Cartoonisation d'image

Samuel Helye, Sylvain Leclerc

13 Novembre 2022

1 Recherche sur la décomposition sémantique

1.1 Théorie

Dans le but de développer notre méthode expliquée dans le premier compte rendu il nous faut se mettre à la page des méthodes sur la détection d'objets dans une image. Pour récapituler nous voulons pouvoir obtenir les objets présents dans une image (sous forme textuel) ainsi que leurs positions pour générer chacun d'entre eux séparément et ensuite les regrouper pour former notre résultat final. C'est ainsi que pour la première étape, nous sommes tombés sur une page web[1] qui présente les différentes méthodes de détection d'objets :

- **Histogram of Oriented Gradients** : la méthode la plus vieille pour la détection, selon le site sa première apparition était en 1986. L'objectif est d'extraire les caractéristiques importantes de l'image qui permettent de déterminer la nature de l'objet. C'est une bonne base de référence que nous allons sûrement implémenter.
- **R-CNN** : Le Region-based Convolutional Neural Networks est une amélioration de la méthode précédente qui couple des outils de sélections (Selective search) et un CNN (pour la classification). Une des principales limitations réside dans la rapidité d'exécution, de ce fait une version Faster R-CNN a été développé. L'objectif principal de cette version est de remplacer la phase de sélection par un réseau entièrement connecté (donc l'image est passée en entière dans le modèle, contrairement à la méthode initiale).

Les autres méthodes sont plus spécifiques et nous nous y attarderons suivant les résultats obtenus avec les deux premiers listés précédemment, il reste important de noter que ce n'est que la première étape de notre méthode, donc dans un premier nous recherchons des résultats cohérents avec un taux de précision exploitable pour préparer la suite. Ainsi la phase de d'optimisation prendra place plus tard si le temps nous le permet.

1.2 Pratique

Dans la pratique, il ne nous sera pas nécessaire de tout recoder, ni entraîner nous même (bien heureusement). Dans la même page web[1] différentes librairies sont listées pour utiliser les différentes méthodes présentées. La plupart de ces solutions sont conçues en Python (avec de l'utilisation de différents backend, comme Cuda notamment pour accélérer les calculs). Nous avons pour l'instant implémenter nos méthodes classiques en C++, nous nous dirigerons donc sûrement vers un une "caisse à outils" plutôt qu'une application bureau pour proposer différentes commandes avec des paramètres pour cartooniser les images.

2 Génération d'image

Dans les derniers mois, la génération d'image par IA a fait couler beaucoup d'encre. La raison de tout ça est l'apparition de deux IA très performantes dans le courant 2022, à commencer par Midjourney en Juillet 2022 suivi par Dall-E 2 en septembre 2022. Sans rentrer dans les détails, ces IA ont soulevé beaucoup de questions dans les domaines artistiques (notamment lié aux droits d'auteur des résultats, la légitimité des résultats, les droits des images utilisé pour l'entrainement, etc.), et tout cela principalement car les résultats proposés sont impressionnants.

FIGURE 1 – Couverture d'un magasine d'origine américaine qui est une image générée par Dall-E 2 (puis légèrement retouchée)[3]

De plus (ainsi qu'heureusement pour nous) OpenAI, une entreprise sur la recherche en intelligence artificielle, offre une API pour utiliser Dall-E2 pour générer des images avec une documentation riche. Le domaine de la génération d'image étant un sujet complexe, nous allons utiliser ces outils pour nous concentrer sur la mise en place et l'étude de notre méthode pour pouvoir conclure sur le sujet initial dans les temps.

3 Implémentation d'une méthode de segmentation

Cette semaine, nous avons implémenté un algorithme appelé "ligne de partage des eaux" cette méthode utilise la norme du gradient de l'image. Dans cet algorithme, l'image en niveau de gris est considéré comme une carte de hauteurs, que l'on va inonder, les différents bassins créés sont seront les différentes zones de l'image.

3.1 Fonctionnement de l'algorithme

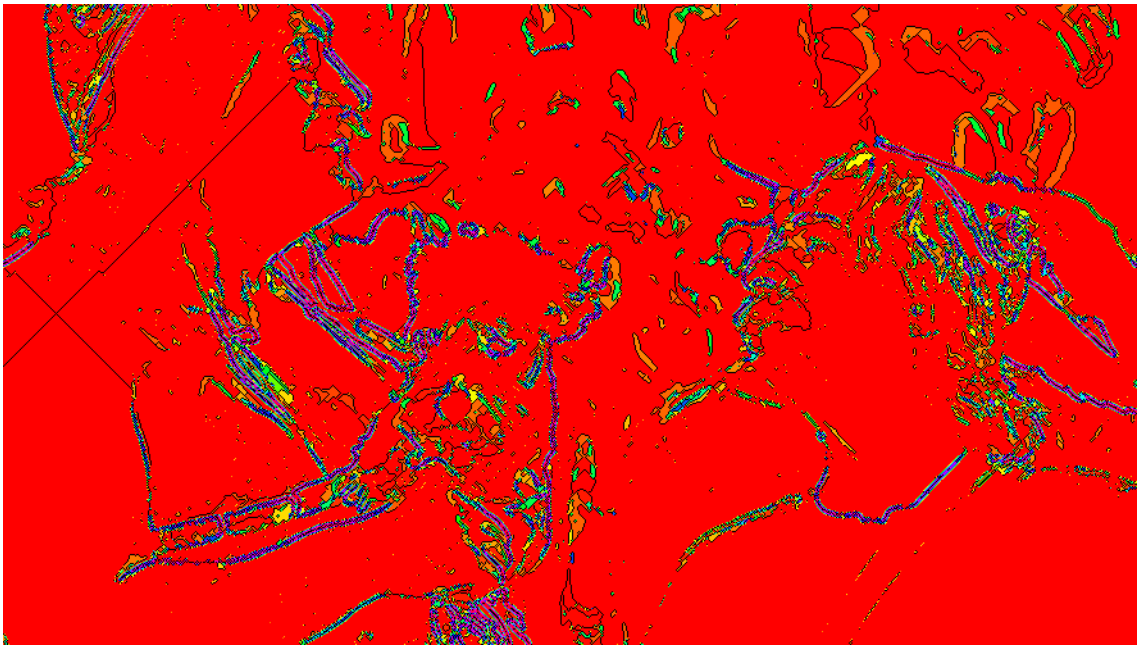
Dans un premier temps, on trie les pixels de l'image par intensité du gradient.
On définit un nombre d'étages, c'est-à-dire le nombre de niveaux d'altitudes différent qui seront considérés.
On crée une structure de donnée dans laquelle on va stoker l'état des pixels, soit non assigné, soit LPE (ligne de partage des eaux, c'est-à-dire la limite entre deux zones), soit le numéro du bassin auquel il appartient.

Pour chaque étage, on commence par regarder les pixels dont au moins l'un des voisins appartient à un bassin, on stocke ces pixels dans une file.

Dans le cas ou aucun pixels n'a de voisins appartenant à un bassin, on crée un nouveau bassin sur l'un des pixels de l'étage, et on ajoute ses voisins (qui sont sur le même étage) à la file d'attente.

Puis tant que la file d'attente n'es pas vide, on traite les pixels un par un. Pour ce faire, on regarde ses voisins. Si tous les voisins traités appartiennent au même bassin ou à une LPE, le pixel prend la valeur du bassin. Si au moins deux des voisins appartiennent à des bassins différents, alors le pixel devient une LPE.

On répète les deux dernières étapes jusqu'à ce que tout l'étage soit traité, puis on passe à l'étage suivant.
Voici un exemple d'image, segmentée par cet algorithme.



On affiche chaque zone avec une valeur de teinte différente dans l'espace HSV, état donné le nombre de zones, différentes zones peuvent se retrouver avec des valeurs identiques une fois la teinte échantillonnée sur 8 bits. Mais nous conservons bien la vraie zone.

3.2 Problèmes rencontrés et prévisions

Le principal problème de cet algorithme, en tout cas de notre implémentation, est son temps d'exécution. Nous n'avons pas encore de solution pour ce problème, si ce n'est de changer d'algorithme.

À court terme, l'objectif va être de corriger quelques problèmes avec notre implémentation. Puis dans un deuxième temps t'extraire des caractéristiques de chaque zone, pour pouvoir générer des textures à partir de ces caractéristiques et enfin remplir les zones avec ces différentes textures.

Références

[1] Bharath K. Object Detection Algorithms and Libraries (July 2022).

[2] OpenAI. Language Models are Few-Shot Learners (July 2020)

[3] Cosmopolitan. The World's Smartest Artificial Intelligence Just Made Its First Magazine Cover (June 2022)

[4] LPE. Segmentation d'image par détection de contours et algorithme "ligne de partage des eaux" université de Savoie (May 2018,)