Université Joseph Fourier

GRENOBLE

UFR IM²AG

Modèle géométrique pour l'image

LEFEVRE Henry - M2 MIA
SEGURET Aymeric - M2 MIA

Rapport de Travaux Pratiques

Seuiler une image

La fonction de seuillage Notre fonction de seuillage d'une image consiste à mettre à 0 tous les pixels ayant une valeur inférieure à un seuil choisi. Et à 255 tous ceux de valeur supérieure. Dans notre cas nous présentons 3 façons différentes pour choisir ce seuil :

1. Seuil fixe : la valeur est fixée à 150

2. Seuil moyen : correspond à la valeur moyenne des pixels

3. Seuil médian : correspond à la valeur médiane des pixels une fois triés

Négatif d'une image

Le négatif d'une image correspond à l'inversion de chaque pixel suivant la transformation linéaire suivante :

$$y = x - 255$$

Composantes Connexes

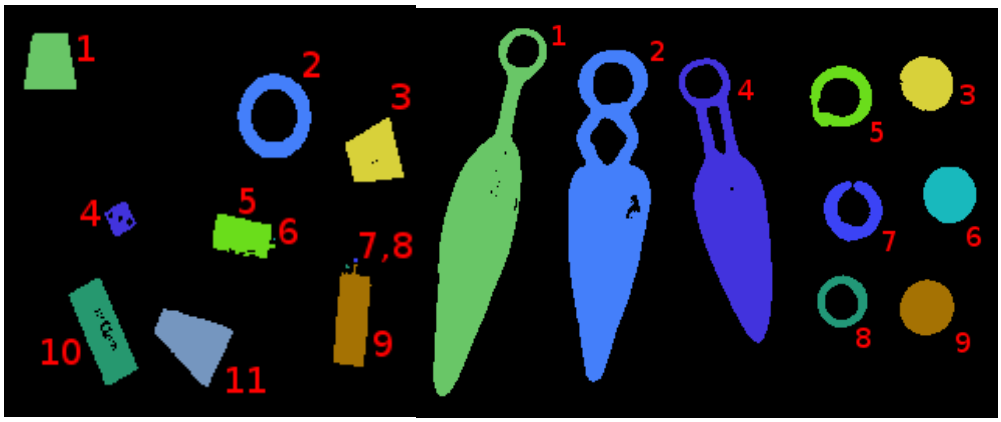
Le principe ici est de compter le nombre de composantes connexes présentent dans une image binaire, puis d'énumérer les trous dans chacune d'elles. Le noir représentant le fond de l'image, le blanc indiquant les objets à compter. Les objets sont en 8-connexité et le fond en 4-connexité.

Pour la première étape, l'algorithme se décompose en 2 phases :

1. étiquetage des objets de l'image avec enregistrement des équivalences
2. ré-étiquetage des classes d'équivalences

L'union-find est la méthode qui a été retenue afin de garder en mémoire les classes d'équivalences. Cette solution est implémentée en utilisant la forêt. Autrement dit, nous avons créé une structure appelée *Nœud* qui contient un entier définissant le numéro de la classe d'équivalence qu'il représente et un pointeur vers le parent de ce *Nœud*. Ce pointeur est égal à *Null* si le nœud est une racine.

Résultats obtenus après traitement :

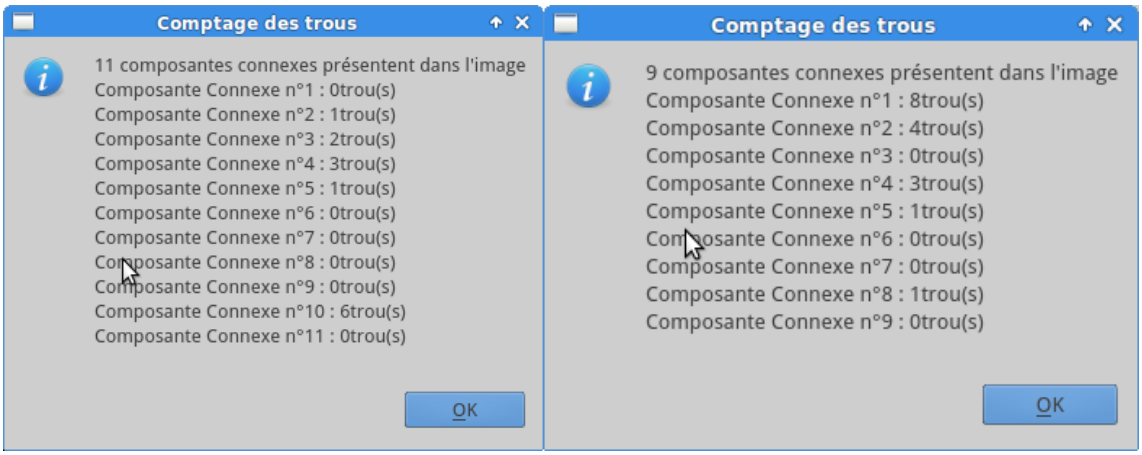


L'algorithme commence par seuiler l'image en entrée avant de la traiter. Par ailleurs, sur la première image les composantes connexes n° 6, 7 et 8 sont composées de quelques pixels isolés qui n'apparaissent pas clairement mais sont comptés comme tel.

La seconde étape pour compter les trous dans une composante connexe consiste à lancer l'algorithme précédent sur le négatif de l'image, et plus précisément sur chacune des composantes connexes. En effet, par ce procédé les objets comptabilisés appartenaient au fond de l'image précédente. Et inversement le fond est représenté par les anciens objets.

Il s'agit donc d'extraire les composantes connexes trouvées afin de les traiter comme des images indépendantes. Des sous-matrices sont construites (contenant chacune une composante) sur lesquelles est lancé l'algorithme précédent. Ainsi, le nombre de composantes connexes moins un (i.e. moins le fond qui est comptabilisé comme un objet) donne le nombre de trous recherchés

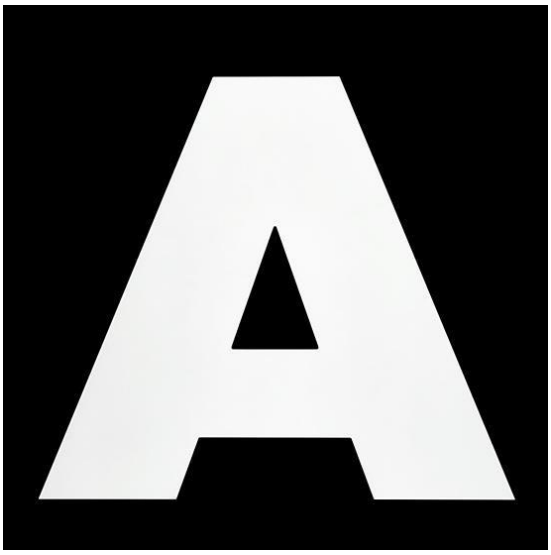
Sur les deux exemples précédents, nous trouvons comme résultats :



Il est évident que la qualité du seuillage influencera directement la qualité du résultat de cet algorithme. A partir du moment où un bruit apparaît dans une zone de l'image. L'idéal serait d'appliquer dilation puis une érosion sur l'image afin de supprimer ou du moins limiter un maximum ces artéfacts.

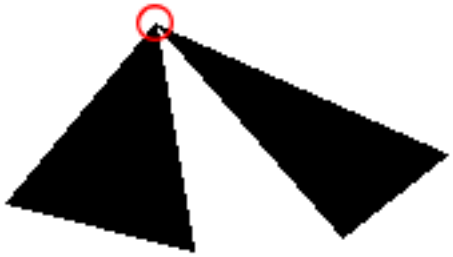
Calcul d'un bord d'un objet

On a découpé cette construction du code du Freeman en plusieurs fonctions. Une première fonction qui trouve le premier pixel blanc de l'image (parcours classique depuis le coin supérieur gauche). Une seconde fonction qui vérifie si le pixel courant est un pixel de bord (qui a un voisin en 4-connexité noir). Et une troisième fonction qui construit le code de Freeman. Pour cela, on part du premier pixel et on regarde les voisins 8-connexes dans un ordre précis afin de voir s'il s'agit d'un pixel de bord (deuxième fonction).



L'image ci-contre a été notre image de test. Après construction du code, nous l'avons vérifiée à la main (aucune méthode algorithme ne nous paraissait utilisable).

L'algorithme s'arrête lorsque l'on retrouve les deux premiers points. Ceci s'explique par le fait que si l'on se contente uniquement du premier point. Il se peut que l'on tombe sur un cas particulier où on ne parcourra pas toute la forme.



L'exemple si contre expose le problème des deux points.

Il existe un certain nombre de limites à l'algorithme implémenté :

1. dans notre cas on ne construit le code de Freeman que pour une seule figure, la première rencontrée.
2. Il se pose un problème, lorsque la figure est en contact avec le bord de l'image. Ce problème peut être facilement corrigé en ajoutant une bordure noire en prétraitement tout autour de l'image.

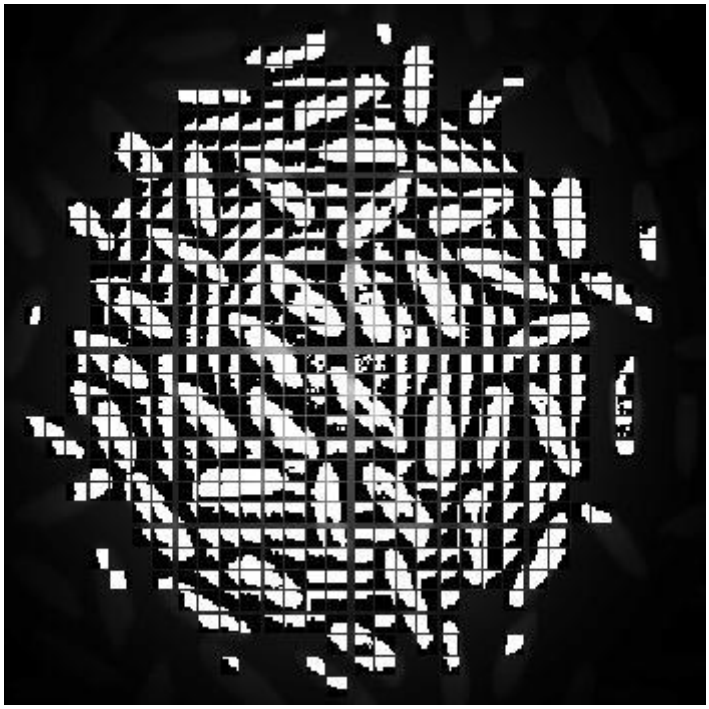
Il existe certainement d'autres limites qui ne nous semblent pas évidentes, ou bien que nous n'avons pas rencontrées.

Compter le nombre de grains sur une image mal éclairée

Notre idée principale pour compter les grains de riz de l'image est, dans un premier temps, d'effectuer un seuillage sur l'image. Cependant ce seuillage n'est pas l'un des trois présentés au-dessus. A cause de la luminosité très variable, il nous est impossible de travailler sur toute l'image. L'idée alors est de décomposer l'image en sections plus petites afin de travailler sur des portions où la luminosité est plus régulière. Puis d'appliquer un seuillage sur chacune de ces sections avec un seuil adapté. Le seuil d'une portion correspond à la moyenne de ses pixels.

Cependant, il faut noter qu'avec cette méthode, toutes les sections présenteront des pixels qui seront au-dessus et en dessous du seuil. Même s'il n'y a pas véritablement d'information à seuiler. Par exemple une section sans objet, donc qui présentera des variations de couleurs noires, apparaîtra en partie blanche après seuillage. On va supposer un seuil moyen minimal (fixé à 15) à partir duquel on suppose que tous les pixels de la section appartiennent au fond.

On a appliqué cette méthode dans l'exemple ci-dessous :



Le quadrillage artificiel a été laissé afin de mettre en avant la découpe de l'image. On voit nettement que certains grains de riz sont oubliés et que certaines cases noires sont malgré tout seuillées. Donc le résultat reste à peaufiner.

Une seconde idée pour répondre à la question consiste en une suite de traitements comme suit :

1. Etalement d'histogramme
2. Détection de contours (gradient)
3. Seuillage médian
4. Erosion

Après érosion, nous ne dilatons pas l'image, car cela aurait entraîné la superposition de certains grains de riz. Cependant, on voit que certains grains n'apparaissent pas suffisamment pour être pris en compte pour les composantes connexes. Le décompte est donc un peu faussé par ces artefacts. Ci-dessous on voit l'image que l'on obtient avec cette méthode ainsi que les composantes connexes associées (245 grains de riz comptés).