Compte rendu #5

Amélioration global du code

CANHOTO Mickaël, BELDJILALI Maxime
Master 1 Imagine, groupe #2.2

I - Introduction

La semaine dernière nous avions implémenter une interface basique afin de gérer les paramètres de nos mosaïques.

Cette semaine nous avons mis l'emphase sur la lecture du dataset CIFAR-10 ainsi que la résolution de problème que nous avions laissé de côté les semaines précédentes. Parmi ces problèmes nous avions la répétition d'images dans nos mosaïques qui était possible et le rapport de forme de l'image en sortie qui ne correspondait pas à celui de l'image en entrée.

Nous allons tout d'abord expliquer les fonctionnalités que nous avons ajouté à notre programme, pour ensuite discuter des problèmes que nous avons résolu durant la semaine.

II - Fonctionnalité ajouté

Nous allons vous présenter les fonctionnalités ajoutées dans notre programme.

1 - Lecture du dataset CIFAR-10

Nous avons rajouté la possibilité de lire le dataset CIFAR-10. Ce dataset est disponible sous la forme de 6 fichiers générés par la librairie **pickle** et est initialement utilisé comme dataset d'apprentissage en intelligence artificielle. Il est composé de 60 000 images d'une taille de 32x32 pixels, ce qui est idéal pour nos photos mosaïques. De plus, les images sont directement stockées dans des tableaux **numpy** qui sont des tableaux bien plus performants que les **list** python. L'utilisation de ce dataset nous évite une phase de redimensionnement très coûteuse en calcul et accélère grandement le prétraitement que nous faisions sur un dataset plus classique comme DIV2K.

À titre indicatif, sur une même machine nous, nous sommes passés d'une durée de prétraitement de 17,5s à 0,3s.

2 - Mosaïque sans répétition

Une des consignes de ce projet était d'assurer la non répétition d'images dans notre mosaïque. Dans nos implémentations précédentes nous n'avions pas encore respecté cette consigne.

Afin de respecter cette consigne nous mettons en place un système de *flag*, permettant de détecter si notre image est déjà utilisée dans la mosaïque. Si l'image est déjà utilisée nous cherchons l'image la plus proche suivante à l'aide de notre KDTree.

Cela diminue grandement la vitesse d'exécution et limite donc nos possibilités en terme de nombre de blocs. Pour faire face à cette nouvelle problématique, nous avons à notre disposition l'utilisation d'un algorithme glouton ou encore d'un algorithme de recherche évolutive comme décrit les semaines précédentes.

3 - Options avancées pour l'utilisateur

De plus, nous avons ajouté un bouton dans notre interface utilisateur. Ce bouton permet de dévoiler à l'utilisateur des paramètres plus avancés dans la création de mosaïque. Actuellement, en appuyant sur ce bouton, l'utilisateur pourra choisir le nombre de mosaïque en X et Y ainsi que leurs tailles.

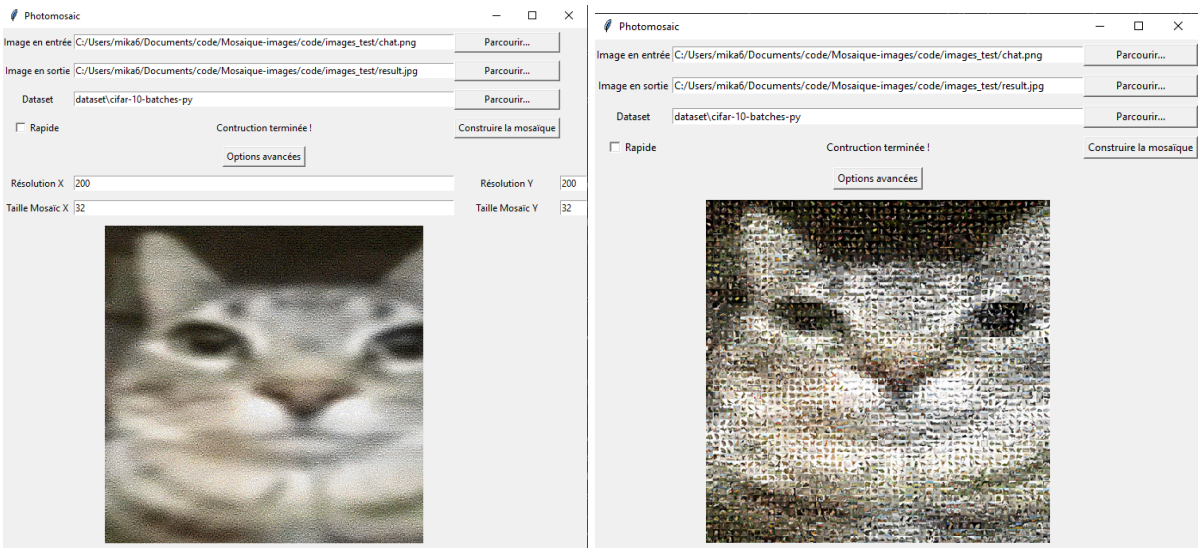

Figure 1 : Options avancées

Ce bouton étant optionnel, l'utilisateur n'est pas obligé d'appuyer dessus.

III - Problème réglé

Voici la liste des problèmes que nous avons réglés durant la semaine.

1 - Erreur de résolution

Un problème qui est survenu au début de notre implémentation, est que notre image finale devait être composée d'un nombre de sous image égal et X et en Y. Ce problème vient d'un mauvais indexage dans la sélection des tuiles et dans la taille de l'image finale. Cela nous permet d'obtenir des images de toutes résolutions et non pas que des imagés carrés.

2 - Gérer les images RGBA

Lors de nos essais avec plusieurs images en entrée, nous avons remarqué que certaines entraînent une erreur dans le programme. Après vérification, nous avons vu que cela provenait d'images en format PNG. Nous en avons conclu que notre programme ne faisait pas la vérification si l'image était encodée sur 4 canaux (RGBA). Nous avons alors ajouté la vérification entraînent une meilleure stabilité.

3 - Calcul de résolution

Afin de disposer d’un calcul de résolution correcte, nous donnons deux possibilités. La première étant la possibilité de forcer une résolution en sortie. La seconde étant de fixer un axe et de calculer automatiquement la résolution sur l’autre axe.

4 - Sécurité

Finalement, nous avons ajouté des mesures de sécurité afin d’éviter que le programme ne soit interrompu. Par exemple, si l’utilisateur ne rentre pas ou partiellement les informations nécessaires (ex : image entrée/image sortie) le programme avertira l’utilisateur au lieu de fermer.

IV - Conclusion

Pour conclure, cette semaine nous à permit de solidifier notre programme. Cet étape est très importante dans le domaine de programmation car elle permet d’éviter des dettes techniques et donc à terme faciliter l’implémentation de fonctionnalités. Cependant, nous avons aussi remarqué durant notre travail qu’il existe d'autres problèmes à corriger pour rendre notre programme totalement fonctionnel. Le problème le plus important, étant le fait que notre image mosaïque ne contient pas l’intégralité de l’image originale.