Rapport TP
Segmentation d'images de texture : expérimentation et validation en utilisant kmeans

SADURNI Thomas

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

Table des matières

Table des figures

1 Introduction

Dans ce TP, nous avons manipulé trois images simples afin de les segmenter en N classes en supposant que chaque objet de l'image correspond à une unique classe. Ces différentes segmentations se font à l'aide de l'algorithme des *kmeans*. Cet algorithme associe chaque donnée de l'image à son centroïde le plus proche. Les centroïdes initiaux peuvent choisis de différentes façons. Après une itération de l'algorithme, chaque centroïde est replacé selon la moyenne des données déjà associées.

Dans un premier temps, nous avons utilisé la fonction *kmeans* de la librairie *OpenCV*, puis j'ai implanté ma propre version de l'algorithme. Afin d'évaluer la qualité de la segmentation j'ai implanté un petit algorithme pour calculer les *True Positive* (TP), *True Negative* (TN), *False Negative* (FN) et *False Positive* (FP) pour obtenir la précision (P), la sensibilité (S) et le coefficient de similarité (DSC) par rapport à une image segmentée de référence. Ainsi, plus ces valeurs sont proches de 1, plus la segmentation est correctement réalisée.

2 Segmentation des images à disposition à partir des kmeans

J'ai d'abord mis mes images en niveau de gris pour faire un *kmeans* sur des en pixels en niveau de gris.

2.1 Images

Voici les résultats de segmentation des différentes images à disposition :

FIGURE 1 – Image 3 de base suivie des images segmentée (*openCV* puis *my_kmeans*) pour k=3

FIGURE 2 – Image 8 de base suivie des images segmentée (*openCV* puis *my_kmeans*) pour k=3

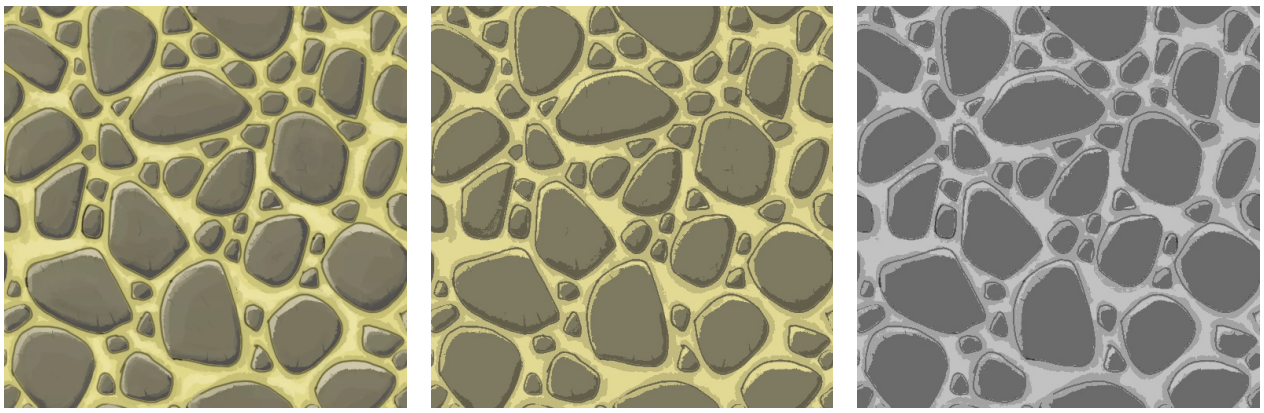


FIGURE 3 – Image 11 de base suivie des images segmentée (*openCV* puis *my_kmeans*) pour k=4

3 Evaluation de la qualité de segmentation

Comme énoncé dans l'introduction, pour évaluer la qualité de la segmentation, il faut caculer les les *True Positive* (TP), *True Negative* (TN), *False Negative* (FN) et *False Positive* (FP) à l'aide d'une image de référence dite de vérité terrain. Pour celà, j'ai impalnté la fonction *qualitySegmentation* prenant en paramètre l'image de référence, l'image semgentée avec *kmeans* de *OpenCV* et l'image segmentée avec *my_kmeans*.
Avant de commencer le calcul des performances, je change mes images pour les mettre en noir et blanc (k=2), j'utilise donc un seuil. Mon algorithme n'est pas optimal car pour chaque image, il faut modifier la valeur du *seuil*, ligne 91 et modifier la comparaison ligne 93 et 97 (changer $<$ en $>$ ou inversement).

```
90    // le seuil est à changer pour chaque image (pas très optimal)
91    int seuil = 100;
92    //conversion en noir et blanc de l'image segmentée avec my_kmeans
93    Mat my_image_nb=my_image_seg<seuil;
```

FIGURE 4 – Modification du seuil pour chaque image

Une amélioration possible aurait été de calculer les valeurs de I et $255 - I$ (avec I les valeurs de l'image segmentée) et conserver le meilleur résultat.

Les résulats suivants sont obtenus avec k=3, qui semble être la valeur la plus optimale.

3.1 Image texture3

Voici ci-dessous les images que nous avions à disposition pour l'image *texture*3.

Regardons maintenant les résultats de nos segmentations, à gauche la fonction de *openCV* et à droite *my_kmean*. Ici le *seuil* est fixé à 100 et j'inverse le blanc en noir et le noir en blanc.

On remarque que les deux segmentations sont quasi-similaires, seuls les coins des éléments blancs diffèrents faiblement. En revanche, par rapport à l'image de référence, les contours sont plus épais avec *kmeans*.

De plus, d'après les résultats de la qualité de segmentation, on remarque que la précision, qui permet de mettre en évidence la proportion de *True Positive* parmi les positifs est légèrement plus importante avec le *kmeans* de *OpenCV*. Dans les deux cas, on est assez loin de la valeur unitaire qui correspond à une segmentation parfaite, cette différence se trouve dans le fait que les contours sont plus épais et que la segmentation n'a pas fait un travail suffisant. En, effet sur l'image de base, on aperçoit une ombre qui devient de plus en plus foncé au niveau des contours de chaque pierre. L'algorithme à attribué à ces pixels une mauvaise classe mais il est difficile pour celui de savoir que ces ombres ne font pas partie de la séparations des pierres, mais des pierres elles-mêmes. C'est pour cela qu'il y a une erreur assez importante.

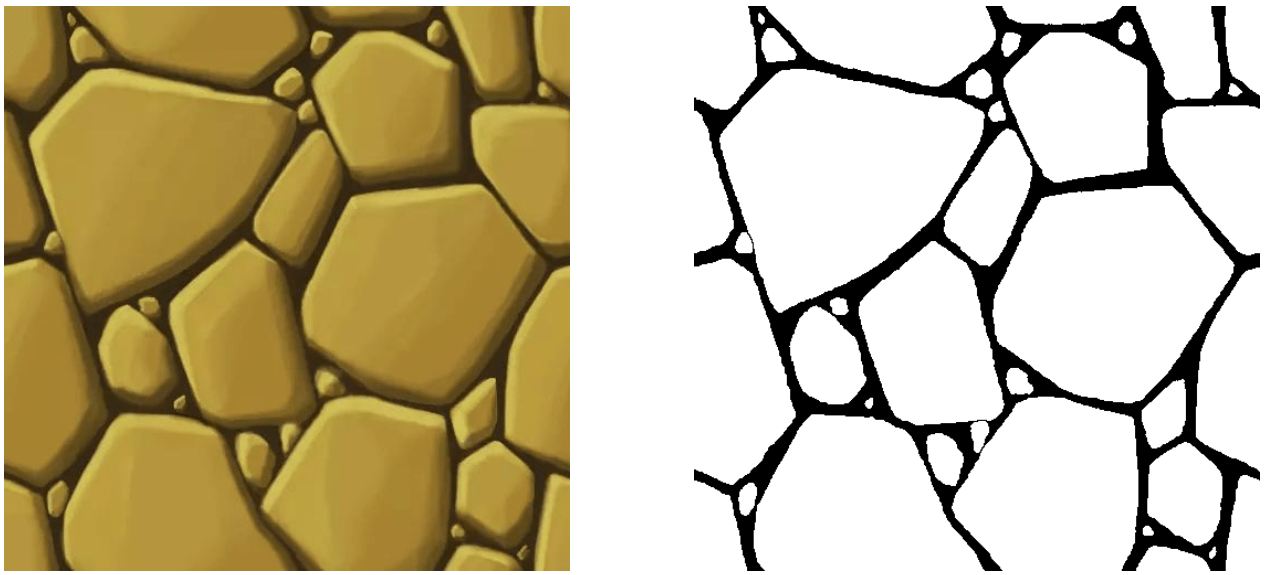


FIGURE 5 – Image *texture*3 de base et image segmentée de référence

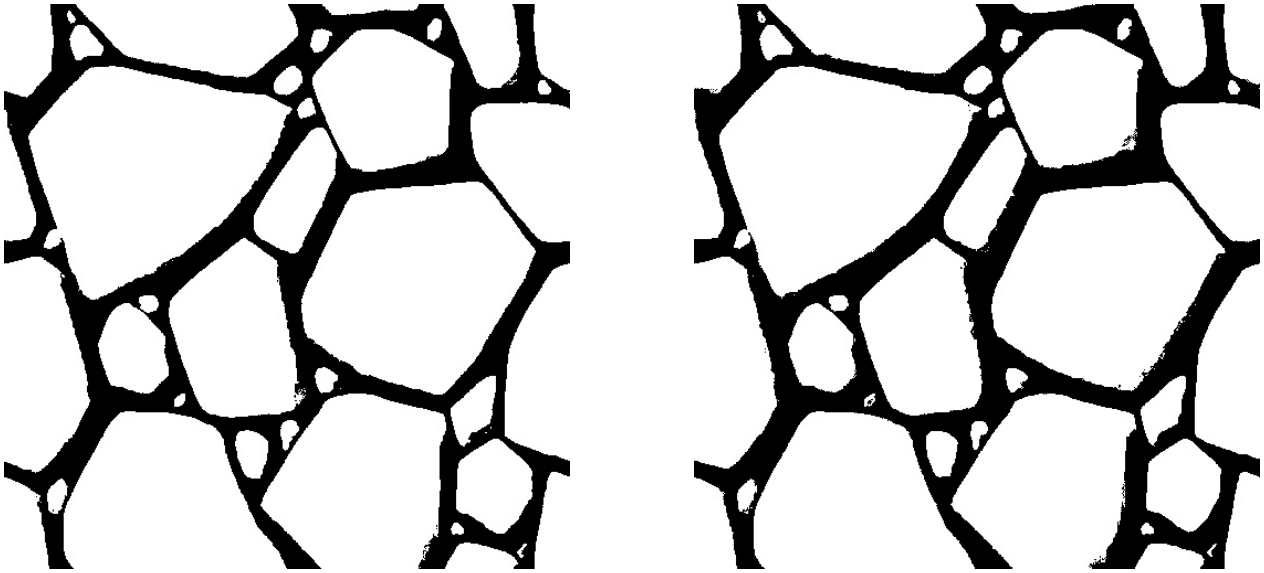


FIGURE 6 – Images *texture*3 segmentées *openCV* et *my_kmeans*

En ce qui concerne la sensibilité, qui met en évidence la proportion de bonnes détéctions par rapport à la forme à segmenter, on voit que *my_kmeans* est significativement plus efficace (un dixième) que *openCV* et le résultat semble satisfaisant.

Enfin, pour le coefficient de similarité ou $DICE$, qui est la moyenne harmonique entre la précision et la sensibilité, nous sommes autour de $0,81$ et $0,83$ respectivement pour *openCV* et *my_kmeans*, ce qui est un résultats correcte mais pouvant être amélioré. Je rappelle que pour une segmentation parfaite, il faut parfois faire appel à plusieurs algorithmes donc un ordre de $0,82$ est, il me semble, satisfaisant.

```
RESULTATS :

Precision openCV: 0.773098
Precision my_kmeans: 0.7399

Sensibilité openCV: 0.855086
Sensibilité my_kmeans: 0.951199

DICE openCV: 0.812028
DICE my_kmeans: 0.832349
```

FIGURE 7 – Evaluation de la qualité de segmentation de l'image 3

3.2 Image texture8

Voici ci-dessous les images que nous avions à disposition pour l'image *texture*8.

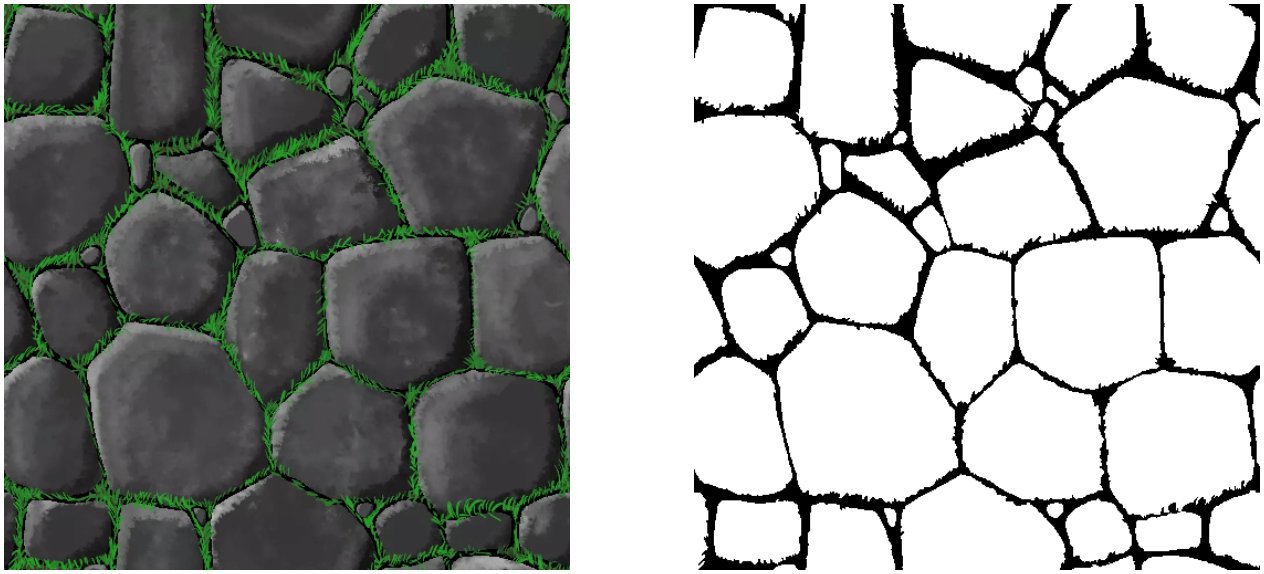


FIGURE 8 – Image *texture*8 de base et image segmentée de référence

Regardons maintenant les résultats de nos segmentations, à gauche la fonction de *openCV* et à droite *my_kmeans*. Ici j'utilise un *seuil* fixé à 120.

Même si visuellement, on semble être assez éloigné de l'image de référence dans les deux cas de *kmeans* avec des tâches noires qui apparaissent, les images obtenues montrent que la fonction *my_kmeans* est plus efficace que *OpenCV*, et les calculs de validation le prouvent. La précision, la sensibilité et par implication le coéfficient de similarité sont tous significativement plus importants pour *my_kmeans* que pour *kmeans*. La différence est plus frappante pour la sensibilité qui avoisine les $0,97$ ce qui est donc très satisfaisant. Une telle différence peut s'expliquer du fait du calcul des centroïdes à chaque itération, qui est plus précis et efficace sur *my_kmeans* que sur le *kmeans* fourni par *OpenCV*. L'image de référence est une segmentation parfaite, et encore une fois, les "ombres" jouent un role majeur dans la determination des classes lors des itérations de l'algorithme.

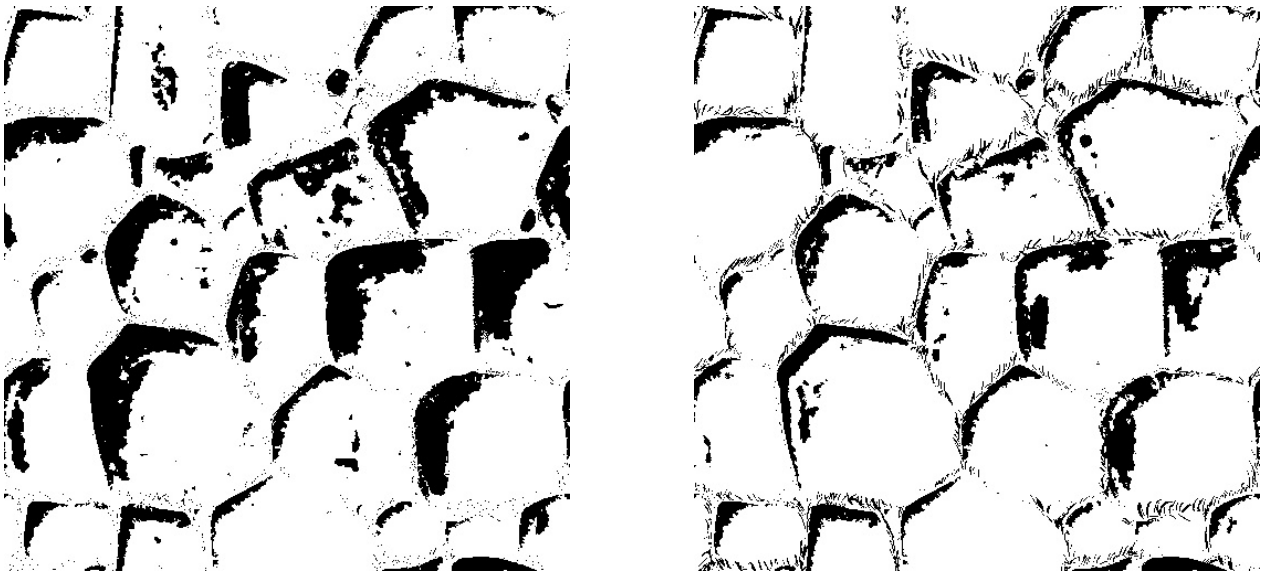
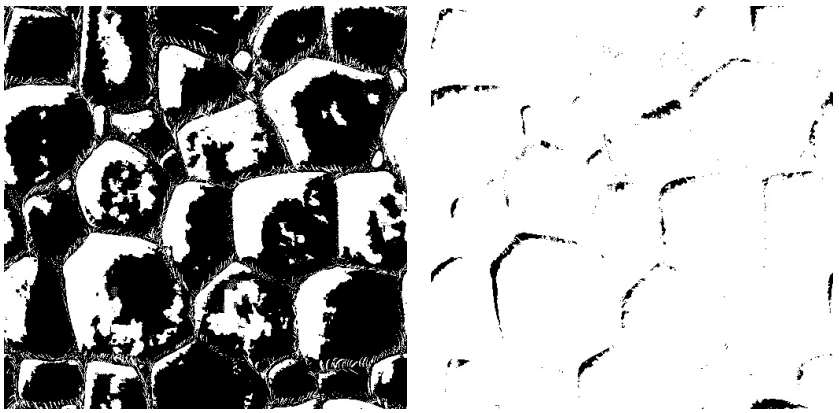


FIGURE 9 – Images *texture*8 segmentées *openCV* et *my_kmeans* pour $k = 3$

```
RESULTATS :

Precision openCV: 0.790721
Precision my_kmeans: 0.848905

Sensibilité openCV: 0.848526
Sensibilité my_kmeans: 0.971693

DICE openCV: 0.818604
DICE my_kmeans: 0.906158
```

FIGURE 10 – Evaluation de la qualité de segmentation de l'image 8 pour k=3

On peut aussi ajouter qu'en mettant $k = 2$ au lieu de 3, on obtient les images et les résulats suivants. L'image *OpenCV* est trop sombre et inversement l'image *my_kmeans* est trop faible. En revanche le coefficient de similarité pour *my_kmeans* est légèrement plus fort, autour de $0,985$ du fait d'une sensibilité quasiment à 1.



```
RESULTATS :

Precision openCV: 0.352006
Precision my_kmeans: 0.977421

Sensibilité openCV: 0.88253
Sensibilité my_kmeans: 0.995288

DICE openCV: 0.503276
DICE my_kmeans: 0.986274
```

FIGURE 11 – Images *texture*8 segmentées *openCV* et *my_kmeans* et résultats pour k=2

3.3 Image texture11

Voici ci-dessous les images que nous avions à disposition pour l'image *texture*11.

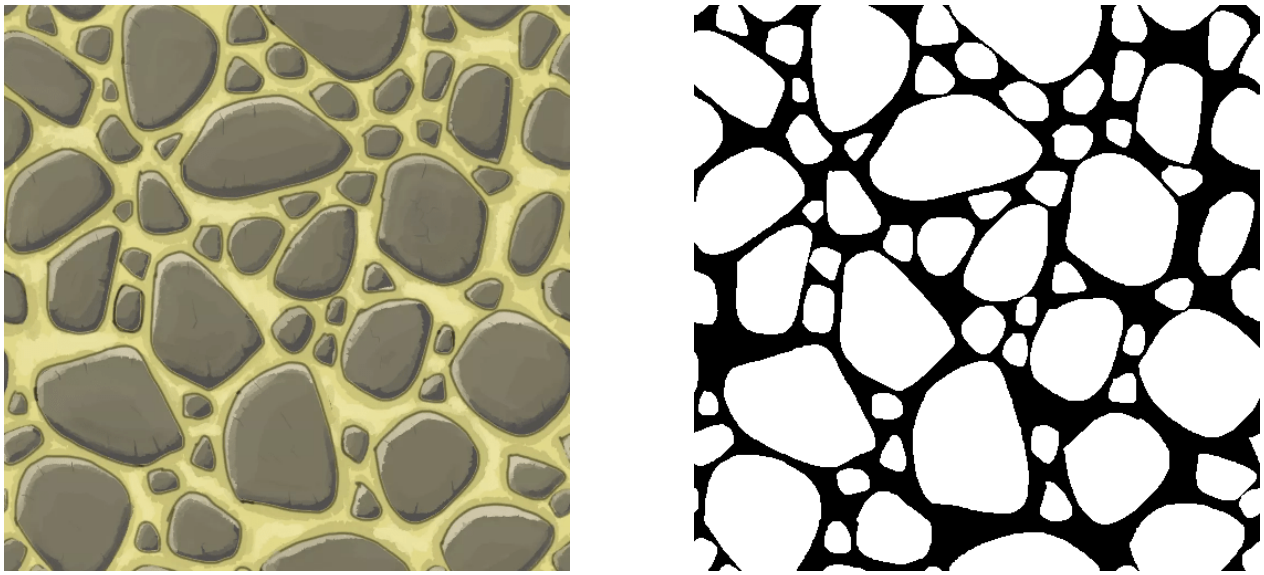


FIGURE 12 – Image *texture*11 de base et image segmentée de référence

Regardons maintenant les résultats de nos segmentations, à gauche la fonction de *openCV* et à droite *my_kmeans*. Ici j'utilise un *seuil* fixé à 100.

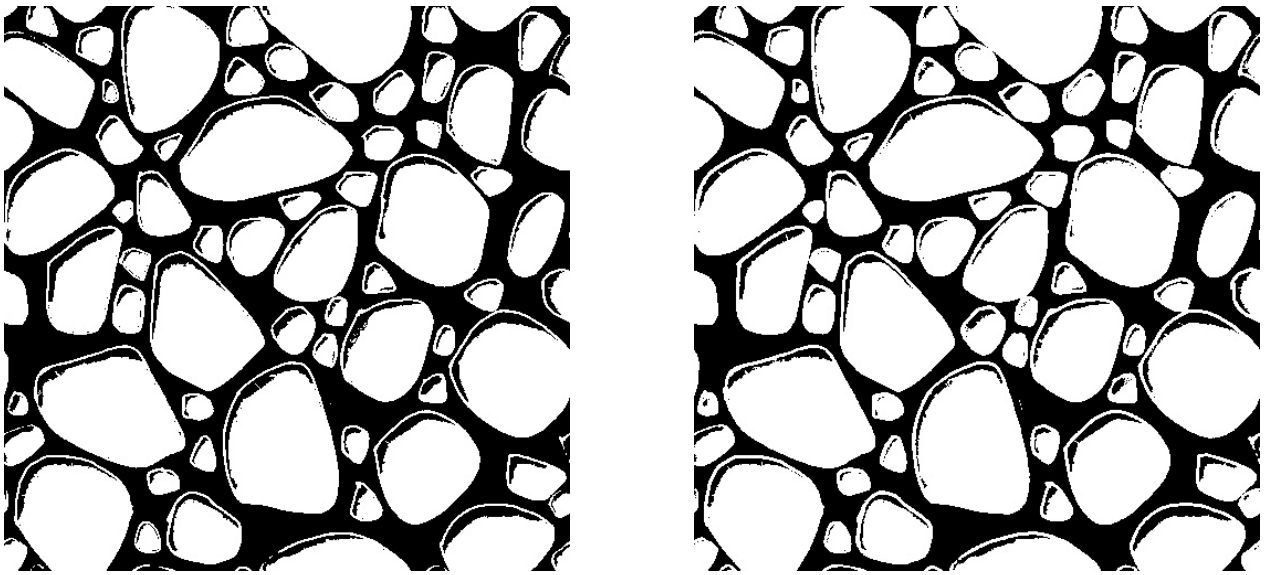


FIGURE 13 – Images *texture*11 segmentées *openCV* et *my_kmeans*

On retrouve le même problème des "ombres" des pierres qui entraînent un defaut d'attribution de la bonne classe. En revanche pour ces images, la segmentation semble plus épurée et mieux réalisée que ce soit du côté de *kmeans* ou de *my_kmeans*. Mais cela ne suffit pas pour avoir un coefficient de précision important, en effet il est particulièrement faible (autour de $0,66$). Comme dit précédemment, les "ombres" sont très certainement la cause de ce défaut.

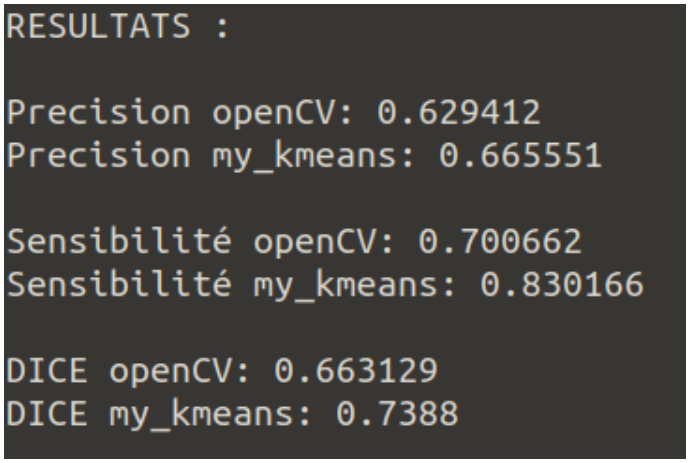


FIGURE 14 – Evaluation de la qualité de segmentation de l'image 11

4 Conclusion

Pour conclure sur ce TP, nous pouvons dire que la fonction *my_kmeans* est une segmentation plus efficace que *kmeans* de *OpenCV*, en revanche, il faudra combiner plusieurs algorithmes sur la même image pour obtenir une segmentation parfaite.
L'utilisation du seuil peut aussi causer quelques problèmes de précision, il existe surement une méthode plus efficace que de seuiller chaque image. Ceci peut être une piste d'amélioration.