Année 2023-2024

Rapport de Projet:

Compression d'image

Projet réalisé par
Jonny MATHANARUBAN

Dans le cadre du cour
"Algorithme Avancée"

Sommaire

I-Objectif:

L'objectif dans ce projet est de limiter la quantité de couleurs d'une image, on tentera de compresser une image par réduction du nombre de couleurs utilisées.
Il fallait :

– écrire une fonction qui construit la Table d'Indirection de Couleurs (Color LUT),

– écrire une fonction qui compresse l'image et permet de l'enregistrer dans un fichier,

– écrire une fonction qui permet de lire un tel fichier,

– écrire une fonction qui permet de décompresser l'image et de l'afficher sur écran.

Il fallait également valider la méthode en termes de :

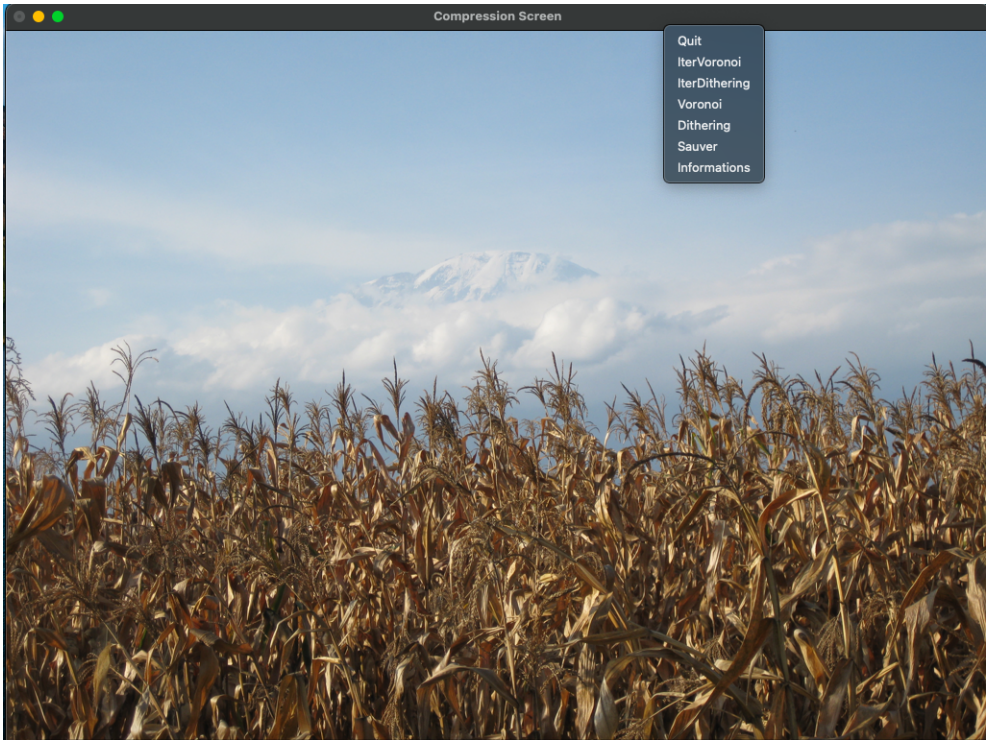
– temps de calcul, compression et décompression,
– perte en qualité éventuelle,
– ratio de compression.

Les algorithmes utilisés dans ce projet sont les algorithmes de Dithering et Voronoï pour la compression .
On va donc comparer essentiellement ces deux là en termes de temps et de performance .

Pour faire fonctionner le programme, il nous faut la librairie OpenGl(GL/glut.h)
que vous pouvez installer sur MacOs avec le gestionnaire de paquet Homebrew et sa commande brew install ou à l'aide de apt-get sur Linux.

II.Fonctionnement

Pour exécuter le programme il faut d'abord effectué un make dans le dossier du projet .
Exécuter le programme (./compression fichier.ppm)
Cela va nous ouvrir une fenêtre avec l'image que l'on cherche à réduire les couleurs.
Il nous faut ensuite cliquer dessus pour pouvoir afficher un bouton et choisir le type de compression que l'on veut (voir ci dessous un exemple d'execution).



On a le choix de plusieur types de compression :

iterVoronoi : Permet d'effectuer plusieur la compression d'image à l'aide de l'algorithme de Voronoï

iterDithering : Permet d'effectuer plusieur la compression d'image à l'aide de l'algorithme de Dithering

Voronoi : Effectue une compression d'image à l'aide de Voronoï

Dithering : Effectue une compression d'image à l'aide de Dithering

Mais aussi plusieur autres bouton :

Sauver : Permet de sauvegarder l'image après la compression ou même avant

Informations : Permet d'afficher la dimensions et le nombre de couleurs présents dans l'image sélectionnée (on peut l'utiliser pour aussi afficher le nombre de couleur après compression)

Quit: Ferme la fenêtre

III.Analyse du code

Ce projet contient plusieur fichier .c et un header .h qui contient mes structure ainsi que la définition de mes fonction , nous allons nous intéresser ici au fonction principale de ce projet .

Il nous faut d'abord créer une une table d'indirection des couleurs

```
void createCLUT(Image *im) {
   int i, dim, j;
   for (i = 0; i < CLUT_SIZE; i++) {
       clut[i].red = i * (255 / (CLUT_SIZE - 1));
       clut[i].green = i * (255 / (CLUT_SIZE - 1));
       clut[i].blue = i * (255 / (CLUT_SIZE - 1));
   }
   Pixel *pixels = (Pixel *) im->data;
   for (dim = 0; dim < im->sizeX * im->sizeY; dim++) {
       Pixel pixellu = pixels[dim];
       for ( j = 0; j < CLUT_SIZE; j++) {
           if (clut[j].red == pixellu.red &&
               clut[j].green == pixellu.green &&
               clut[j].blue == pixellu.blue ) {
               break;
           }
       }

   }
}
```

Cette fonction nous permet de créer de nouvelles couleurs , de les stocker dans le tableau clut et de voir si cette couleur existe dans l'image utilisée.

```c
void dithering_compression(Image * im){
   createCLUT(im);
   Pixel *pixel=(Pixel *)im->data;
   Pixel ancpixel,newpixel;
   int x, y, j;
   int qterrorRed, qterrorGreen, qterrorBlue ,colornumber = 0,indexdescouleurs;
   int *countcolors = (int *)malloc(CLUT_SIZE * sizeof(int));

   for (y = 0; y < im->sizeY-1 ; y++) {
       for (x = 1; x < im->sizeX-1 ;x++) {
           ancpixel = pixel[y * im->sizeX + x];
           newpixel = clut[couleurproche(ancpixel)];
           qterrorRed = ancpixel.red - newpixel.red;
           qterrorGreen = ancpixel.green - ancpixel.green;
           qterrorBlue = ancpixel.blue - newpixel.blue;

           pixel[y * im->sizeX + x +1].red += qterrorRed * 7 /16;
           pixel[y * im->sizeX + x +1].green += qterrorGreen * 7 /16;
           pixel[y * im->sizeX + x +1].blue += qterrorBlue * 7 /16;

           pixel[(y + 1) * im->sizeX + x - 1].red += qterrorRed * 3 / 16;
           pixel[(y + 1) * im->sizeX + x - 1].green += qterrorGreen * 3/16;
           pixel[(y + 1) * im->sizeX + x - 1].blue += qterrorBlue * 3 / 16;

           pixel[(y + 1) * im->sizeX + x].green += qterrorGreen * 5 / 16;
           pixel[(y + 1) * im->sizeX + x].red += qterrorRed * 5 / 16;
           pixel[(y + 1) * im->sizeX + x].blue += qterrorBlue * 5 / 16;

           pixel[(y + 1) * im->sizeX + x + 1].red += qterrorRed / 16;
           pixel[(y + 1) * im->sizeX + x + 1].green += qterrorGreen / 16;
           pixel[(y + 1) * im->sizeX + x + 1].blue += qterrorBlue / 16;

       }

   }
   memset(countcolors, 0, CLUT_SIZE * sizeof(int));

   for ( j = 0; j < im->sizeX * im->sizeY; j++) {
       indexdescouleurs = couleurproche(pixel[j]);
       if (countcolors[indexdescouleurs] == 0) {
           colornumber++;
       }
       countcolors[indexdescouleurs]++;
   }

   free(countcolors);
   printf("Nombre de couleurs après dithering : %d\n", colornumber);
}
```

La fonction dithering_compression est notre fonction principale pour la compression dithering , le but de l'algorithme de dithering est de propager la couleur de son pixel à pixel voisin (utilisation de la propagation de l'erreur).
On va d'abord voir sélectionner un pixel(ancpixel) et ensuite voir son voisin(newpixel).
On applique ensuite la différence entre ces deux pixel à ces voisin pour pouvoir la propager qui va donc de pouvoir comprimer l'image.

```c
void voronoi_compression(Image *im){
   createCLUT(im);
   int iter, nb_iter = 10, i, x, y,cpt;
   int red, blue, green;
   int *cell = malloc(im->sizeX * im ->sizeY * sizeof(int));
   for(iter = 0;iter < nb_iter; iter++ ){
       sites(im,cell);
       for (i = 0 ; i < CLUT_SIZE; i ++){
           red = 0;
           green = 0;
           blue = 0;
           cpt = 0;
           for (y =0; y < im->sizeY; y ++){
               for (x = 0; x <im->sizeX ; x++) {
                   if (cell[y * im->sizeX] == i){
                       red += im->data[(y * im->sizeX + x)*3];
                       green += im->data[(y * im->sizeX + x)*3+1];
                       blue += im->data[(y * im->sizeX + x)*3+2];
                       cpt++;
                   }
               }
           }
           if(cpt > 0){
               clut[i].red = red/cpt;
               clut[i].green = green/cpt;
               clut[i].blue = blue/cpt;
           }

       }
   }
   resites(im,cell);
   int val = countColorswithclut(im,clut);
   printf("Nombre de couleurs après Voronoï : %d\n",val);

   free(cell);

}
```

La fonction voronoi_compression effectue plusieurs itérations du processus Voronoï. À chaque itération, elle utilise la fonction sites pour attribuer à chaque pixel une étiquette correspondant à la couleur la plus proche. Ensuite, elle calcule la moyenne des couleurs associées à chaque étiquette et met à jour la table de recherche de couleurs. Ce processus itératif réduit progressivement le nombre de couleurs dans l'image, aboutissant finalement à une version comprimée de l'image originale.

IV. Observation

Nous pouvons observer que Voronoï compresse efficace l'image en réduisant la couleur au fur à mesure des itérations mais qui assez gourmand en terme de temps

```
voronoi Compression with iteration
Nombre de couleurs après Voronoï : 55
Nombre de couleurs après Voronoï : 24
Nombre de couleurs après Voronoï : 20
Nombre de couleurs après Voronoï : 17
Nombre de couleurs après Voronoï : 16
Temps d'éxecution IterVoronoi 41971120
```

Dithering est plus rapide que voronoï mais garde le nombre de couleur assez constant apres son utilisation.

```
dithering Compression with iteration
Nombre de couleurs après dithering : 59
Nombre de couleurs après dithering : 58
Nombre de couleurs après dithering : 59
Nombre de couleurs après dithering : 60
Nombre de couleurs après dithering : 61
Temps d'éxecution IterDithering :8256744
```

V.Conclusion

Voronoï rend une image très compressée , donc on peut difficilement observer l'image qu'il renvoie.
Dithering rend une image avec les couleurs qui sont altérées , on retrouve assez facilement l'image compressée .

Sources

Documentation:

Floyd–Steinberg dithering **documentation**.
Disponible sur: Wikipédia

Voronoï Diagramms **documentation**.
Disponible sur : Wikipédia