Logo

Description automatically generated

INF8770

Technologies multimédias

*Travail pratique #2 – Méthodes de codage*

Soumis par :

*Gaudet, Alexandre – 2090935*

*Sant’anna, Abdel – 2088865*

*Le 18 février 2023*

**Question 1**

**Question 2**

Pour concevoir l’algorithme 2, nous nous sommes inspirés du code source fournit dans le repo d’exemples, notamment le programme de transformée KL, d’évaluation et de quantification. Nous avons choisi la quantification 8/4/0 en plus des 3 quantifications de l’énoncé. Nous avons récupéré les données d’évaluations pour les 40 compressions et les avons regroupés dans les tableaux ci-dessous.

|  |  |  |  |
| --- | --- | --- | --- |
| **Images** | **Compression** | **PSNR** | **SSIM** |
| 1-YUV | 0.33 | 42.74 | 0.99 |
| 2-YUV | 0.33 | 32.96 | 0.99 |
| 5-YUV | 0.33 | 35.99 | 0.97 |
| 13-YUV | 0.33 | 39.84 | 0.99 |
| 23-YUV | 0.33 | 34.56 | 0.97 |
| 1-RGB | 0.33 | 45.03 | 1 |
| 2-RGB | 0.33 | 42.73 | 1 |
| 5-RGB | 0.33 | 37.06 | 0.97 |
| 13-RGB | 0.33 | 39.28 | 0.99 |
| 23-RGB | 0.33 | 37.58 | 0.97 |

Résultats de l’algorithme 1 avec la quantification 8/4/4

|  |  |  |  |
| --- | --- | --- | --- |
| **Images** | **Compression** | **PSNR** | **SSIM** |
| 1-YUV | 0.17 | 48.53 | 1 |
| 2-YUV | 0.17 | 44.3 | 1 |
| 5-YUV | 0.17 | 40.52 | 0.99 |
| 13-YUV | 0.17 | 40.43 | 0.99 |
| 23-YUV | 0.17 | 37.94 | 0.98 |
| 1-RGB | 0.17 | 49.87 | 1 |
| 2-RGB | 0.17 | 49.67 | 1 |
| 5-RGB | 0.17 | 39.45 | 0.98 |
| 13-RGB | 0.17 | 39.54 | 0.99 |
| 23-RGB | 0.17 | 42.82 | 0.99 |

Résultats de l’algorithme 1 avec la quantification 8/8/4

|  |  |  |  |
| --- | --- | --- | --- |
| **Images** | **Compression** | **PSNR** | **SSIM** |
| 1-YUV | 0 | 51.25 | 1 |
| 2-YUV | 0 | 51.01 | 1 |
| 5-YUV | 0 | 51.06 | 1 |
| 13-YUV | 0 | 51.2 | 1 |
| 23-YUV | 0 | 51.29 | 1 |
| 1-RGB | 0 | 52.2 | 1 |
| 2-RGB | 0 | 50.78 | 1 |
| 5-RGB | 0 | 51.16 | 1 |
| 13-RGB | 0 | 50.97 | 1 |
| 23-RGB | 0 | 51.13 | 1 |

Résultats de l’algorithme 1 avec la quantification 8/4/8

|  |  |  |  |
| --- | --- | --- | --- |
| **Images** | **Compression** | **PSNR** | **SSIM** |
| 1-YUV | 0.5 | 35.75 | 0.98 |
| 2-YUV | 0.5 | 22.65 | 0.91 |
| 5-YUV | 0.5 | 21.37 | 0.68 |
| 13-YUV | 0.5 | 17.11 | 0.61 |
| 23-YUV | 0.5 | 18.74 | 0.75 |
| 1-RGB | 0.5 | 37.73 | 0.98 |
| 2-RGB | 0.5 | 39.59 | 1 |
| 5-RGB | 0.5 | 26.2 | 0.78 |
| 13-RGB | 0.5 | 17.71 | 0.62 |
| 23-RGB | 0.5 | 23.53 | 0.8 |

Résultats de l’algorithme 1 avec la quantification 8/4/0

Tableau 1. Performance de LZW

|  |  |  |
| --- | --- | --- |
| Source | Taux de compression | Temps d’encodage (secondes) |
| texte\_1 | 0,119 | 0,0010 |
| texte \_2 | 0,266 | 0,0010 |
| texte \_3 | 0,108 | 0,0010 |
| texte \_4 | 0,001 | 0,0010 |
| texte \_5 | 0,266 | 0,0024 |
| image\_1 | -0,332 | 0,2940 |
| image \_2 | 0,969 | 0,0366 |
| image \_3 | 0,560 | 0,6989 |
| image \_4 | 0,151 | 0,9969 |
| image \_5 | 0,986 | 0,1697 |

**Question 3**

L’utilisation de LZW sur les données fournies a connu quelques limitations. Premièrement, pour l’image 1, le taux de compression obtenu est négatif, ce qu’il est assurément souhaitable d’éviter. Deuxièmement, les taux de compression des fichiers textes sont bas, surtout comparativement aux images. Le fait que les textes sont très courts rend plus difficile de trouver de longues sous-chaines et profiter au maximum des avantages de LZW. Il est surement possible de faire mieux à ce niveau.

En tenant compte de ces faiblesses, le codage Huffman, qui est basé sur l’entropie, semble être une alternative intéressante. Cet algorithme permet de coder les caractères les plus fréquents avec de petits codes binaires. Dans le cas d’une distribution parfaitement uniforme, Huffman est équivalent au codage binaire. Cela permet d’éviter d’avoir une compression négative pour l’image 1, qui se rapproche d’une distribution uniforme et cause des problèmes avec LZW en raison du manque de répétition. Aussi, Huffman produit le meilleur code binaire possible basé sur la fréquence de chaque caractère. Son efficacité n’est donc pas limitée par la longueur des textes comme c’était le cas pour LZW. De plus, plusieurs des textes ont des lettres plus fréquentes. Par exemple, le texte 3 contient beaucoup de « O », mais très peu de « B » ou de « D ». Finalement, Huffman devrait aussi bien fonctionner sur certaines des images. Notamment, l’image 3 et l’image 5 qui ont respectivement des fonds uniformes blanc et noir, ce qui implique une haute fréquence de faibles ou hautes valeurs RGB.

L’implémentation choisie est directement basée sur celle utilisée dans le cadre du cours [2]. Le tableau 2 présente les taux de compression obtenus avec Huffman. Comme attendu, l’algorithme a bien fonctionné pour l’image 3 et 5. L’image 1 a aussi un taux de compression positif. Les textes ont un bon taux de compression, autre que le texte 1 pour lequel Huffman ne semble pas être optimal.

Le tableau 2 présente aussi le temps d’encodage pour chaque fichier. Celui-ci est similaire à LZW pour les textes, mais Huffman est généralement plus couteux en termes de temps pour les images. Contrairement à LZW, il y a un coût supplémentaire pour construire l’arbre et générer les codes préfixes avant de coder les données.

Tableau 2. Performance de Huffman

|  |  |  |
| --- | --- | --- |
| Source | Taux de compression | Temps d’encodage (secondes) |
| texte\_1 | 0,000 | 0,0010 |
| texte \_2 | 0,234 | 0,0010 |
| texte \_3 | 0,358 | 0,0010 |
| texte \_4 | 0,188 | 0,0010 |
| texte \_5 | 0,300 | 0,0022 |
| image\_1 | 0,084 | 0,7829 |
| image \_2 | 0,000 | 0,0173 |
| image \_3 | 0,464 | 3,4592 |
| image \_4 | 0,031 | 3,0442 |
| image \_5 | 0,695 | 0,4126 |

**Question 4**

Il convient maintenant de comparer les performances des deux méthodes de compression. La figure 1 présente les taux de compression obtenus par chacun des algorithmes à titre de comparaison. Pour les fichiers textes, le codage Huffman obtient un meilleur taux de compression pour les numéros 3, 4 et 5, alors que LZW a un meilleur taux pour les numéros 1 et 2. La méthode la plus constante est certainement Huffman qui obtient un taux supérieur à 0,15 pour quatre textes, contre seulement deux pour LZW. Le tableau 3 présente le gain moyen de compression, qui est obtenu en faisant la moyenne des différences entre les taux de compression. Ainsi, en moyenne, l’utilisation de Huffman au lieu de LZW aboutit en un gain moyen de 0,064 pour les fichiers utilisés. Considérant ces facteurs, Huffman est la méthode la plus performante pour les textes. Ceci n’est pas surprenant, considérant que LZW est quelque peu limité par la longueur des textes. En effet, sur des textes plus courts, il est moins possible d’encoder de longues sous-chaines à la fois. À l’inverse, Huffman n’est pas limité par ce facteur et peu pleinement tirer avantage des fréquences plus élevées de certaines lettres. La seule limitation de Huffman est sur le texte 1, car le petit alphabet (2 caractères) fait que Huffman est équivalent au code binaire classique.

Pour les images, Huffman obtient seulement un meilleur taux de compression pour l’image 1, alors que LZW est meilleur pour le reste. C’est aussi LZW qui a le plus d’images avec un taux supérieur à 0,15, avec quatre comparativement à deux pour Huffman. De plus, en moyenne, le gain de compression est négatif en passant de LZW à Huffman pour les images, tel que présenté au tableau 3. En considérant ces facteurs, LZW semble être le meilleur choix pour les images. Cependant, il faut noter que le taux de compression de LZW est nettement négatif pour l’image 1. Si c’est quelque chose qu’il faut absolument éviter, il pourrait être plus intéressant de considérer Huffman.

La grande efficacité de LZW sur les images fournies peut s’expliquer par la capacité à coder de très longues sous-chaines de pixels qui se répètent lors de l’utilisation de couleurs uniformes. Notamment, sur les images 2 et 5, le fond noir et le cercle rouge sont extrêmement bien codés par LZW qui atteint d’impressionnants taux de compression supérieurs à 0,95. Bien que Huffman soit efficace sur l’image 5 due aux répétitions plus fréquentes de certains pixels, il est limité par le fait de coder un seul symbole à la fois et ne peut donc pas atteindre des taux aussi élevés que LZW, qui en code plusieurs ensemble. Cependant, LZW a aussi la limitation d’être très mauvais dans le cas où il y a très peu de répétition spatiale et produit un taux de compression négatif, il est fortement dépendant de l’axe par lequel on décide de l’utiliser afin de favoriser la redondance. À l’inverse, Huffman est équivalent au code binaire classique dans le pire cas (image 2).

Figure 1. Comparatif des taux de compression

Tableau 3. Gain moyen de compression avec Huffman

|  |  |
| --- | --- |
| Type de données | Gain moyen de compression |
| Textes | 0,064 |
| Images | - 0,212 |

Références

[1] Bilodeau, G (2020) Codage LZW [Code source]. <https://github.com/gabilodeau/INF8770/blob/master/Codage%20LZW.ipynb>

[2] Bilodeau, G (2024) Codage Huffman [Code source]. <https://github.com/gabilodeau/INF8770/blob/master/Codage%20Huffman.ipynb>