Faculté des sciences de Montpellier

Projet : Mosaïque d'images

Compte rendu semaine 7

Gonzalez Oropeza Gilles
Goussem Ayoub
Dupuis Thibaut

2025-04-06

Table des matières

1) Implémentation de la méthode par alignement

1.1) Qu'est ce que la méthode par alignement ?

1.1.1) Cas général

En bio-informatique, on peut calculer l'alignement par exemple de deux séquences d'ADN afin d'obtenir un score d'alignement pour les comparer.

L'algorithme implémenté est celui de Needleman-Wunsch. Dans sa version normale, il prend en paramètre 2 chaînes de caractères s et t à comparer de taille n et m, et 3 entiers pour le score de « match » (positif), de « mismatch » (négatif) et de « gap » (négatif) qui représentent la pénalité ou le bonus d'alignement ou de non alignement.

On va construire un tableau de taille $(n+1)(m+1)$ et précalculer le score d'alignement caractère par caractère dans chacune de ses cases. Les 1ères lignes et colonnes sont d'abord remplis par $i * \text{gap}$ et $j * \text{gap}$ respectivement. Ensuite, pour chaque case $D(i,j)$, sa valeur est le max des trois valeurs suivantes :

- Haut-gauche :
  - si $s[i-1] = t[i-1]$: $\text{Haut-gauche} = D(i-1, j-1) + \text{match}$
  - sinon : $\text{Haut-gauche} = D(i-1, j-1) + \text{mismatch}$
- $\text{Haut} = D(i, j-1) + \text{gap}$
- $\text{Gauche} = D(i-1, j) + \text{gap}$

Une fois le tableau construit, $D(n,m)$ indique le score d'alignement.

Dans l'algorithme de Needleman-Wunsch, on peut vouloir remonter le tableau afin de voir l'alignement des chaînes de caractères mais dans le cas de nos mosaïques, seul le score nous intéresse et nous venons de le calculer, on peut donc s'arrêter ici dans l'algorithme.

1.1.2) Application aux mosaïques

L'approche dans le cadre des mosaïques est similaire : on va calculer le score d'alignement entre deux images S (le bloc de l'image d'entrée) et T (l'image de la base de données redimensionnée à la taille du bloc) pixel par pixel. Lors de la génération d'une mosaïque, les tailles n et m sont égales et correspondent à la taille d'un bloc, ce qui nous permet de savoir à l'avance quelle taille allouer au tableau de score afin de l'allouer une fois en amont de tous les calculs et de le réutiliser.

Par rapport à la version générale, on va ici calculer le score d'alignement de chaque ligne et faire la somme des score des lignes pour avoir le score d'alignement du bloc avec l'image.

De plus, les valeurs de « match », « mismatch » et « gap » sont différentes. Le « match » et le « mismatch » sont calculés via une seule même formule : $-(|S_R(i) - T_R(i)| + |S_G(i) - T_G(i)| + |S_B(i) - T_B(i)|)$ qui représente la distance entre les pixels i et j des lignes de S et T respectivement. Le « gap » quand à lui vaut -30. Les 1eres lignes et colonnes sont toujours initialisées à $i * \text{gap}$ et $j * \text{gap}$ et pour $D(i,j)$, on a donc :

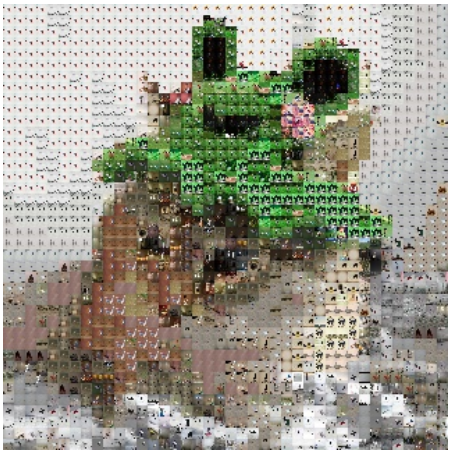
- $\text{Haut-gauche} = -(|S_R(i) - T_R(i)| + |S_G(i) - T_G(i)| + |S_B(i) - T_B(i)|)$
- $\text{Haut} = D(i, j-1) - 30$
- $\text{Gauche} = D(i-1, j) - 30$

et $D(i,j) = \max(\text{Haut-gauche}, \text{Haut}, \text{Gauche})$ et enfin $D(n,n)$ le score de ligne qu'on cherche.

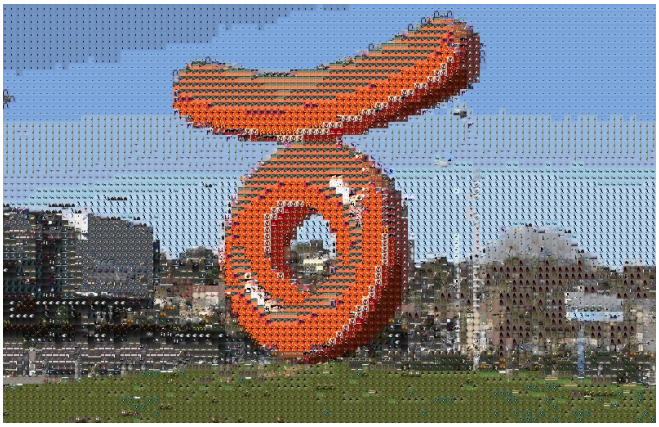
1.2) Résultats

Chat.jpg originale, Taille : 320x320

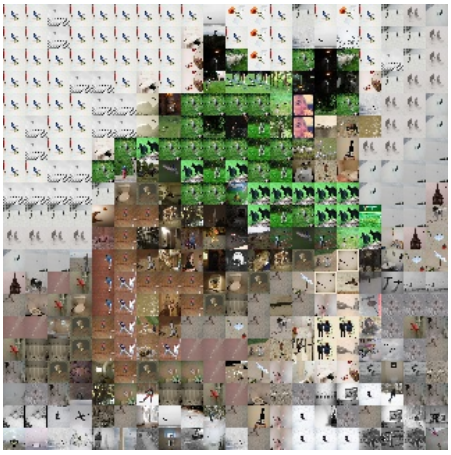
Rond-point.jpg originale, Taille : 800x512



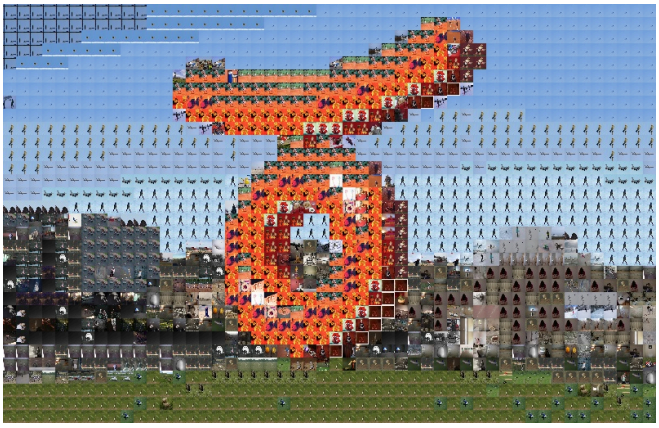
Chat.jpg avec taille de blocs : 8
PSNR = 14.94 dB, Temps : 477 secondes



Rond-point.jpg avec taille de blocs : 8
PSNR = 14.68 dB, Temps : 1913 secondes



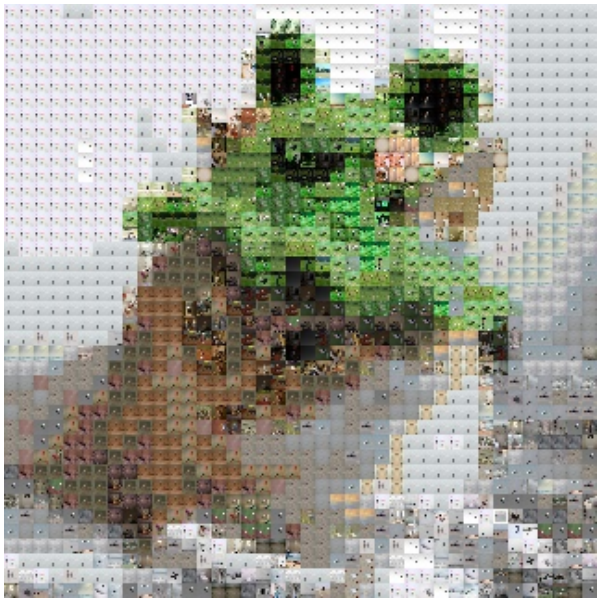
Chat.jpg avec taille de blocs : 16
PSNR = 14.04 dB, Temps : 949 secondes



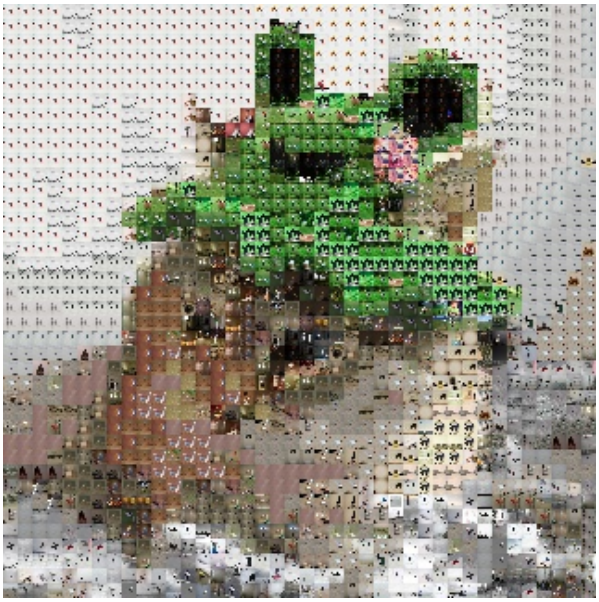
Rond-point.jpg avec taille de blocs : 16
PSNR = 14.52 dB, Temps : 3797 secondes

Fig. 1. – Exemples de mosaïques obtenues avec la méthode de l'alignement obtenues avec un CPU AMD Ryzen 5 5500H. On remarque que les temps de rendus sont très longs, et ce malgré un multithreading appliqué à la génération de la mosaïque et l'utilisation de seulement 10 000 images de la base de données sur les 120 000 disponibles.

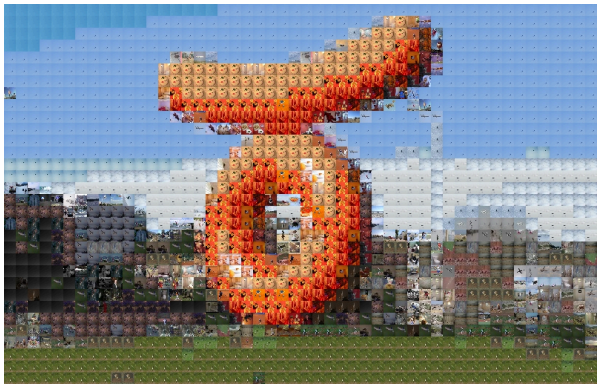
1.3) Comparaison avec la méthode par couleur et variance moyenne



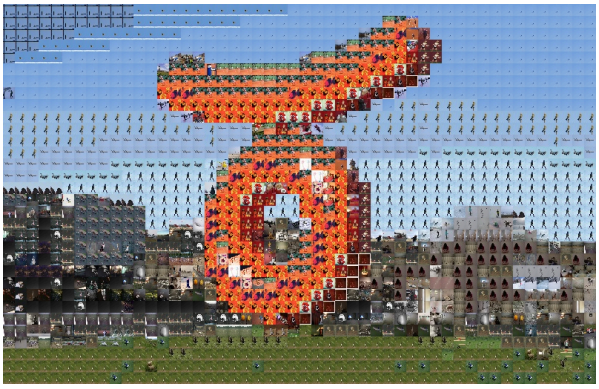
Chat.jpg avec taille de blocs : 8
Méthode couleur moyenne et variance
PSNR = 17.08 dB, Temps : 3 secondes



Chat.jpg avec taille de blocs : 8
Méthode alignement
PSNR = 14.94 dB, Temps : 477 secondes



Rond-point.jpg avec taille de blocs : 16
Méthode couleur moyenne et variance
PSNR = 16.98 dB, Temps : 3 secondes



Rond-point.jpg avec taille de blocs : 16
Méthode alignement
PSNR = 14.52 dB, Temps : 3797 secondes

Fig. 2. – Comparaison entre le rendu des mosaïques avec les deux méthodes pour des images et paramètres donnés. Le nombre d'images du dataset a été limité à 10 000 et la réutilisation des imagettes est autorisées pour les deux méthodes. Les rendus en couleur moyenne et variance ne sont pas multithreadés mais ceux en alignement le sont.

Pour la fin du projet, dans le cadre de la méthode par alignement, nous prévoyons de permettre d'avoir des imagettes uniques et enfin de créer des images avec les méthodes afin de les proposer dans un sondage pour déterminer subjectivement quelle est la préférée.

2) Création de l'interface

Pour notre projet, nous souhaitons implémenter une interface pour utiliser notre projet, dedans nous souhaiterons, dans le meilleur des cas, faire en sorte de choisir quel condition prendre pour notre image puis l'afficher et l'enregistrer. Nous voulons aussi faire en sorte de pouvoir calculer toutes les possibilités d'images en mosaïque et les montrer à l'utilisateur pour qu'ils puissent choisir celles qu'il souhaite prendre.
Pour ce faire, nous avons chercher differents moyens pour créer une interface, et n'étant pas natif dans C++, nous sommes allés chercher des bibliothèques existantes, comme par exemple Qt, qui est beaucoup utilisé mais nécessite une licence payante pour être utilisé donc nous avons choisi une autre option qui fonctionne aussi bien, FLTK qui est une bibliothèque open source permettant de faire des interfaces graphiques et elle est pratique car elle est légère et simple d'utilisation. https://www.fltk.org/index.php