

Rapport de stage LaBRI

Lisa Weisbecker

20 août 2023

Résumé

Ceci est un rapport succinct de mon travail et des résultats obtenus lors de mon stage au LaBRI du 05/06/2023 au 18/08/2023, sous la supervision de Michaël CLEMENT et de Rémi GIRAUD.

Table des matières

1	Introduction	1
2	Travail réalisé	1
2.1	Comparaison des algorithmes	1
2.2	Tests sur la recolorisation	3
2.3	Réentraînement de SSN	4
2.4	Affinage du dataset COCO	4
2.5	Visualisation de l'attention	6
3	Conclusion et suite	7

1 Introduction

Mon stage consistait principalement à étudier l'effet de l'utilisation de différents algorithmes de segmentation par Superpixels dans le cadre du travail de Hernan Carrillo, Michaël Clément et Aurélie Bugeau, du papier "Super-attention for exemplar-based image colorization".

Les algorithmes que j'ai pu explorer sont les suivants : SLIC (celui utilisé de base dans le papier), SSN (Superpixel Sampling Network), SH (Superpixel Hierarchy), ainsi que brièvement SCALP (Superpixel Linear Path), LSC (Linear Spectral Clustering) et SL (Superpixel Lattice).

2 Travail réalisé

2.1 Comparaison des algorithmes

Pour commencer, j'ai écrit un script Matlab (*comparaison.m*) permettant de comparer les performances des différents algorithmes de segmentations par Superpixels selon différentes métriques : ASA (Achievable Segmentation Accuracy), EV (Explained Variation) et GR (Global Regularity). ASA mesure la capacité de l'algorithme à suivre les contours de l'image (selon une vérité terrain), EV l'homogénéité colorimétrique de la décomposition par Superpixels et GR la régularité et la compacité des Superpixels.

J'ai obtenu les résultats suivants :

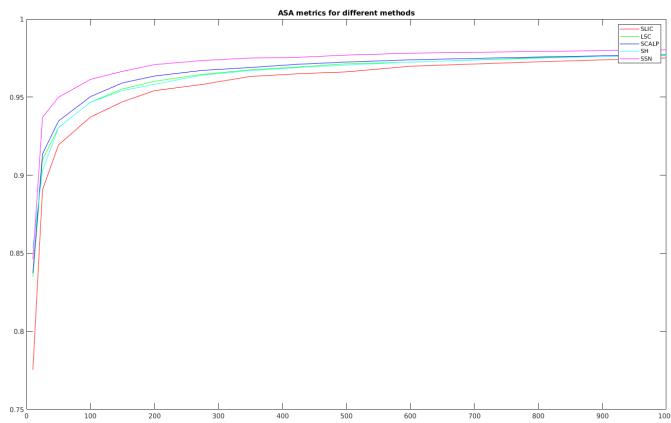


FIGURE 1 – Comparaison des performances sur ASA

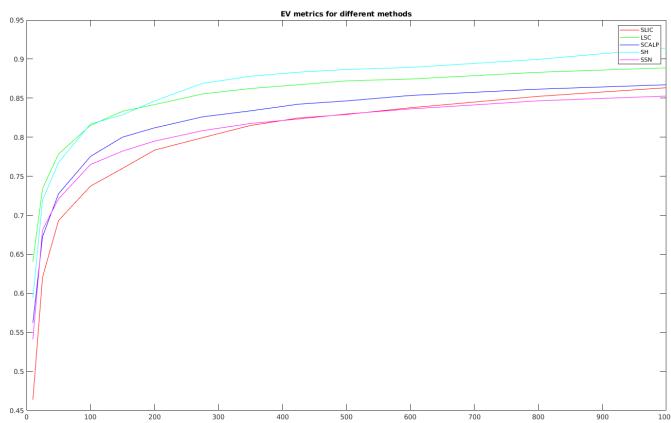


FIGURE 2 – Comparaison des performances sur EV

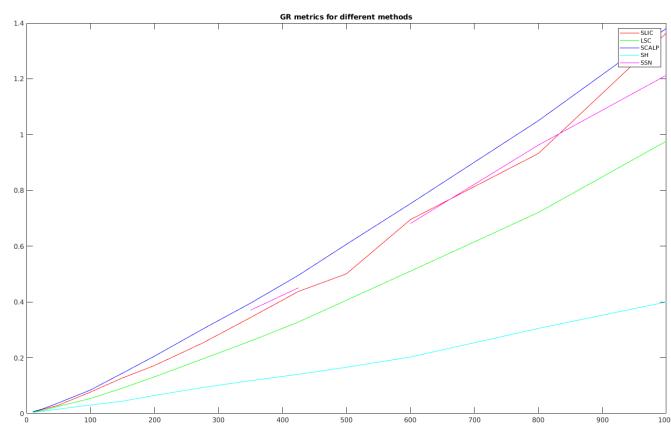


FIGURE 3 – Comparaison des performances sur GR

On peut observer qu'il est difficile de trouver un algorithme qui performe mieux que les autres

sur toutes les métriques, ce qui est assez logique : imposer une contrainte spatiale pour la régularité des superpixels réduit leur capacité à respecter correctement les contours. Il s'agit donc de trouver la métrique la plus "utile" au problème de recolorisation. Par la suite, j'ai donc utilisé certains de ces algorithmes dans la pipeline du code de Hernan Carrillo pour étudier leur effet.

2.2 Tests sur la recolorisation

Pour tester les différents algorithmes, j'ai changé l'implémentation de la segmentation par superpixels dans le code de H. Carrillo. J'ai décidé de réaliser mes tests avec SSN et SH, car ils semblaient être les plus intéressants au vu des résultats obtenus précédemment lors des comparaisons.

SH n'étant pas écrit en python mais en Matlab, il faut réaliser un pré-processing et calculer en avance les segmentations par superpixels. Ce serait aussi nécessaire pour tester d'autres algorithmes, à part SSN et SLIC qui sont en Python.

Pour futures utilisations, le script **hierarchy_preprocess.m** dans `\Superpixel_methods\SH` permet de réaliser ce préprocessing avec SH et **hierarchy_calcul.m** permet de visualiser les segmentations par superpixels pour différents nombres de Superpixels.

A noter qu'il faut au préalable calculer les contours des images avec un algorithme de détection de contours (cf [Pdollar edges](#)).

Une des particularités des superpixels hiérarchiques étant de très bien respecter les contours même avec un faible nombre de Superpixels, j'ai fait des tests avec différents nombres de Superpixels dans la segmentation. Pour SSN, j'ai testé différents paramètres de `color_scale` et `pos_scale` qui peuvent être changés lors des inférences et représentent le poids des features spatiales et colorimétriques de l'image.



(a) Image cible



(b) Image de référence



(c) Image recolorisée avec SLIC



(a) `pos_scale=2.0, color_scale=0.9`



(b) `pos_scale=5.0, color_scale=0.8`



(c) `pos_scale=2.5, color_scale=0.26`

FIGURE 5 – Résultats de la recolorisation avec SSN, avec différents paramètres



(a) 56 Superpixels



(b) 112 Superpixels



(c) 224 Superpixels

FIGURE 6 – Résultats de la recolorisation avec SH, avec différents nombres de superpixels

J'ai pu en conclure que l'algorithme de segmentation utilisé a en effet une influence sur la colorisation finale. Cependant, j'ai utilisé ici une image où les résultats sont assez flagrants, mais ce n'est pas le cas de la plupart.

Pour observer une claire influence sur le résultat final, il faudrait réentraîner le modèle en segmentant l'entièreté du dataset avec le-dit algorithme, car bien que j'utilise un nouvel algorithme lors des inférences, le modèle reste entraîné avec SLIC.

L'ensemble des résultats obtenus avec les différents algorithmes énoncés, et les segmentations par superpixels associées (à différents niveaux de résolution) sont stockés dans `\color-superattention\results`.

2.3 Réentraînement de SSN

Le modèle de SSN étant entraîné sur un dataset restreint de 500 images (BSDS500), il semble judicieux de le réentraîner sur un modèle plus large pour observer une différence éventuelle sur les performances.

Après entraînement sur le dataset COCO dans les mêmes conditions que le modèle entraîné sur BSDS500, j'ai obtenu les résultats suivants :

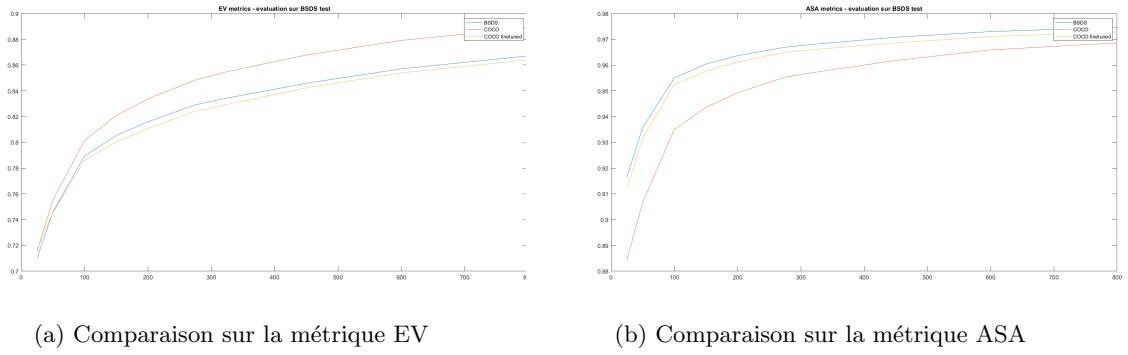


FIGURE 7 – Evaluation des performances des modèles de SSN sur BSDS500 test

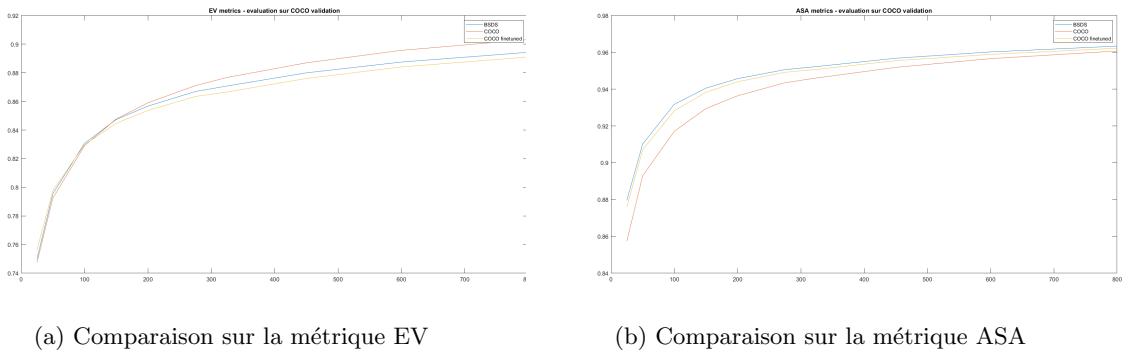


FIGURE 8 – Evaluation des performances des modèles de SSN sur COCO validation

J'ai comparé les performances entre le modèle entraîné sur BSDS500, celui sur COCO, ainsi que celui entraîné sur COCO et finetuné sur BSDS500 sur 50000 itérations.

Les résultats ne montrent pas d'améliorations significatives de performances avec un modèle entraîné sur le dataset COCO. Il se comporte cependant un peu mieux sur la métrique EV, ce qui traduit une meilleure homogénéité des couleurs.

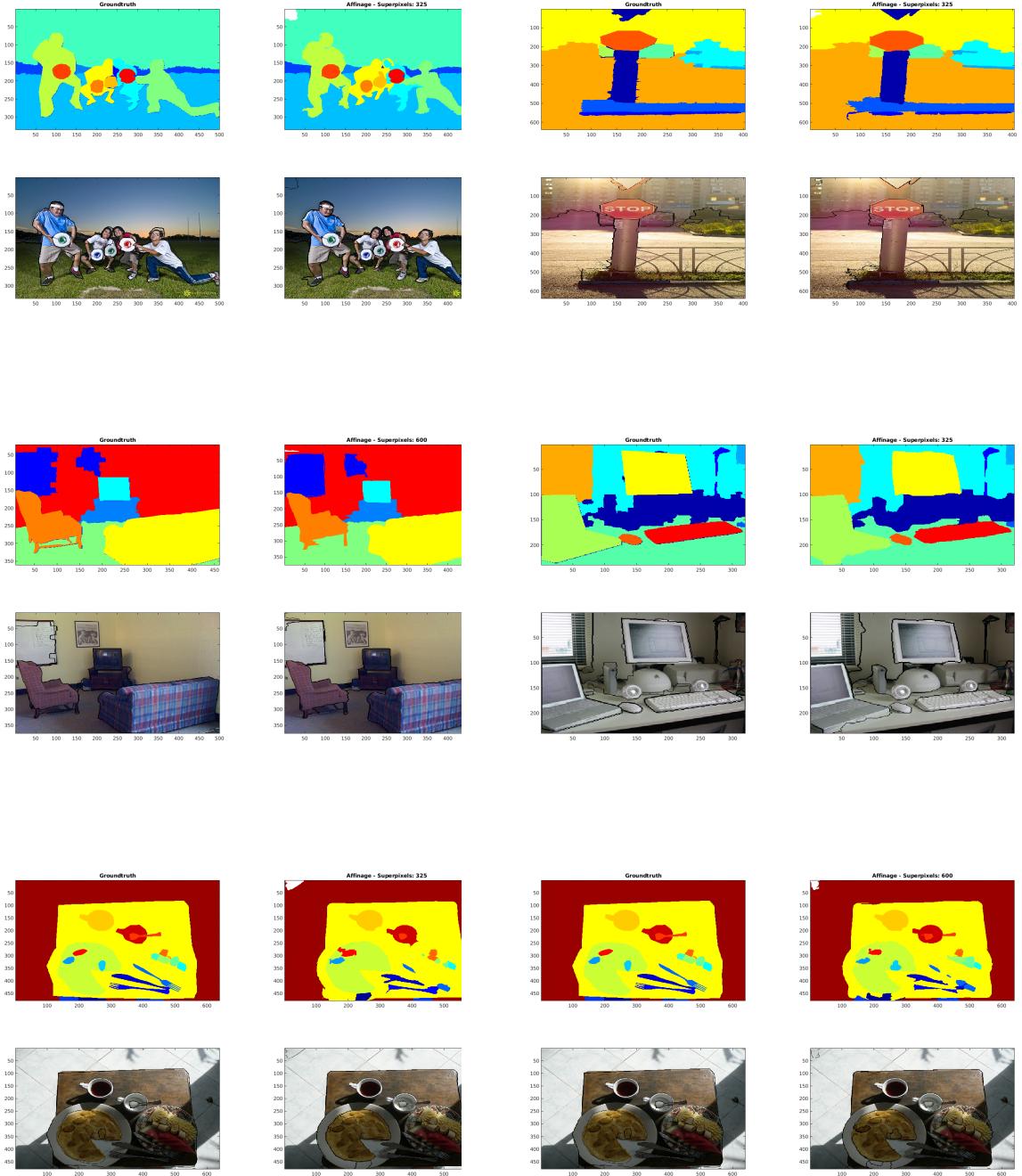
2.4 Affinage du dataset COCO

Une des raisons possibles pour la non-amélioration des performances avec un dataset plus large pourrait-être que COCO comporte des segmentations assez grossières. Pour corriger ce problème, nous

avons donc pensé à un script matlab permettant l'affinage d'une segmentation d'une image utilisant des superpixels. Les étapes sont les suivants :

1. Réaliser la segmentation par superpixels de l'image.
2. Pour chaque superpixel, associer le label de la classe la plus présente dans la groundtruth à l'intérieur de ce superpixel.

On obtient alors une nouvelle segmentation, sensée mieux respecter les contours de l'image. J'ai pu obtenir des résultats de ce type :



On peut observer plusieurs défauts à cette méthode :

1. Lorsque les couleurs de l'image ne sont pas assez contrastées, la segmentation est très approximative (bureau blanc et gris)

2. Lorsque la vérité terrain est très fine (fourchette, détails du fauteuil dans le salon), on perd beaucoup en précision.

Bien qu'on gagne effectivement en précision dans certains cas (tableau du salon, l'équipe d'ultimate, le panneau stop...), cette méthode ne semble pas donner de résultats assez consistants pour être utilisée sur un dataset entier, à moins de faire du cas par cas (ce qui nécessiterait beaucoup de travail humain).

Pour réutiliser le code : voir **affinage.m** pour utiliser SSN ou **affinage_slic.m** pour utiliser SLIC dans \Superpixel_methods.

2.5 Visualisation de l'attention

Au cours de mon stage, j'ai également réalisé une simple interface graphique en python permettant de visualiser l'effet de l'attention lors de la recolorisation avec les superpixels. En voici un aperçu :

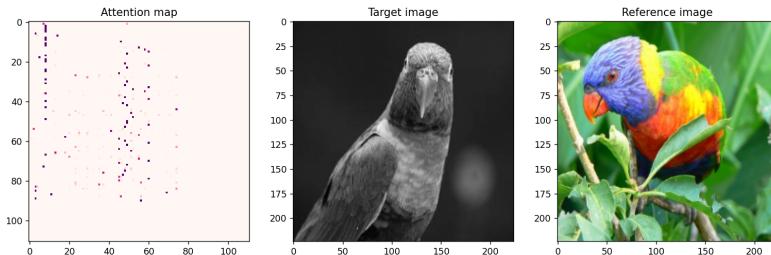


FIGURE 12 – Aperçu de l'interface

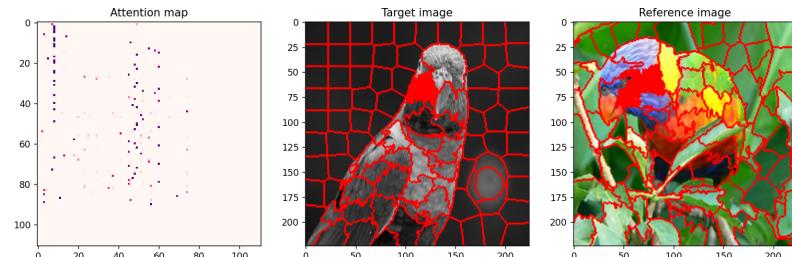
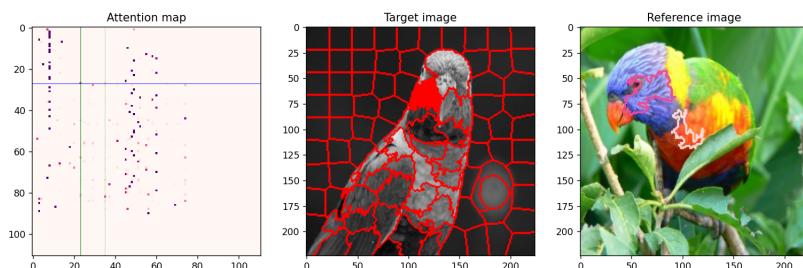


FIGURE 13 – En cliquant sur un pixel de la carte d'attention (à gauche), on peut mettre en valeur les superpixels correspondants sur la référence et la cible



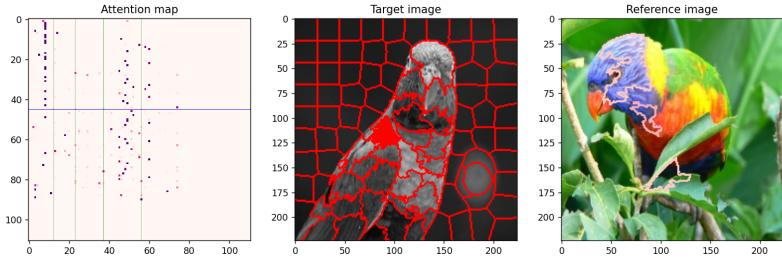


FIGURE 14 – En cliquant sur un superpixel de l'image cible, on peut mettre en valeur sur la carte d'attention et sur l'image de référence les Superpixels ayant le plus "influencé" sa colorisation (ceux ayant l'attention la plus élevée avec celui-ci)

Pour réutiliser le code : voir **attention.py** dans `\color-superattention`.

3 Conclusion et suite

Au cours de ce stage, j'ai pu considérablement améliorer mes connaissances en deep learning et en algorithmes de segmentation par superpixels.

Pour obtenir une conclusion plus claire sur mon sujet, il pourrait être intéressant (comme précisé précédemment) de réentraîner le modèle de recolorisation par super-attention en utilisant les différents algorithmes de segmentation par superpixels mentionnés ici. On pourrait également entraîner SSN avec de nouveaux datasets et explorer d'autres idées que je n'ai pas eu le temps d'implémenter : par exemple, exploiter la topologie lattice de superpixels générés par l'algorithme de superpixels lattices, ou la hiérarchie des superpixels hiérarchiques. Il faut alors modifier la structure du modèle de super-attention.