Rapport Projet Algorithmique avancée

Limiter le nombre de couleurs par une adaptation de l'algorithme de Voronoi

Aguesse Nathan
L3-B N°21001877

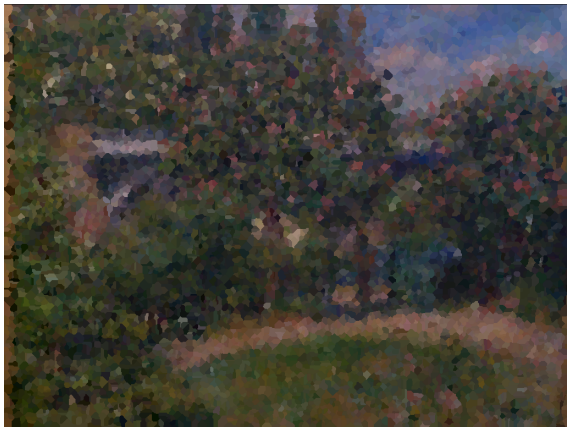
Aguesse Nathan
L3-B N°21001877

Contents

Aguesse Nathan
L3-B N°21001877

Image Original

Compression - Qualité basse / Ratio grand



Compression - Qualité moyenne / Ratio moyen

Compression - Qualité haute / Ratio petit

- Exemple de compression sur une image -

1 Fonctionnement du programme

1.1 Utilisation

Pour exécuter le programme, il suffit d'exécuter le Makefile, puis de lancé la commande **./voronoi nom-de-l'image nombre-de-sites**, sachant qu'il est possible de mettre "-1" (ou juste n'importe quel nombre négatif), pour que le programme utilise le nombre de sites maximum. (donc pour avoir la meilleur qualité de compression)

1.2 Présentation

Cet algorithme de compression d'image permet de compresser une image en réduisant le nombre de couleurs dans celle-ci, en s'inspirant de l'algorithme de Voronoi.

Le principe et le fonctionnement étant simple, il suffit de généré des sites de manière aléatoires sur l'image, qui prend en compte la couleur du pixel où ils se situent, et bien sur leurs positions.

Ensuite, on peut donc sauvegardé ces sites, donc dans notre fichier "compressé", pour après pouvoir lire ce fichier et récupéré les informations dessus.

Pour enfin pouvoir "décompressé" notre image, où l'on va prendre tous ces sites, et bien sûr construire le "Diagramme de Voronoi" avec.

2 Génération des sites

```
typedef struct Site {
    GLubyte r;
    GLubyte g;
    GLubyte b;
    unsigned short x;
    unsigned short y;
} Site;

typedef struct VoronoiCLUT {
    Site *sites;
    unsigned long sitesNmb;
    // taille de l'image
    unsigned short sizeX;
    unsigned short sizeY;
} VoronoiCLUT;
```

On va donc utilisé deux structures, une pour les sites en elles même, et une autre pour les

stocké, tout en prenant compte du nombre de sites que l'on a généré, et la taille de l'image.

Une fonction **'VoronoiCLUT generatevoronoi(int sitesNmb, Image *im)'** va donc nous permettre de généré tout cela, cependant il y a 2 choses à noter, la première étant que la fonction limite le nombre de sites maximum, aux nombre de pixels de l'image divisé par 10, car il n'y a pas vraiment d'avantage à aller au delà. La deuxième chose étant qu'il ai possible que des sites en doubles soit enregistré (à la même position), mais cela ne va pas posé de problème au reste du programme, mais une amélioration pourrait donc faire en sorte que cela ne soit pas le cas, malgré le fait que cela prendrait sûrement du temps supplémentaire.

3 Manipulation de fichier - Compression

3.1 Écriture

On va donc devoir écrire seulement notre "CLUT" dans un fichier, comprenant les informations essentielles à écrire, les dimensions de l'image et bien sûr tous les sites de Voronoi généré.

On écrira donc d'abord les dimensions sur les deux axes, chacun sur 2 octets (short) pour 4 octets au total, et ensuite chacun des sites, étant donc la couleur RGB, et sa position XY, la couleur va prendre 3 octets, et la position comme pour la dimension de l'image 4 octets.

On peut donc déjà déduire la taille du fichier compressé rien qu'en sachant le nombre de sites généré. (4 + (3+4) * NombreDeSites)octets

3.2 Lecture

Pour la lecture de notre fichier, cela sera le même fonctionnement, à un petit détails près, est que l'on connaît pas le nombres de sites dans un fichier, puisque l'on ne l'a pas noté dans

celui-ci, bien sur la raison étant que cela n'était pas essentielle, car un moyen de le connaître est de regardé la taille du fichier, puisque l'on sait que chaque site prend 7 octets en tout.

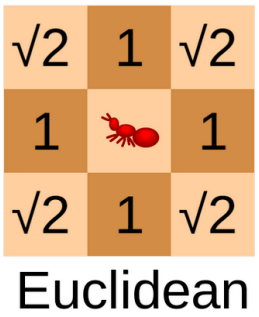
4 Reconstitution de l'image - Décompression

Ainsi ayant notre "CLUT" contenant nos sites de Voronoi sous la main, on va pouvoir appliqué nos sites sur une image, et les faire se répandre pour arrivé à un "Diagramme de Voronoi" à la fin.

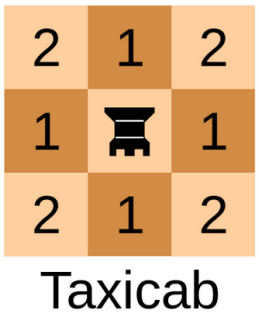
Pour cela, on va d'abord commencer par remplir l'image par un "fond vert", étant une couleur non-utilisé par les sites, où ils pourront donc se fier à cette couleur pour se propager.

Ensuite bien sûr, on va propager les couleurs de chaque sites petit à petit, jusqu'à que tous les sites n'ont plus de places pour s'agrandir, et ainsi notre programme aura terminé son fonctionnement.

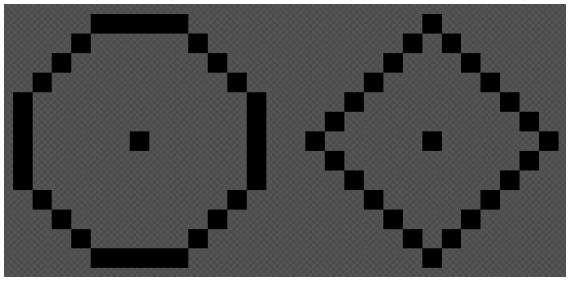
Cependant, il y a une chose assez importante à noter dans cette étape, est que l'on n'utilise pas vraiment le Voronoi avec une "mesure de distance habituelle", mais plutôt avec la distance de Manhattan, changeant de manière non-négligeable le rendu.



Euclidean



Taxicab



Ainsi, dessiner un "cercle" n'aura pas le même résultat par exemple, mais sera plus simple dans notre cas, que dans l'autre étant donné qu'il faudra seulement dessiner les 4 cotés du

"cercle".

Bien sûr, cela est donc un point étant possible d'améliorer, pour obtenir un "véritable diagramme de voronoi", mais demandant un code plus complexe pour dessiner les cercles, et peut être un peu plus coûteux en temps.

Mais malgré le fait que cela change le rendu, le plus la densité de sites est élevée, le moins on s'en apercevra.

Comparaison entre les deux systèmes de distances

5 Qualité de la compression

Pour finir, on peut maintenant regarder le ratio entre l'image originale et le fichier compressé, mais celui-ci varie tout simplement avec le nombre de sites généré, cela va donc être le rapport entre le nombre de pixels de l'image.ppm originale, comparé aux nombre de sites généré où l'on pourra obtenir le ratio de la compression, si l'on prend un exemple, le nombre maximum de sites que le programme autorise, étant le nombres de pixels de l'image divisé par 10, de plus la taille d'un pixel étant 3 octets, et la taille d'un site étant 7 octets on peut calculé le ratio en pourcentage avec **((10 * 7) / 3) = 23.3333**%, et donc, tout dépend du ratio du nombres de sites, par le nombres de pixels, si au lieu de 10 cela aurait était 100, on aurait donc perdu en qualité, mais la compression prendrait 10 fois moins de place, pour seulement 2.333% de la taille original.

Aussi à noter que toutes les fonctions sont en O(n) par rapport aux nombres de sites et de pixels de l'image, permettant une compression très rapide.