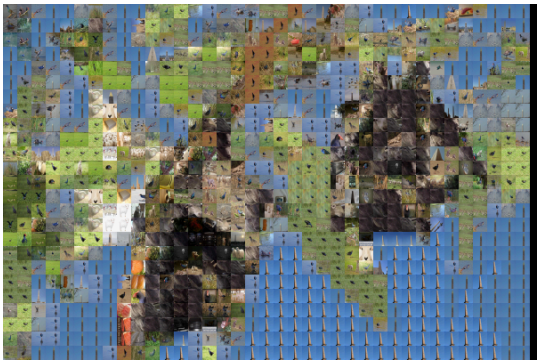
Projet photo-mosaïque -
Compte rendu semaine du 24/03

Viguier Killian Wang Xihao

M1 IMAGINE
Faculté des Sciences
Université de Montpellier

30 Mars 2025



Résumé

Cette semaine, nous avons effectué un sondage à propos de notre travail jusqu'ici. Nous avons également continué de réfléchir à la question de l'utilisation de Cuda pour l'amélioration des performances et nous nous sommes penchés sur une nouvelle méthode avancée sur laquelle travailler.

1 Réalisation du sondage

La plus grande partie de notre temps cette semaine à été accordée à la réalisation du sondage sur les photo-mosaïques grâce à nos méthodes. Le sondage est disponible via ce lien et les réponses seront recueillies pour les prochains rendus et pour la fin du projet.

2 Cuda du GPU contre Cache du CPU

Cette semaine, nous avions initialement prévu de comparer les temps d'exécution entre CPU et GPU sur les méthodes autre que celle sur moyenne, afin de vérifier si l'utilisation de CUDA permet d'accélérer notre programmes (parce que sur certains ordinateurs, les différences de performance de méthode moyenne ne sont pas significative). Cependant, nous avons rencontré des difficulté lors de l'implémentation de CUDA pour la méthode par histogramme. Nous n'avons pas finalement pu obtenir des résultats complets pour les comparaisons.

2.1 Informations obtenues

Cependant, nous avons quand même tiré quelques informations importantes pour cibler la partie du programme qui nécessite l'accélération.

Pour la méthode basée sur l'histogramme, il est nécessaire de commencer par redimensionner chaque image de la base de données à la taille du bloc(taille_bloc), puis de calculer la distance entre les blocs et imagettes et choisir l'une la plus correspondante.

D'après la mesure de temps effectuée, on a constaté que la réduction de la taille de chaque imagette est la plus coûteuse en temps.

```
resize() et compter histo des imagettes en : 76.5447s
comparer et distribuer en : 29.8579s
total en : 106.403s
```

Notre objectif actuel est d'accélérer la fonction resize() à l'aide de CUDA.

2.2 Accélération du cache du CPU

Nous savons que le CPU possède également son propre système d'accélération. Le cache permet d'accélérer le processus lorsqu'on charge la même base de données la deuxième fois, ce qui réduit le temps nécessaire) l'appel de la fonction resize().

```
premier fois (rien de données en mémoire)
    resize() et compter histo des imagettes en : 76.5447s
    comparer et distribuer en : 29.8579s
    total en : 106.403s
deuxième fois
    resize() et compter histo des imagettes en : 25.5032s
    comparer et distribuer en : 29.2003s
    total en : 54.7035s
```

Ce que nous avons faire ensuite, c'est plutôt comparer le temps de l'approche GPU avec celui de la deuxième lecture de base de données sur CPU.

3 Méthode avancée : détection des zones importantes

Un des inconvénients énoncés lors des précédents comptes rendus était le manque de couleur dans le bas des photo-mosaïques créées sans répétition d'imagette car celle-ci parcourt linéairement l'image d'origine.

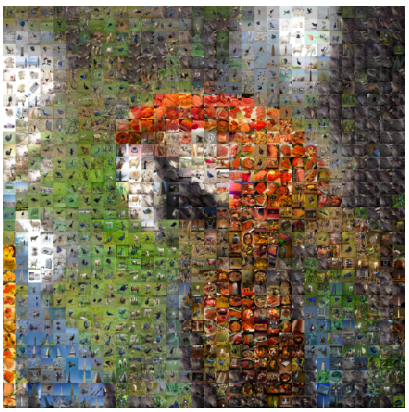


Fig. 1. – Photo-mosaïque de perroquet sans répétition d'imagette

3.1 Principe de la méthode

Pour palier ce problème, nous avons réfléchi à une nouvelle méthode pour tenter d'améliorer ce rendu sans répétition en utilisant une méthode vue lors du TP4 d'Analyse et Traitement d'images. Celle-ci consiste à prendre, à partir d'une image en couleurs ou en niveaux de gris, un seuil permettant de différencier l'objet du fond de l'image.

(a) Image originale (b) Image seuillée

Fig. 2. – Seuillage d'une image couleurs

Une fois que nous obtenons l'image seuillée, on peut désigner l'importance d'un bloc de manière binaire : Si le bloc contient des pixels noirs dans l'image seuillée correspondante, alors le bloc sera priorisé pour l'attribution des imagettes. De cette manière, les imagettes seront d'abord attribuées aux zones importantes de l'image.

3.2 Pour aller plus loin

On pourrait également modifier la condition pour juger de l'importance d'un bloc : au lieu qu'il ne doive contenir un seul pixel, on pourrait dire qu'il doit avoir au moins la moitié des pixels noirs. De plus, on pourrait établir une hiérarchie d'importance avec un seuillage en niveau de gris.