Segmentation d'images de texture : expérimentation et validation

BOUDILI Younes

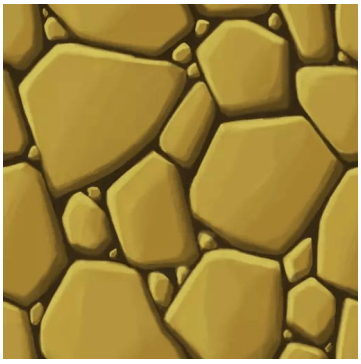
Département Sciences du Numérique - Filière Image et Multimédia
2020-2021

Table des matières

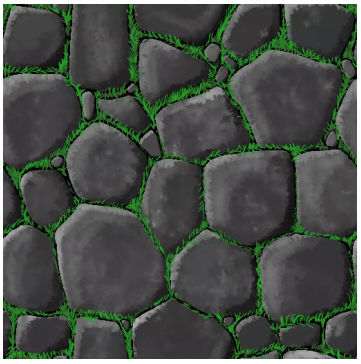
Table des figures

1 Introduction

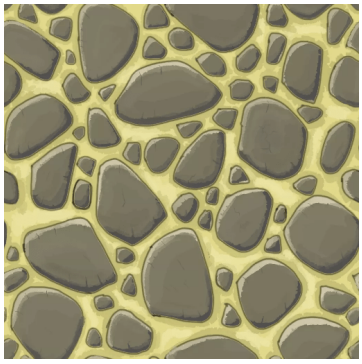
Le but de ce TP est de manipuler les outils d'`OpenCV` à travers `C++` pour traiter des images de textures et de les segmenter afin d'en extraire une segmentation de type fond/forme, ou une segmentation en N classes en supposant que chaque objet correspond à une unique classe. Nous allons travailler sur les trois images suivantes.



(a) Texture 3



(b) Texture 8



(c) Texture 11

FIGURE 1 – Les images à segmenter

2 Segmentation fond/forme (binaire) à partir de kmeans d'OpenCV

2.1 Utilisation

D'après la documentation d'OpenCV de la fonction kmeans, nous avons :

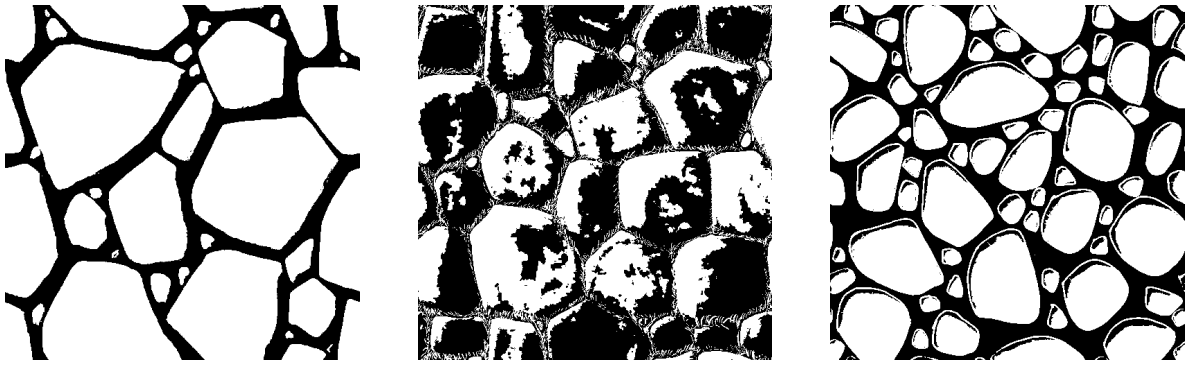
```
double cv::kmeans(
    InputArray  data,
    int         K,
    InputOutputArray    bestLabels,
    TermCriteria        criteria,
    int         attempts,
    int         flags,
    OutputArray         centers
)
```

Pour pouvoir utiliser `kmeans` d'OpenCV, il faut d'abord linéariser l'image à segmenter en un tableau de coordonnées flottantes. La fonction : `Mat linearizeImage(Mat image)` définie dans `kmeans.cpp` réalise cette tâche.

Ensuite, `kmeans` renvoie une matrice `bestLabels` qui, pour chaque pixel, contient le numéro de la région à laquelle il appartient. Nous obtenons aussi la matrice `centers` qui contient les informations des différents centres des régions.

A partir de ces résultats, nous pouvons construire l'image segmentée. Dans le cas d'une segmentation binaire ($k = 2$), nous choisissons arbitrairement entre le noir et le blanc, sinon les pixels prennent la couleur du centre de leurs régions.
Voir : `Mat buildSegmentedImage(Mat image, Mat centers, Mat bestLabels, int k)`.

2.2 Résultats



(a) Texture 3 (b) Texture 8 (c) Texture 11

FIGURE 2 – Résulats de la fonction kmeans d'OpenCV pour une segmentation fond/forme sur les textures (iterMax = 10000, seuil = 0.0001)

Nous remarquons que `kmeans` n'arrive pas à bien isoler les pierres dans des cas complexes. Il reste sensible aux ombres comme dans `Texture 8`, ou aux réflections/tons clairs comme dans `Texture 11`.

3 Segmentation fond/forme (binaire) à partir de votre kmeans

3.1 Algorithme

Pour implémenter l'algorithme de `kmeans`, j'ai suivi les étapes suivantes :

1. Découper l'image en N régions. Le critère choisi est un simple découpage suivant les niveaux de couleurs. Voir : `void initialize()`.
2. Estimer les centres C_k, $k \in [1; N]$, de ces régions. Cette estimation est faite pendant le découpage dans `initialize()`, par un calcul de moyenne sur chaque canal.
3. Tant que les centres ne sont pas modifiés (calcul d'une moyenne sur les écarts entre les anciens et nouveaux centres), et tant que nous n'avons pas atteint le nombre d'itérations maximal, faire :
   (a) Vider toutes les régions (réinitialisation des labels) ;
   (b) Pour chaque pixel de l'image, trouver le centre C_k le plus proche, et donner au pixel l'étiquette correspondante ;
   Voir : `void updateClusters()`.
   (c) Mettre à jour les centres C_k des N régions ainsi constituées.
   Voir : `void updateCenters()`.

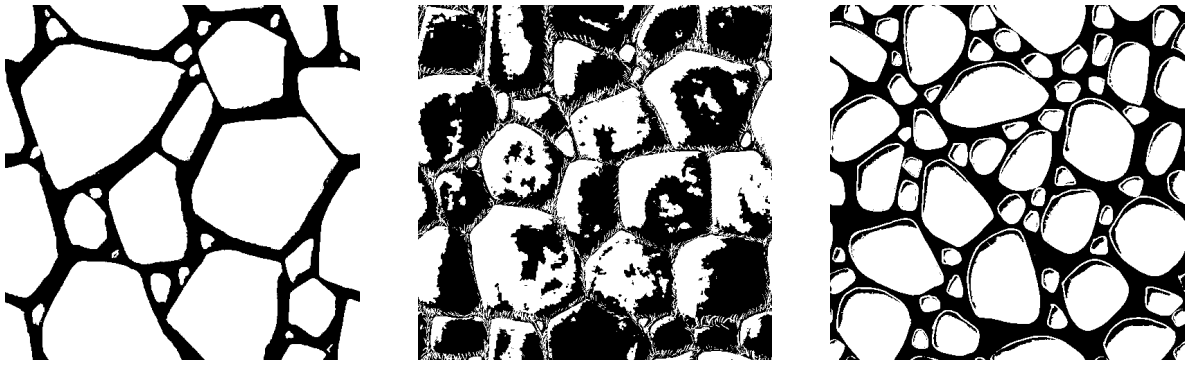
3.2 Distance

Dans l'algorithme précédent, pour calculer le centre le plus proche à un pixel, ou calculer l'écart entre deux centres, il faut avoir défini une distance sur les pixels. Pour cela, nous choisissons de la définir comme la norme euclidienne sur les valeurs RGB :

$$\begin{aligned} \|(r,g,b)\| : \mathbb{R}^3 &\to \mathbb{R} \\ (r,g,b) &\mapsto \sqrt{r^2+g^2+b^2}. \end{aligned}$$

Voir : `float dist(Vec3b p1, Vec3b p2)`.

3.3 Résultats



(a) Texture 3 (b) Texture 8 (c) Texture 11

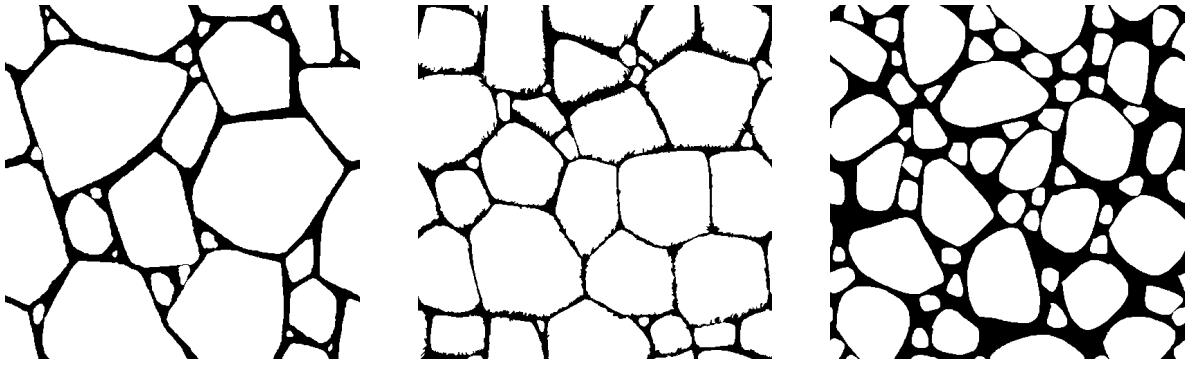
FIGURE 3 – Résulats de la fonction kmeans implentée pour une segmentation fond/forme sur les textures (iterMax = 10000, seuil = 0.0001)

Nous remarquons aussi que le `kmeans` implanté est sensible aux différences de couleurs dans un même objet. Les résultats restent proches visuellement de celui d'OpenCV.

4 Évaluation de la qualité d'une segmentation binaire

4.1 Principe

En fournissant une image de référence comme troisième argument, nous pouvons évaluer la qualité de la segmentation en calculant 3 critères (précision, sensibilité et similarité), qui combinent les pourcentages des vrais positifs, faux positifs, vrais négatifs et faux négatifs



(a) Texture 3 (b) Texture 8 (c) Texture 11

FIGURE 4 – Les images de référence

4.2 Résultats

En gardant les même paramètres précédents : iterMax = 10000, seuil = 0.0001, nous obtenons :

	OpenCV	Perso
Texture 3	Précision : 0.987463 Sensibilité : 0.885051 Similarité : 0.933456	Précision : 0.987463 Sensibilité : 0.885051 Similarité : 0.933456
Texture 8	Précision : 0.829637 Sensibilité : 0.556894 Similarité : 0.66644	Précision : 0.837837 Sensibilité : 0.551341 Similarité : 0.665046
Texture 11	Précision : 0.98874 Sensibilité : 0.922012 Similarité : 0.954211	Précision : 0.988729 Sensibilité : 0.919697 Similarité : 0.952965

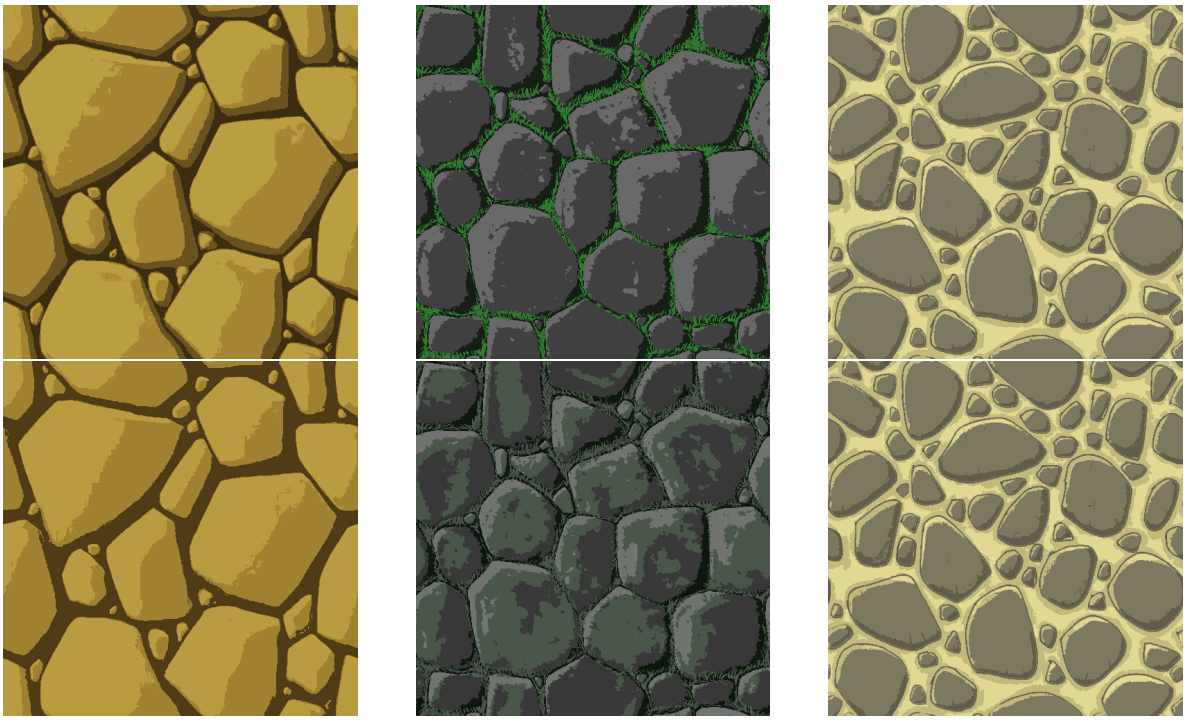
Nous allons nous concentrer sur le dernier critère qui combine les deux premiers. Nous remarquons que la différence entre les deux algorithme est très petite, mais celui d'OpenCV est plus précis et sensible.

5 Cas général

Dans le cas d'une segmentation binaire, les deux algorithmes ont un comportement très proche. Pour mieux tester la qualité, nous allons étendre le problème à une segmentation en 4 classes.

5.1 Résultats

La première ligne représente les résultats d'OpenCV, et la deuxième du kmeans que j'ai implémenté :



(a) Texture 3 (b) Texture 8 (c) Texture 11

FIGURE 5 – Segmentation des textures en 4 régions par les deux algorithmes

5.2 Évaluation de la qualité

A ce niveau, c'est difficile de quantifier les critères utilisés dans le cas de la segmentation fond/-forme. Nous allons alors s'appuyer sur les résultats visuels.

Nous remarquons que l'algorithme implémenté présente des limitations. Dans le cas de la texture 3, il y a un problème où deux régions ont des centres très proches, presque indiscernable à l'oeil, quand `kmeans` d'OpenCV sépare bien les ombres du fond. Ou bien dans le cas de la texture 8, notre algorithme combine l'herbe avec les pierres. Ces observations sont peut-être causé par la différence entre les initialisations des régions (aléatoire/découpage).

En conclusion, ces évaluations restent subjectives et ne permettent pas à bien déterminer la qualité de notre algorithme. Dans des cas d'étude différents, il se peut que `kmeans` implanté s'approche mieux du résultat souhaité.