FIGURE 3 – ZigZag

4.1.2 Huffman

4.1.2.1 Généralités et principes :

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.
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.

4.1.2.2 Applications :

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

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

Voici l'arbre correspondant :

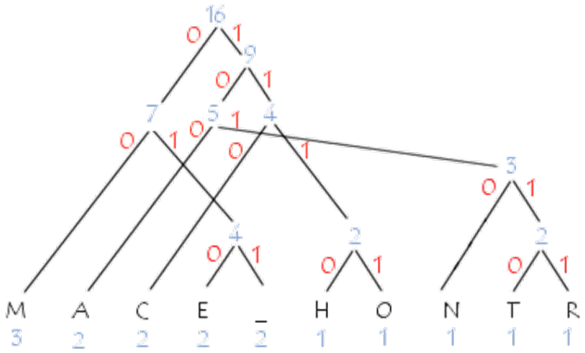

FIGURE 4 – Arbre

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.

4.2 Codage dynamique (utilisant un dictionnaire)

4.2.1 LZ7W :

4.2.1.1 Généralités et principes :

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. 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.

4.2.1.2 Applications :

- 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 LZ8 :

4.2.2.1 Généralités et principes :

L'algorithme LZ78 est un algorithme de compression sans perte. Le fichier compressé est composé du dictionnaire et du fichier où les séquences de données ont été remplacées par des codes.

4.2.2.2 Applications :

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.

5 Notion de compression d'images :

5.1 Définition, généralités et principes :

5.1.1 Classification :

5.1.1.1 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 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 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 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 :

Échantillonnage de Couleur :

L'image est souvent convertie de l'espace de couleur RGB en l'espace de couleur YCbCr, ce qui sépare la luminance (Y) des composantes de chrominance (Cb et Cr). Cela permet de réduire la quantité de données à traiter tout en maintenant une qualité d'image acceptable.

Découpage en Blocs :

L'image est divisée en blocs de pixels, généralement de taille 8x8 pixels.

Transformation en Cosinus Discrète (DCT) :

Chaque bloc est transformé en domaine de fréquence à l'aide de la DCT, une technique qui permet de représenter l'information spatiale sous forme de fréquences.

Quantification :

Les coefficients de la DCT sont quantifiés pour réduire la précision des valeurs. Cette étape permet de supprimer les détails moins visibles de l'image.

Codage par Huffman :

Les coefficients quantifiés sont ensuite compressés à l'aide d'un codage par Huffman, qui attribue des codes binaires courts aux symboles fréquents et des codes binaires plus longs aux symboles moins fréquents.

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 :

Indexation des Couleurs :

Chaque pixel de l'image est associé à un index correspondant à une couleur dans la palette de couleurs.

Réduction de la Palette :

Si l'image contient plus de couleurs que la taille maximale autorisée dans la palette, une réduction de la palette est effectuée pour sélectionner les couleurs les plus représentatives de l'image.

Compression LZW :

Les données de l'image sont compressées en utilisant l'algorithme LZW. Cet algorithme recherche et remplace les motifs répétitifs dans les données par des références à des motifs déjà rencontrés, ce qui permet de réduire la taille du fichier sans perte de qualité.

Écriture des Données :

Les données compressées sont écrites dans le fichier GIF avec les informations d'entête nécessaires pour la décompression.

7 Étapes des algorithmes de compression JPEG et GIF :

7.1 JPEG :

Les étapes typiques de compression JPEG comprennent :

Conversion en Espace de Couleur YCbCr :

L'image est généralement convertie de l'espace de couleur RGB en espace de couleur YCbCr pour séparer la luminance des composantes de chrominance.

Découpage en Blocs :

L'image est divisée en blocs de pixels, généralement de taille 8x8 pixels.

Transformation en Cosinus Discrète (DCT) :

Chaque bloc est transformé en domaine de fréquence à l'aide de la DCT.

Quantification :

Les coefficients de la DCT sont quantifiés pour réduire la précision des valeurs.

Codage par Huffman :

Les coefficients quantifiés sont compressés à l'aide d'un codage par Huffman.

Entrelacement et Écriture des Données :

Les données compressées sont organisées et écrites dans le fichier JPEG de manière à optimiser la compression et la décompression.

7.2 GIF :

Les étapes typiques de compression GIF comprennent :

Indexation de la Couleur :

Chaque pixel de l'image est associé à un index correspondant à une couleur dans la palette de couleurs.

Réduction de la Palette :

Si nécessaire, une réduction de la palette est effectuée pour sélectionner les couleurs les plus représentatives de l'image.

Compression LZW :

Les données de l'image sont compressées en utilisant l'algorithme LZW.

Écriture des Données :

Les données compressées sont écrites dans le fichier GIF avec les informations d'entête nécessaires 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 Formalisme de l'algorithme de compression IRM :

9.1 Entête :

9.2 Données :

10 Étapes de l'algorithme de compression IRM :

10.1 Création de la palette de couleurs :

FIGURE 5 – Palette de Couleur

Dans le contexte de la compression d'image, la création de la palette de couleurs est une étape cruciale, principalement utilisée dans le format GIF. Cette étape implique la sélection d'un ensemble limité de couleurs qui seront utilisées pour représenter l'image. La palette de couleurs est généralement limitée à 256 couleurs dans le format GIF. Pour créer cette palette, différentes techniques peuvent être utilisées, telles que la quantification des couleurs, l'échantillonnage, ou même la sélection manuelle des couleurs les plus représentatives de l'image.

10.2 Mappage :

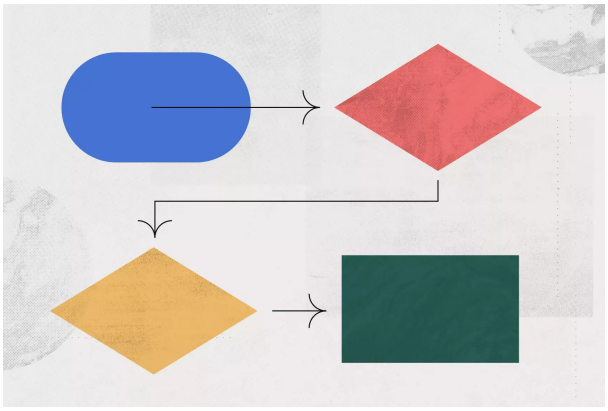

FIGURE 6 – Mappage

Le mappage fait référence à l'association de chaque pixel de l'image à une couleur de la palette créée dans l'étape précédente. Chaque pixel de l'image

est remplacé par l'index de la couleur correspondante dans la palette de couleurs. Cette étape permet de réduire la quantité d'informations nécessaires pour représenter chaque pixel, ce qui contribue à la compression de l'image.

10.3 Subdivisions en blocs de pixels :

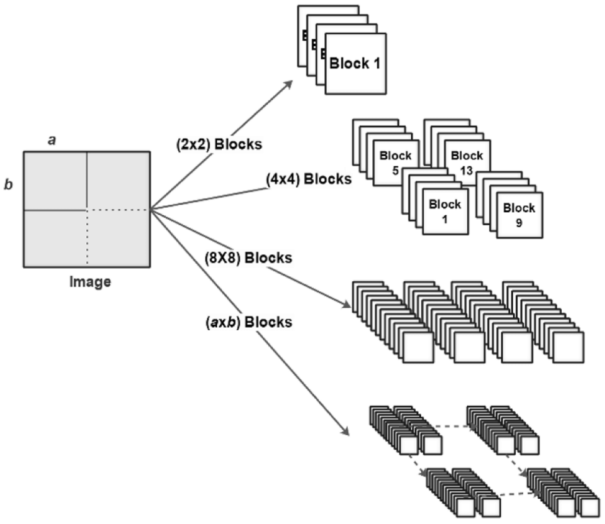

FIGURE 7 – Subdivision

Cette étape consiste à diviser l'image en blocs de pixels de taille fixe. Dans de nombreux algorithmes de compression, notamment dans JPEG, les blocs sont généralement de taille 8x8 pixels. Cette subdivision permet de traiter l'image par petits blocs, ce qui facilite l'application de certaines techniques de compression, telles que la transformation en cosinus discrète (DCT).

10.4 Sous échantillonnage :

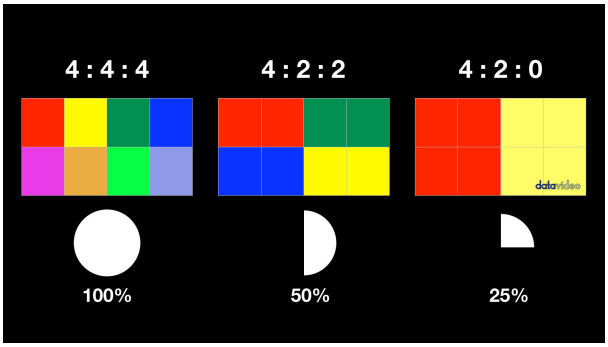

FIGURE 8 – Sous-Echantillonnagr

Le sous-échantillonnage est une technique utilisée dans certains algorithmes de compression, comme JPEG, pour réduire la quantité de données de chrominance. Étant donné que l'œil humain est moins sensible aux changements de chrominance qu'à ceux de luminance, le sous-échantillonnage consiste à réduire la résolution des composantes de chrominance de manière sélective, tout en préservant la résolution de la luminance. Cela permet de réduire la taille des données sans compromettre de manière significative la qualité visuelle de l'image.

10.5 DCT :

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left(\frac{(2x+1)i\pi}{2N}\right) \cos\left(\frac{(2y+1)j\pi}{2N}\right)$$

- N : la largeur d'un bloc, ici $N = 8$.
- i, j : les indices d'un coefficient de la DCT dans un bloc.
- x, y : les indices d'un pixel de l'image dans un bloc.
- $DCT(i, j)$: la valeur d'un coefficient dans un bloc.
- $C(x) = \frac{1}{\sqrt{2}}$ si $x = 0, C(x) = 0$ sinon.

FIGURE 9 – DCT

La DCT est une transformation utilisée dans la compression JPEG pour convertir les blocs de pixels de l'image de leur représentation spatiale à une représentation fréquentielle. Chaque bloc est transformé en une série de coefficients de fréquence qui représentent la contribution de différentes fréquences spatiales au contenu de l'image. La DCT permet de concentrer l'énergie de l'image dans un petit nombre de coefficients, ce qui facilite leur quantification ultérieure.

10.6 Quantification :

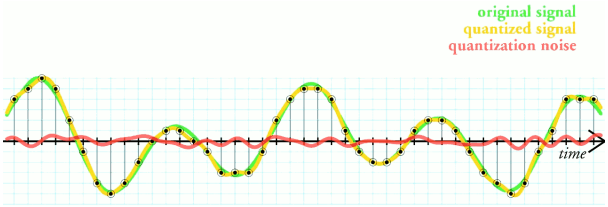

FIGURE 10 – Quantification

La quantification est une étape clé de la compression JPEG où les coefficients de la DCT sont arrondis à des valeurs discrètes. Cette étape permet de réduire la précision des coefficients, ce qui entraîne une perte d'information, mais permet également de réduire la taille des données. Les coefficients sont quantifiés en utilisant des tables de quantification spécifiques qui déterminent le niveau de compression et la qualité de l'image finale.

10.7 Vectorisation :

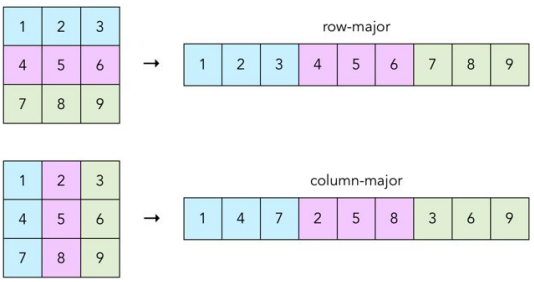

FIGURE 11 – Vectorisation

Dans certains algorithmes de compression, notamment ceux utilisés pour la compression d'images à base de vecteurs, une étape de vectorisation est effectuée. Cette étape consiste à représenter les formes ou les motifs récurrents de l'image sous forme de vecteurs, ce qui permet une représentation plus compacte et efficace de l'image. La vectorisation peut être utilisée dans le cadre de la compression d'images simples ou d'images contenant des formes géométriques répétitives.

10.8 Codage de Huffman :

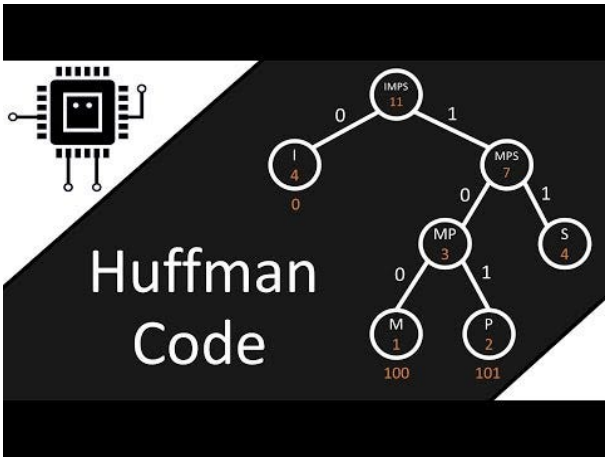

FIGURE 12 – Huffman

Le codage de Huffman est une technique de compression sans perte largement utilisée dans de nombreux formats d'image, y compris JPEG et GIF. Cette technique consiste à remplacer les symboles les plus fréquents par des codes binaires plus courts et les symboles moins fréquents par des codes binaires plus longs. Ces codes sont déterminés en fonction de la fréquence d'apparition des symboles dans les données. Le codage de Huffman permet une compression efficace en réduisant la longueur moyenne des codes nécessaires pour représenter les données.

10.9 Codage LZW de ce code Huffman

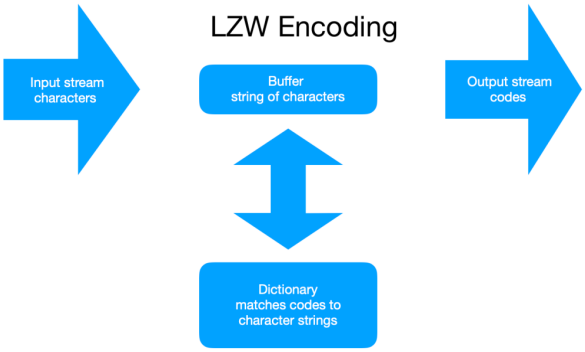

FIGURE 13 – LZW

Le codage LZW (Lempel-Ziv-Welch) est une méthode de compression sans perte utilisée dans le format GIF pour compresser les données après qu'elles ont été codées par Huffman. Cette méthode consiste à rechercher et

à remplacer les motifs répétitifs dans les données par des références à des motifs déjà rencontrés. Le codage LZW fonctionne en créant un dictionnaire de motifs rencontrés au fur et à mesure que les données sont traitées, ce qui permet une compression plus efficace des données répétitives. Une fois les données codées par Huffman, le codage LZW est appliqué pour réduire davantage la taille du fichier sans perte de qualité.

Développement de l'interface

11 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.

12 Langage de développement :

12.1 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. 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 à la prototypage.

En raison de ses nombreuses qualités, Python est souvent recommandé comme premier langage de programmation pour les débutants.

12.1.1 Bibliothèques et modules :

12.1.1.1 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.

12.1.1.2 skimage.io :

Scikit-image est une bibliothèque Python dédiée au traitement d'images. Son module io fournit des fonctionnalités pour lire et écrire des images dans différents formats, notamment PNG, JPEG, TIFF, etc. Il permet également de charger des images à partir d'URLs ou de les convertir en tableaux NumPy pour une manipulation ultérieure.

12.1.1.3 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.

12.1.1.4 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. Il prend en entrée le nombre de clusters désiré et les données à regrouper, puis retourne les centres des clusters et les étiquettes associées à chaque point de données.

12.1.1.5 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.

12.1.1.6 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.

12.1.1.7 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.

12.1.1.8 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.

12.1.1.9 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.

12.1.1.10 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.

13 Outils de développement :

13.1 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. En résumé, le Jupyter Notebook est un outil essentiel pour l'analyse de données interactive et la communication des résultats.

14 Interface :