



UNIVERSITÉ PARIS 8 - VINCENNES - SAINT-DENIS

Rapport C K-means réduction de couleurs

Raphaël MOSCATELLI

LICENCE INFORMATIQUE ET VIDÉOLUDISME ALGORITHMIQUE AVANCÉE
TROISIÈME ANNÉE
2023/2024

Table des matières

1 Présentation Projet	4
1.1 Introduction	5
2 Etat de l'art	6
2.1 Algorithmes de compression d'images	7
2.2 Méthodes de réduction du nombre de couleurs	7
2.3 Techniques de dithering	7
3 Conception du projet	9
3.1 Choix algorithmiques	9
3.2 Description fonctionnelle	10
3.3 Structure générale du programme	10
4 Implémentation	11
4.1 Algorithme de k-means	11
4.2 Création de la CLUT	11
4.3 Codage/Décodage RLE	11
4.4 Gestion mémoire	12
4.5 Réutilisation de fichiers	12
5 Tests et validation	13
5.1 Méthodologie de test	13
5.2 Résultats	13
5.3 Images	13

<i>TABLE DES MATIÈRES</i>	3
6 Analyse et conclusion	17
6.1 Bibliographie	18

Chapitre 1

Présentation Projet

*

1.1 Introduction

Ce rapport vise à présenter le projet de développement d'un programme de compression d'images réalisé dans le cadre du cours d'algorithme, en langage C.

Ce projet a été développé de manière seule et incrémentale au fil de plusieurs itérations et de nombreux essais, dans le but de déboguer et améliorer pas à pas le code source. Les images tests utilisées sont au format PPM.

L'image numérique occupe une place considérable dans le stockage de données. Sa compression permet de réduire cette empreinte mémoire. L'objectif de ce projet était d'implémenter les principales étapes de la compression et décompression d'une image fixe :

1. Quantification des couleurs via l'algorithme des k-means
2. Création d'une table d'indexation des couleurs (CLUT)
3. Compression et décompression de l'image selon le codage RLE
4. Décompression et affichage

En outre, la fiche projet stipulait :

1. L'application éventuelle d'un algorithme de dithering
2. La validation des performances en termes de qualité et ratios de compression
3. L'implémentation des fonctionnalités sous forme de menu interactif

Ce rapport a pour but de synthétiser la démarche de conception, d'implémentation et de test du programme, en détaillant les choix techniques effectués et en identifiant les perspectives d'amélioration.

Chapitre 2

Etat de l'art

*

2.1 Algorithmes de compression d'images

Les principaux algorithmes de compression d'images sont :

JPEG : basé sur la transformation en cosinus discrète et la quantification des coefficients.
Cible les images photographiques.

PNG : sans perte, utilise la méthode LZW de compression par dictionnaire. Adapté aux diagrammes, logos.

GIF : sans perte, utilise la méthode LZW. Peu adapté aux photos mais largement utilisé sur le web.

WebP : développé par Google, plus performant que PNG et JPEG pour les images sans texte.

Ces formats utilisent généralement une réduction du nombre de couleurs comme pré-traitement.

2.2 Méthodes de réduction du nombre de couleurs

Les principales techniques sont :

Quantification (seuillage) : réduit le nombre de niveaux de gris ou de couleurs.

K-means : regroupe les pixels par clusters en fonction de leur couleur. Utilisé par JPEG.

Médiane cut : divise récursivement l'espace colorimétrique.

2.3 Techniques de dithering

Petit rappel de définition de dithering : La diffusion d'erreur est une technique d'amélioration du rendu des images infographiques en cas de diminution du nombre de codes de couleurs. Elle se combine à l'anticrénelage. Elle sert dans les systèmes informatiques capables d'afficher seulement un petit nombre de couleurs, et dans les imprimantes, pour lesquelles les pixels n'ont que deux valeurs, avec du toner ou sans.

Elle consiste à répartir sur les pixels voisins les erreurs de quantification issues de l'assimilation de la couleur d'un pixel à une couleur prise dans une gamme réduite.

Le dithering permet de lisser les erreurs de quantification :

Diffusion d'erreur (Floyd-Steinberg) : propage l'erreur sur les pixels voisins.

Bayer dithering : utilise une matrice de points bleus et rouges.

Dot diffusion : placement aléatoire des points.

Chapitre 3

Conception du projet

3.1 Choix algorithmiques

Pour la quantification des couleurs, j'ai choisi l'algorithme des k-means car c'est une méthode efficace permettant de regrouper les pixels en un nombre paramétrable de classes de couleurs représentatives de l'image, en un temps raisonnable. Cet algorithme itératif recalcule à chaque étape la moyenne des couleurs de chaque classe pour affiner les regroupements.

Mon choix c'est aussi influer sur k-means plutôt que par exemple BSP ou Octrees car je connaissais déjà le fonctionnement de k-means avec les Clusters mais avec le langage Python, ce qui fût un défi intéressant mais compliqué à réaliser.

Pour la table des couleurs CLUT a été choisi car recommander dans la consigne, le principe d'une CLUT (Color LookUp Table) a été choisi afin d'associer à chaque pixel de l'image compressée non plus sa couleur réelle mais un indice pointant vers la couleur la plus proche dans la CLUT. Cela permet une compression efficace.

Le codage RLE (Run Length Encoding) a été implémenté car il s'agit d'une méthode simple à mettre en oeuvre pour la compression sans perte des pixels. Elle repose sur l'encodage des suites de pixels identiques par un couple (pixel,nombre).

La gestion dynamique de la mémoire en allouant/libérant les tableaux en C est nécessaire pour traiter des images de tailles variables, sans connaitre la taille mémoire nécessaire au préalable.

3.2 Description fonctionnelle

Le programme offre une interface utilisateur permettant de charger une image, de régler le nombre de couleurs voulu, d'afficher la CLUT générée et de comparer visuellement l'image d'origine et compressée. Il comprend également des fonctions de codage/décodage RLE pour passer de l'image compressée à la forme binaire et inversement, aussi un K-Mean clustering, un essai de dithering pour effectuer un comparatif et des informations disponible avec Infos et une option de sauvegarde de l'image modifié.

3.3 Structure générale du programme

Le code a été découpé en plusieurs parties, en modules indépendants traitant de la lecture/écriture d'images, de l'algorithme k-means, de la CLUT, du codage RLE et de l'interface. Les données images sont stockées dans des tableaux 2D alloués dynamiquement. Les traitements s'enchaînent de manière séquentielle en fonction des entrées/sorties de chaque fonction.

1. Un header ima.h
2. main.c
3. La ppm.c
4. modif.c
5. la CLUT.c
6. le Makefile
7. Mes différents fichiers d'algorithmes.
 - (a) Kmeans.c
 - (b) Dither.c
 - (c) RGB-HLS.c

Chapitre 4

Implémentation

4.1 Algorithme de k-means

L'algorithme a été codé de manière itérative, avec les étapes suivantes à chaque tour :

Attribution des pixels aux classes en fonction de leur couleur Calcul des nouveaux centroïdes des classes Arrêt selon critère de convergence ou nombre max d'itérations

4.2 Création de la CLUT

La CLUT est stockée dans un tableau de structures contenant la couleur (3 composantes RGB) et l'indice associé dans la CLUT.

Les pixels sont alors remplacés par l'indice de leur couleur la plus proche dans la CLUT.

4.3 Codage/Décodage RLE

Le codage RLE parcourt l'image ligne par ligne et compresse les suites de pixels identiques.

La décompression reconstitue ligne par ligne à partir du flux binaire RLE.

4.4 Gestion mémoire

Les tableaux images et CLUT sont alloués/libérés dynamiquement au début/fin du programme.

L'utilisation de valgrind m'as permit de vérifier que je n'ai pas de fuite/leak de mémoire, pour ne pas avoir un code qui résulte avec une Segmentation Fault au moment où selectionnez dans le menu.

Listing 4.1 – Valgrind

```
==7159==  
==7159== HEAP SUMMARY:  
==7159== in use at exit: 0 bytes in 0 blocks  
==7159== total heap usage: n allocs , n frees , n bytes allocated  
==7159==  
==7159== All heap blocks were freed — no leaks are possible  
==7159==  
==7159== For counts of detected and suppressed errors , rerun with: -  
v  
==7159== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from  
0)
```

4.5 Réutilisation de fichiers

J'ai utilisé des fichiers déjà présent sur ce site https://jj.up8.site/AA/AA23_Images.html, pour avoir une base de fichiers avec les librairies exigés, et ce qui m'a aussi permit de comprendre qu'est-ce que CLUT et comment l'utiliser dans le projet.

Chapitre 5

Tests et validation

5.1 Méthodologie de test

J'ai effectué des tests sur différents fichiers ppm fournis sur le site https://jj.up8.site/AA/AA23_Images.html, 2 exemples de fichiers de taille différentes en rendus avec des résultats différents.

5.2 Résultats

Image Cordiliere.ppm :

Taille originale : 16056 octets Taille compressée : 2356 octets Réduction obtenue : 85.33

Image Agelmam.ppm :

Taille originale : 16056 octets Taille compressée : 2468 octets Réduction obtenue : 84.64

5.3 Images

Ci-dessous 2 exécutions bien différentes de K-means (Figures 5.2 et 5.3)

Ci-dessus une réduction de couleurs utilisant CLUT. (Figure 5.1)

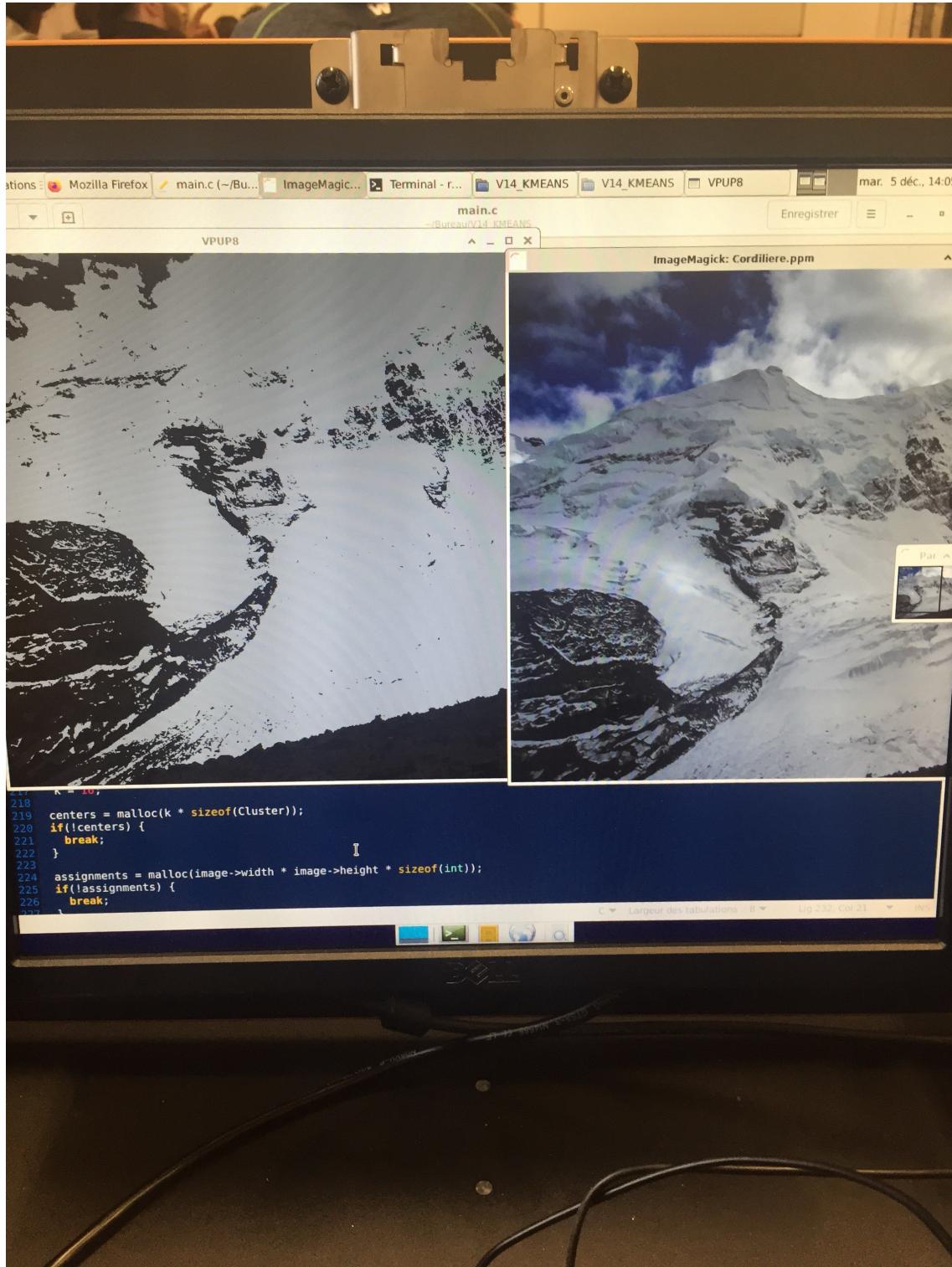


FIGURE 5.1 – Reduce Colors

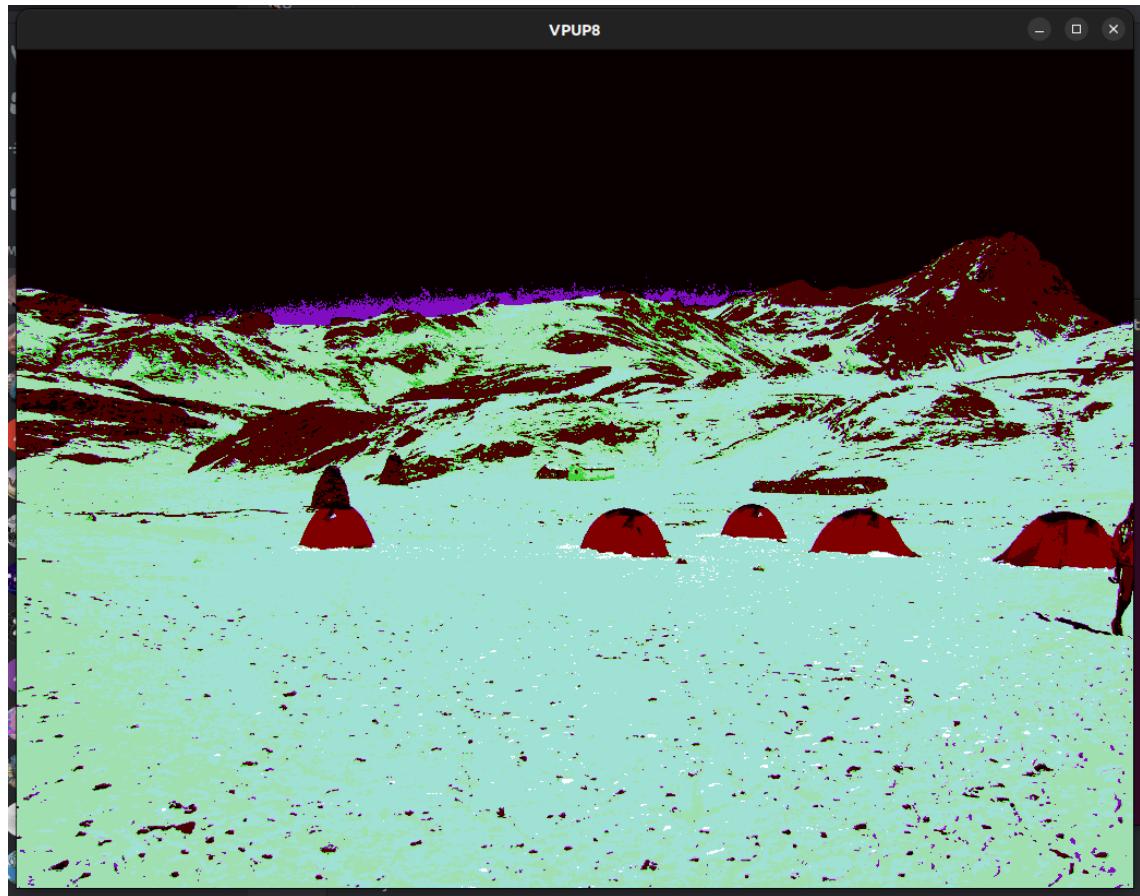


FIGURE 5.2 – Kmeans

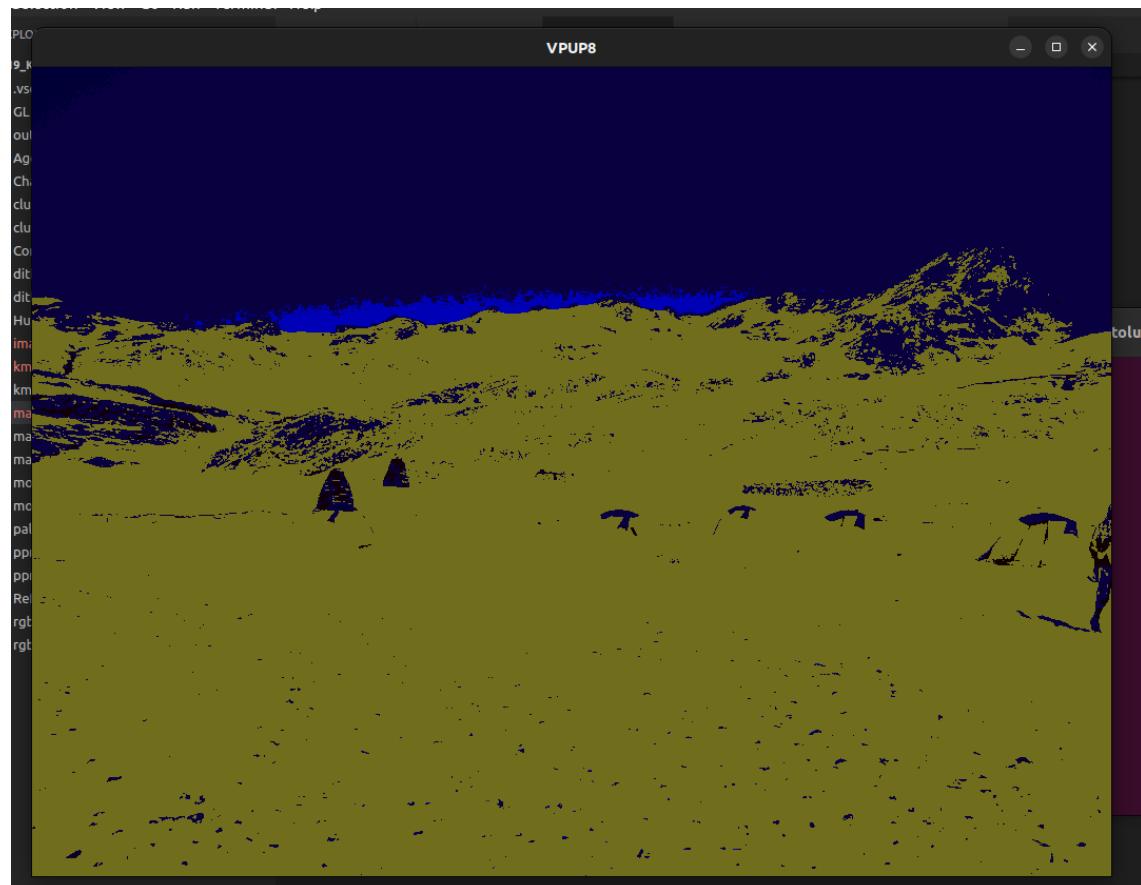


FIGURE 5.3 – Kmeans

Chapitre 6

Analyse et conclusion

Ce projet visait à appliquer diverses techniques de compression et de quantification d'images numériques afin de les analyser et comparer leurs performances.

Plusieurs algorithmes ont été implémentés :

Le codage RLE (Run Length Encoding) pour la compression sans perte des données. Cet algorithme simple mais efficace permet de réduire significativement la taille des images contenant de grandes plages de couleurs identiques.

Le clustering k-means pour la quantification des couleurs. En regroupant les pixels par couleurs similaires, cet algorithme permet de réduire le nombre de couleurs dans une image avec une très légère perte de qualité visuelle.

La construction d'une CLUT (Color LookUp Table) suivie d'une réduction du nombre de couleurs pour créer une palette personnalisée.

Un essai de dithering pour la diffusion d'erreurs lors de la quantification.

Ces techniques ont été implémentées de manière modulaire dans différents fichiers sources en utilisant des structures de données et des fonctions communes définies dans un fichier d'en-tête.

L'utilisation de structures globales partagées a nécessité l'ajout de déclarations extern et la définition unique des variables dans un fichier source pour éviter les erreurs de liens multiples.

Ce projet a permis d'appréhender des notions fondamentales de traitement numérique des images telles que la compression, la quantification et la gestion de projet avec langage C. Les différentes implémentations ouvrent des perspectives d'analyses plus poussées et de

comparaisons objectives des performances des algorithmes.

Ce projet pourrait être étendu et plus complet en implémentant d'autres algorithmes, en automatisant davantage les tests ou en l'interfaçant avec une bibliothèque de rendu 3D.

6.1 Bibliographie

Bibliographie

Liens intéressants qui ont pu m'aider :

https://jj.up8.site/AA/AA22_TP_Images.html

https://jj.up8.site/AA/AA23_Images.html

https://fr.wikipedia.org/wiki/Diffusion_d%27erreur

https://en.wikipedia.org/wiki/K-means_clustering