Cartoonisation d'image

Samuel Helye, Sylvain Leclerc

13 Novembre 2022

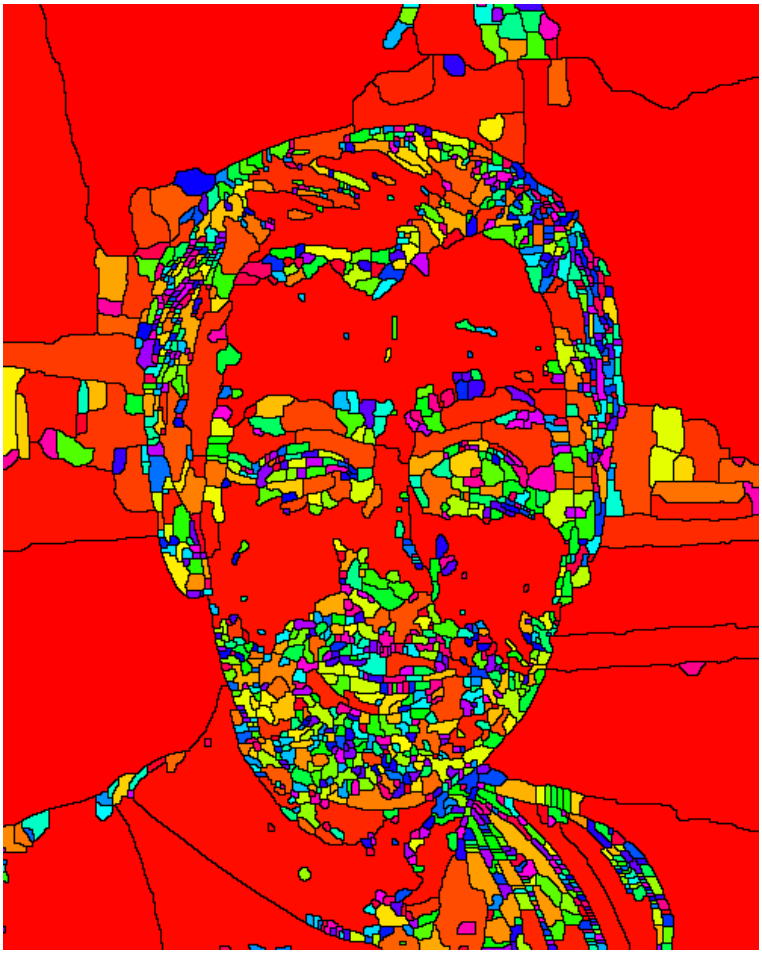
1 Cartoonisation par méthodes classiques

1.1 Correction de bugs

La semaine dernière, nous avions implémenté une méthode de segmentation par LPE, voici le résultat que nous avions obtenu.

La semaine dernière, il restait quelques bugs, que nous avons à présent corrigés.

Voici le résultat obtenu :



1.2 Aplats de couleur

L'étape suivante est de remplir les différentes zones non plus avec des couleurs aléatoires, mais avec textures générées à partir d'informations extraites de l'image d'origine.
Comme première méthode de remplissage, on calcule la couleur moyenne de la zone (dans l'image d'origine) puis on utilise cette couleur pour remplir les zones.
Voici le résultat :

1.3 Couleur des contours

Avant de développer d'autres méthodes de remplissage, nous voulions trouver une solution pour rendre les contours plus jolis. Une première idée, comme la version par défaut, est de les laisser noirs. On peut également les colorer en blanc :

Pour avoir l'impression qu'il n'y a pas de contour, on peut également les colorer avec la moyenne des couleurs des zones voisines :

Le meilleur résultat, selon nous, est obtenu en utilisant la moyenne comme dans la méthode précédente, et en l'assombrissant, entre 30% et 40%

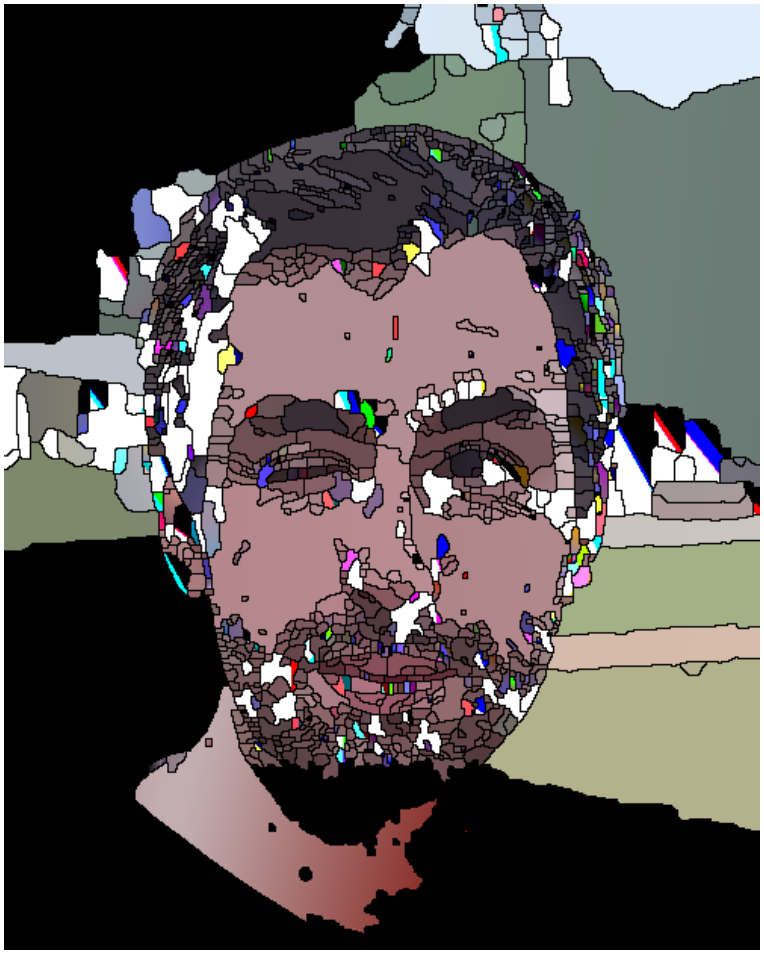
1.4 Génération de texture

Nous nous intéressons maintenant à la génération de texture, pour remplacer les aplats de couleurs.

La première approche que nous essayons est de générer un dégradé de couleur. Pour ce faire, il faut dans un premier temps trouver la direction principale du dégradé dans la zone d'origine. On calcule donc la moyenne de la direction du gradient. Puis, on place deux points sur l'image représentant le début et la fin du dégradé. Il faut ensuite calculer la couleur en ces deux points, pour le moment, nous avons décidé d'attribuer à un point, la moyenne des pixels étant plus proche de ce point que de l'autre.

Ensuite, au moment de reconstituer l'image, on interpole la couleur du pixel en fonction de sa position par rapport au deux points dont on connait la couleur.

Normalement cette méthode devrait fonctionner, mais il reste de toute évidence des bugs.



On peut cependant voir un dégradé qui semble bien formé au niveau du coup de Mathieu.

Pour la suite, il faudra corriger les problèmes de dégradés puis nous souhaiterions générer des textures plus complexes.

2 Génération d'images

Cette semaine pour la technique de génération d'images, il n'y a pas eu de résultats pour l'instant, mais nous avons pris le temps de tester et mettre en place l'environnement Python avec les différentes librairies. Pour la détection d'objets, nous avons utilisé le modèle YoloV3 (on va sûrement prendre le temps de mettre en place et essayer YoloV5 en entraînant nos propres objets) avec la libraire ImageAI. La librairie de Dall-E permet une intégration facile et nous allons pouvoir proposer des résultats dans le courant de la semaine à venir.

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,)