Segmentation et Vectorization

Travaux sur Machine Encadrés

Partie I - Réduction de la Palette des Couleurs

1 - Familiarisation

1. Compilez le programme : g++ -lglut -lGL -lGLU -I. *.cpp
2. Lancez le programme : ./a.out
3. Regardez main.cpp et image.h
4. Chargez une autre image.
5. Inversez l'image - une valeur de 1 est mappé sur 0 et vice versa. Affichez le résultat à l'écran.
6. Utilisez la fonction *mouse* pour récupérer les pixels ou l'utilisateur a cliqué et affichez les avec la fonction *drawPoints*.
7. Regardez la documentation de la classe vector
   (par exemple : http ://www.cplusplus.com/reference/stl/vector/)

2 - Segmentation

Écrivez l'algorithme de K-Means pour clusteriser des couleurs. (Consultez : Wikipedia K-means Clustering) Les centres utilisés seront les points donnés par l'utilisateur.

K-Means : l'idée est d'attribuer chaque pixel au centre le plus proche (en couleur et distance - controlez par une norme parametrée). Mettez à jour les centres en faisant une moyenne des pixels attribué, itérez le processus. Normalement ceci se fait jusqu'à la convergence mais fixez vous un nombre maximal d'itérations.

1. Que se passe-t-il quand la norme n'utilise que la distance en espace RGB ? Même question pour la distance spatiale.
2. Modifiez la fonction *keyboard* pour manipuler les paramètres de la norme et testez les.
3. Attribuez une couleur moyenne à chaque cluster pour réduire le nombre total des couleurs.

3 - Nettoyage

1. Implementez un filtre pour nettoyer l'attribution des clusters. En pratique, on va utiliser un filtre median (Consultez Wikipedia Median filter). *Filtre Median :* Autour d'un pixel, regardez son voisinage, et attribuez le median. Affichez le résultat.
2. Essayez d'appliquer ce filtre d'une manière itérative.
3. Peut-on faire mieux ? Essayez de supprimer les petites regions et attribuez le cluster dominant dans un fenêtre autour du pixel.

Partie II - Vectorization

Si vous n'avez pas réussit la segmentation, vous pouvez continuer avec le fichier MJBlackAndWhite.ppm.

1 - Le format SVG

1. Familiarisez-vous avec le format SVG et testez la classe SVGWriter. Il y a une fonction *testSVG* dans le fichier main.cpp.
2. Quel est l'effet de chaque commande dans cette fonction ?
3. Le but de cet exercice est de créer des résultats qui ressemblent à ce que vous trouvez dans le repertoire exemples.

2 - Extraction des regions

1. Trouvez un algorithme pour se ballader sur la bordure de chaque region de l'image.
2. Écrivez une version vectorielle de l'image qui n'utilise que des segments en utilisant le SVGWriter.

3 - Amélioration du résultat

Le but est de trouver un algorithme simple pour reduire la complexité de la version vectorielle.

1. Quels points de la représentation vectorielle faut-il garder au même endroit pour éviter des trous dans la représentation simplifié ?
2. Comment les détecter ?
3. Simplifiez les segments entre ces points avec un algorithme glouton : Essayez de fusioner deux segments, si tous les points restent dans une proximité epsilon du nouveau segment, gardez le, sinon continuez avec le prochain segment.
4. Question ouverte : Comment faire pour avoir une représentation en splines ?