TP - Modèles géométriques et morphologiques pour l'image

Pierre-Antoine Augereau & Kévin Polisano

19/11/2012

1 Exercice 1 : Reconnaître des objets différents dans une image PGM

Nous travaillons dans cette partie sur l'image suivante :

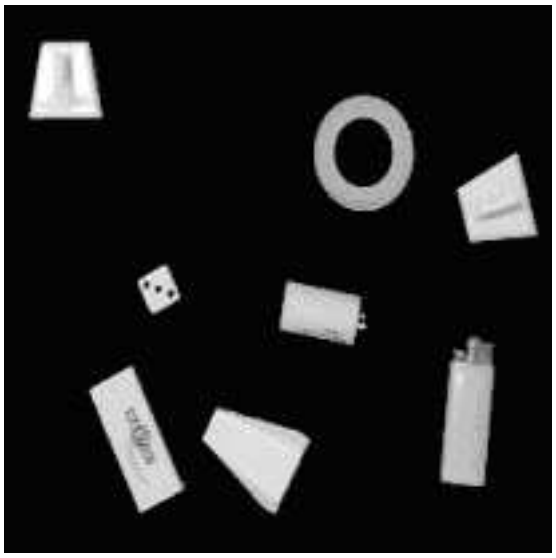

Figure 1: Image originale objets.pgm

1.1 Nombres d'objets contenus dans l'image

L'image `objets.pgm` étant en niveau de gris, il a fallu dans un premier temps décider de ce qui était ou non considéré comme objet dans l'image. Comme les objets apparaissent visuellement comme plus clair que le fond nous avons

naturellement seuillé l'image (typiquement ici nous avons pris 50), ainsi tous les pixels "objet" sont à 1 et les pixels du fond à 0. Nous pourrions envisager un seuillage automatique en utilisant l'histogramme mais ce n'est pas le point intéressant ici. Voici l'image binaire obtenue :

Figure 2: Image binaire obtenue par seuillage d'objets.pgm

Pour dissocier ensuite les objets de l'image nous avons implémenté et appliqué un algorithme d'étiquettage des composantes connexes. Le nombre d'objets est donc le nombre de composantes trouvées (on compte le nombre d'étiquettes différentes utilisées à l'issue de l'algorithme).

1.2 Nombre de trous de chaque objet

On souhaite désormais déterminer le nombre de trous de chaque objet, ce qui implique de pouvoir travailler sur chaque objet de manière isolée. C'est ce que nous avons donc fait en créant une image différente pour chaque objet. Pour cela il suffit de balayer l'image comportant l'étiquettage des composantes connexes, et pour chaque étiquette différente ne conserver que les pixels qui ont cette valeur. Voici donc ci-dessous les 8 objets isolés dans une image différente :

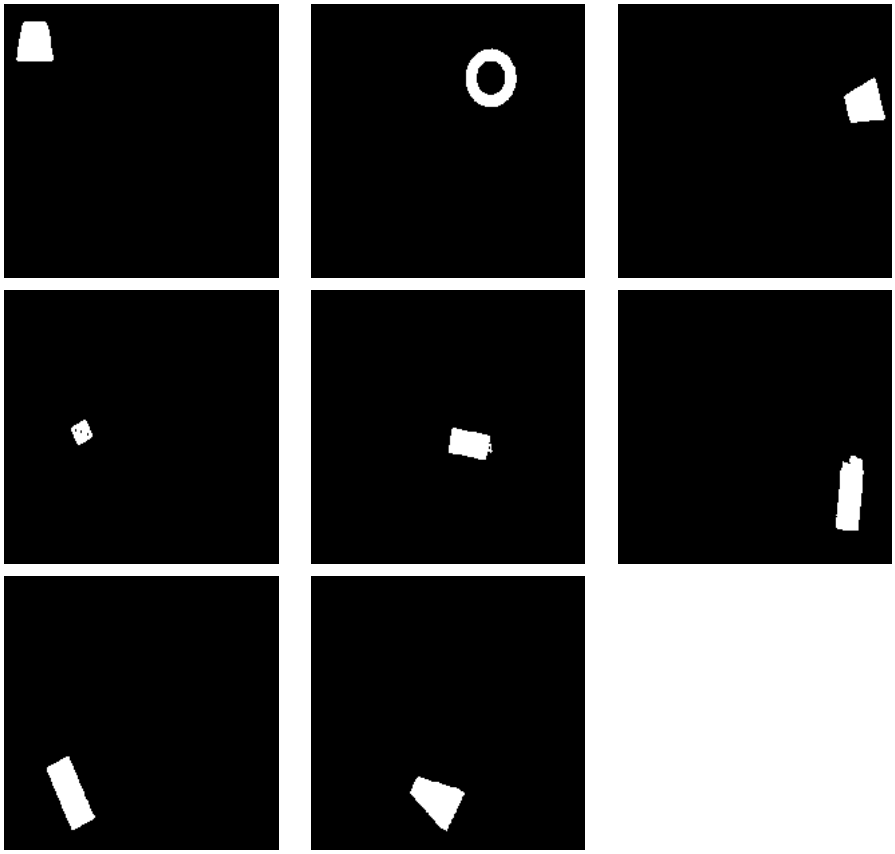

Figure 3: Images contenant chaque objet isolé

1.3 Epaisseur moyenne d'un objet

Voici la façon dont nous envisageons estimer l'épaisseur moyenne d'un objet. Tout d'abord on appliquerai à l'objet (pris de manière isolée comme précédemment) une transformée en distance. A partir de celle-ci on est en mesure de déterminer l'ensemble des boules incluses dans l'objet, puis l'axe médian. On peut alors calculer la moyenne du rayon des boules le constituant, ce que nous prenons comme définition de l'épaisseur moyenne de l'objet. Il est possible d'obtenir davantage d'information sur l'objet en calculant l'écart type des rayons (pour savoir comment varie l'épaisseur autour de la moyenne), ou bien faire un histogramme sur la valeur du rayon. Si on observe par exemple deux modes dans l'histogramme on peut dire que l'objet est composé de parties de deux épaisseurs différentes (typiquement une haltère).

1.4 Longueur d'un objet

Nous avons réfléchi à plusieurs façons d'estimer la longueur d'un objet, dépendantes de la définition que l'on se donne de la longueur.

Une première approche, qui a le mérite d'être simple à implémenter, consiste à calculer la boite englobante de l'objet et de poser comme définition de la longueuer de l'objet, celle du plan grand côté du rectangle englobant.

L'approximation est plutôt bonne pour un objet convexe qui est parallèle à un des axes (x) ou (y). Si l'objet est incliné de 45 degrés par exemple alors l'estimation est moins bonne, c'est pourquoi nous avons pensé utiliser plutôt la diagonale du rectangle englobant, qui donnera en moyenne une meilleure estimation pour des objets de quelconque orientation.

Toujours dans la même optique on a pensé se ramenait au cas "parallèle", en calculant au préalable l'orientation de l'objet (par analyse en composantes principales), puis en effectuant une rotation de l'image.

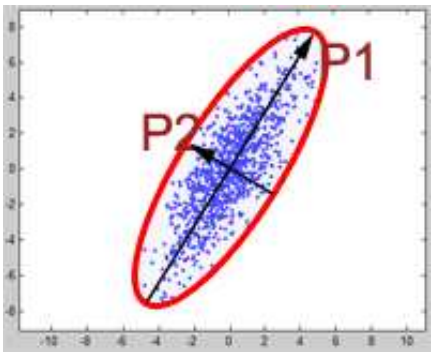

Figure 4: Orientation de l'objet par PCA

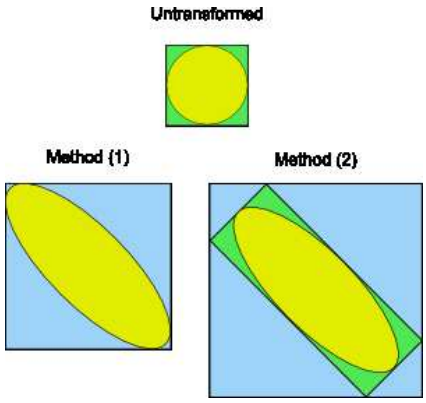

Figure 5: En haut on utilise la largeur de la boite englobante, à gauche la diagonale de la boite, et à droite on applique la PCA puis une rotation

En revanche pour un objet non convexe cette approche se révèle non efficace, c'est pourquoi nous avons posé une autre définition, se basant sur l'obtention du squelette homotopique (par amincissement, érodés ultimes, ou autre méthode). Une fois déterminé, le squelette peut se voir comme un graphe, et nous posons comme définition de la longueur le plus long chemin entre deux extrémités du graphe (dual des algorithmes classiques de type Dijkstra). Typiquement un squelette dont le graphe est cyclique n'aura pas de longueur. La définition ne sera cependant pas très bien adaptée pour des objets avec beaucoup de branches (par exemple une étoile).

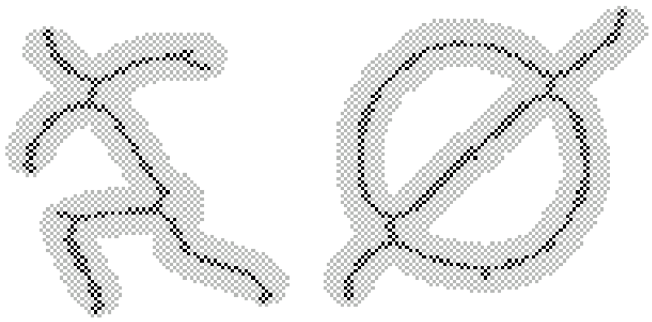

Figure 6: Squelette homotopique, à gauche la longueur est celle du chemin allant de la tête au pied à droite ; non définie pour un graphe cyclique

2 Exercice 2 : Compter un nombre de grains sur une image mal éclairée

2.1 Implémentation

Pour simplifier l'implémentation des opérateurs morphologiques, on présume que la fonction caractéristique est nulle sur son domaine de définition. Implémenter l'érosion et la dilation reviennent alors à implémenter l'addition et la soustraction pour de telles fonctions caractéristiques. Nous avons utilisés pour cela notre classe image en C++ que l'on a codé lors des TD de vision par ordinateur et de géometrie discrète.

Les fonctions caractéristiques sont stockées sous forme d'une image binaire où tous les points à 1 sont considérés dans le domaine de définition alors que les points à 0 ne le sont pas. Ensuite nous avons créé une fonction qui à partir de l'image de la fonction caractéristique et d'une coordonnée dans l'image renvoie un vecteur contenant tous les voisins du point (voisinage au sens du domaine de définition de la fonction caractéristique).

Cette implémentation nous permet de définir très rapidement des fonctions caractéristiques à l'aide d'un logiciel de manipulation d'image type Gimp, et l'enregistrement en pgm de type P2 nous permet aussi de les créer facilement avec un éditeur de texte.

Cette implémentation peut très facilement être modifié pour admettre des fonctions caractéristiques non nulles, soit en rajoutant une image donnant les valeurs en chaque point du Df, soit en rajoutant une composante dans l'image de base (donc pgm type P3) (par exemple enregistrer l'image en couleur avec le rouge correspondant au Df et le vert et/ou le bleu au valeur en chaque point du Df).

Afin de gérer les bords, nous voyons deux possibilités, soit tester à chaque fois si le voisin que l'on souhaite rajouter dans le vecteur existe ce qui est très coûteux, soit rajouter un cadre de pixel de valeur -1 de taille correspondant au rayon du Df.

L'implémentation de l'addition (respectivement de la soustraction) consiste à récupérer les voisins via la fonction précédente, puis de determiner le maximum (respectivement le minimum) du vecteur. Pour gérer les bords, lors de l'addition on considère que le vecteur est un vecteur d'entiers, ainsi

les bords seront à -1 et ne seront jamais les maximum. Pour la soustraction on travaille en unsigned int, ainsi les bords seront à -1 donc a $2^{32} - 1$ et par suite ne seront jamais les minimum. Cette astuce nous permet de remplacer les 2*n*m*p vérificationsde condition(avec n*m les dimensions de l'image et p le nombre de pixels du Df) par 2 copies de vecteur et une complexité du min/max qui augmente légèrement sur les bords mais qui reste néanmoins plus que négligeable. (car les vecteurs sont de taille fixe alors que précédemment sur les bords ils étaient plus petits)

Les opérateurs morphologiques plus complexes ne sont ensuite que des dérivés de ces deux opérateurs, il suffit de les combiner entre eux afin de créer l'opérateur dont on a besoin. Les fonctions addm(masque) et subm(masque) correspondent respectivement à la dilatation et à l'érosion avec un masque où toutes les valeurs sont à 0, et sub (image) fait la difference de deux images. La liste des masques que l'on a utilisé est la suivante : masque7, masque11, masque21,masque35 correspondent à des boules du diamètre indiqué ; masque9.9 et 21.21 à des carrés, 21.3 et 3.21 à une ligne droite horizontale et verticale, 21y à la diagonole y=x et 21x à y=-x.

2.2 Résultats obtenus

Nous travaillons sur l'image suivante, l'objectif étant de compter le nombre de grains de riz présents dans l'image en utilisant des procédés de morphologie mathématiques.

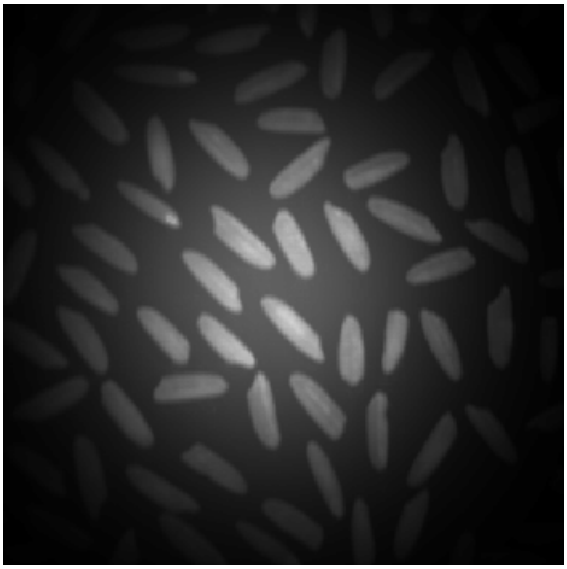

Figure 7: Image originale riz.pgm

Premier problème : la présence d'un halo lumineux dans le fond rendant impossible de faire directement un seuillage. En effet voilà ce que nous obtenons avec un seuillage à 60 et à 100 :

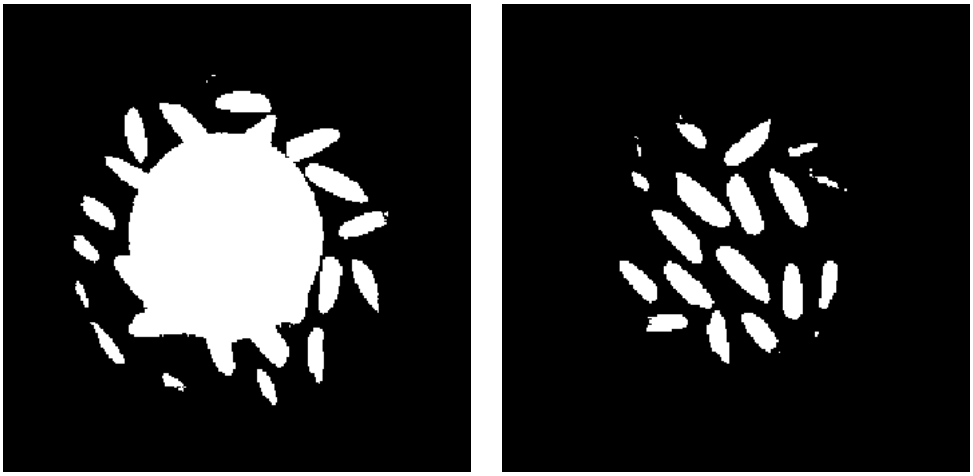

Figure 8: Image directement seuillée à 60 et à 100

Par conséquent nous effectuons au préalable une technique de correction de fond (image - ouverture de l'image) afin de supprimer le halo

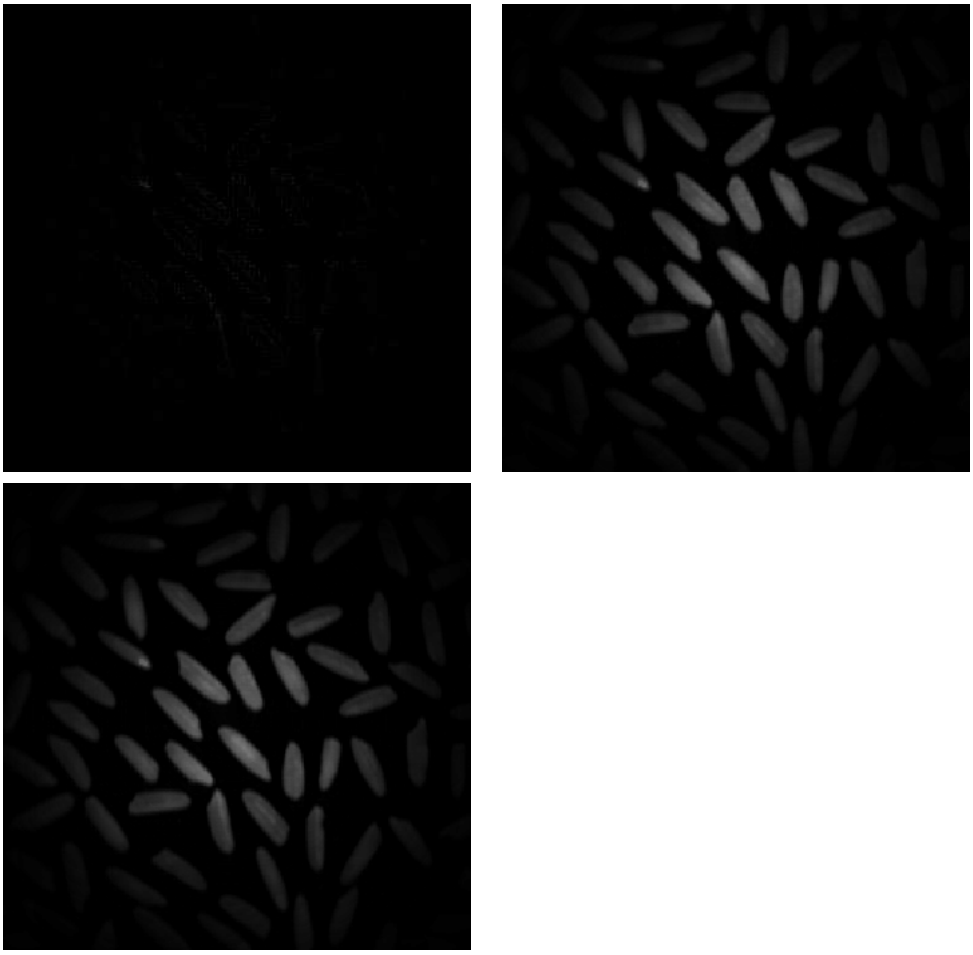

Figure 9: Correction de fond en utilisant une boule de diamètre 5, 21 et 35

On voit bien que les résultats sont dépendants de la taille du masque. Par exemple la fonction caracteristique definis sur une boule de diamètre 5, le Df étant plus petit que les grains de riz, ceux-ci se font rogner lors de l'ouverture et sont donc considérés comme faisant partis du fond, tandis que pour un diamètre de 21 et de 35 le fond est bien corrigé car le Df est plus grand qu'un grain de riz, cependant il reste du bruit. Pour enlever le bruit, soit on applique avant la correction un filtre gaussien sur l'image, soit on seuile davantage, au risque de perdre certains grains de riz situés aux extremités, là où l'ombre est vraiment trop importante pour faire quoi que ce soit. A l'inverse si on ne seuile pas suffisamment nous avons trop de bruit et des composantes connexes qui fusionnent.

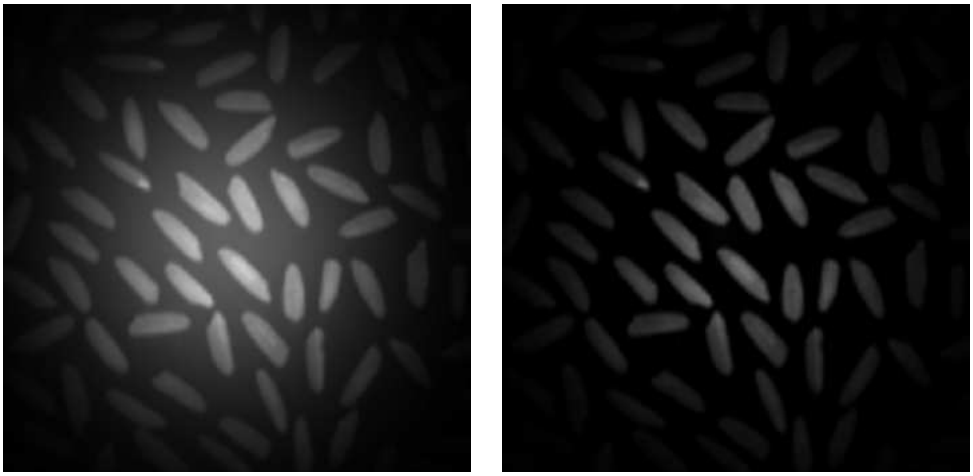

Figure 10: Application préalable d'un filtre gaussien puis la même image avec le fond corrigé

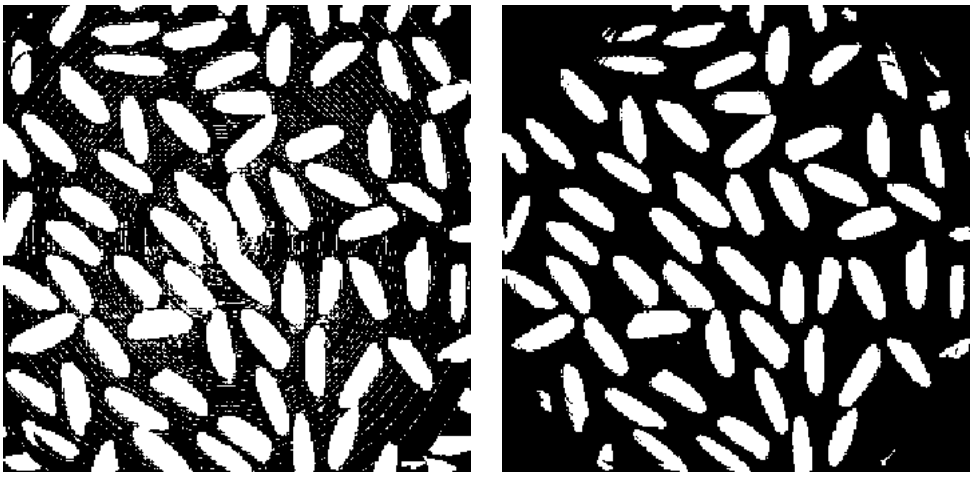

Figure 11: seuil trop faible, puis trop fort

On peut récupérer le bruit à l'aide d'une correction de fond grâce à une fonction carateristique ayant un Df plus petit que les grains de riz. Il suffit ensuite de soustraire le bruit à l'image originale (cela fonctionne aussi bien en niveau de gris qu'en binaire).

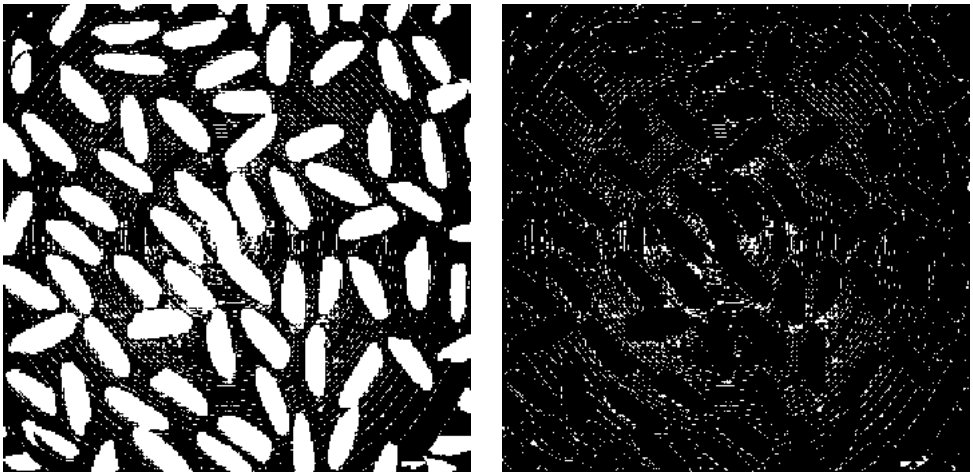

Figure 12: Soustraction de l'image de bruit (à droite) à l'image de gauche

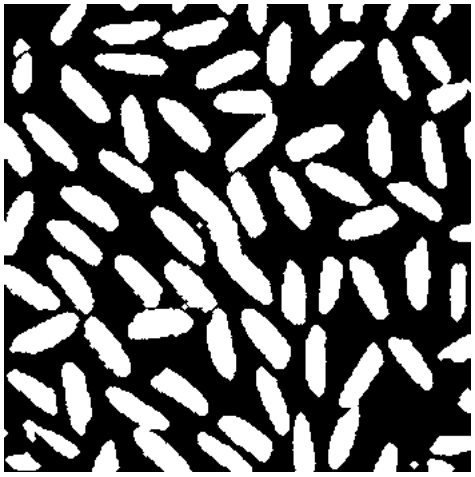

Figure 13: Résultat de la suppression de bruit

En ce qui concerne les composantes connexes qui fusionnent on peut utiliser des Df en ligne,qu'il suffirait de débruiter via la technique ci dessus et de trouver une façon de les combiner entre elles pour bien séparer les composantes connexes, tout en conservant tous les grains de riz.

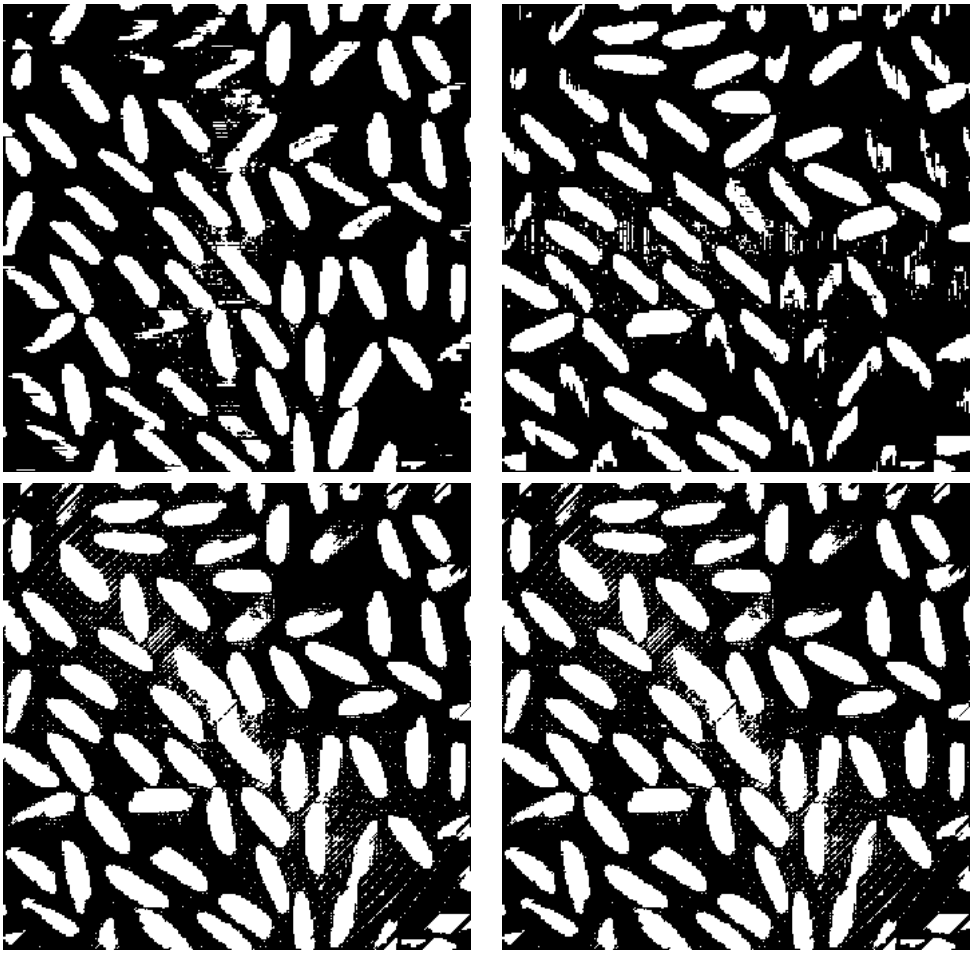

Figure 14: Df horizontal, vertical, diagonal (y=x) et diagonal (y=-x)

On voit bien pour chaque Df que le centre ne nous pose plus de problème cependant certains grains de riz sont rogné par l'ouverture, selon leur orientation par rapport a l'élement structurant.

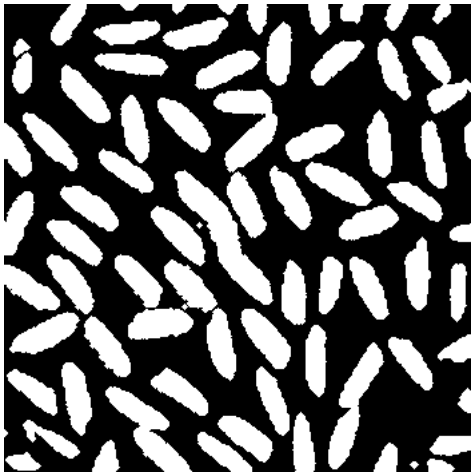

Figure 15: Résultat de la suppression de bruit

Sinon, si le but est seulement de compter les composantes mais pas de les mesurer, une séparation de type montée des eaux pourrait être envisagée une fois l'image debruitée et seuillée. Nous avons par ailleurs essayer de jouer avec le gradient dans l'optique de récupérer les bords afin de séparer plus facilement les grains de riz tout en gardant leur caractéristique, ce que ne permet pas la montée des eaux. Toutefois on est confronté au même problème que pour l'image de base, à savoir un seuillage difficle (voir riz_grad_5_seuil_3). Donc en faisant le gradient sur l'image avec correction de fond et débruitée (riz_correct_35_debruit) on obtient riz_correct_35_debruit_grad_5_seuil.

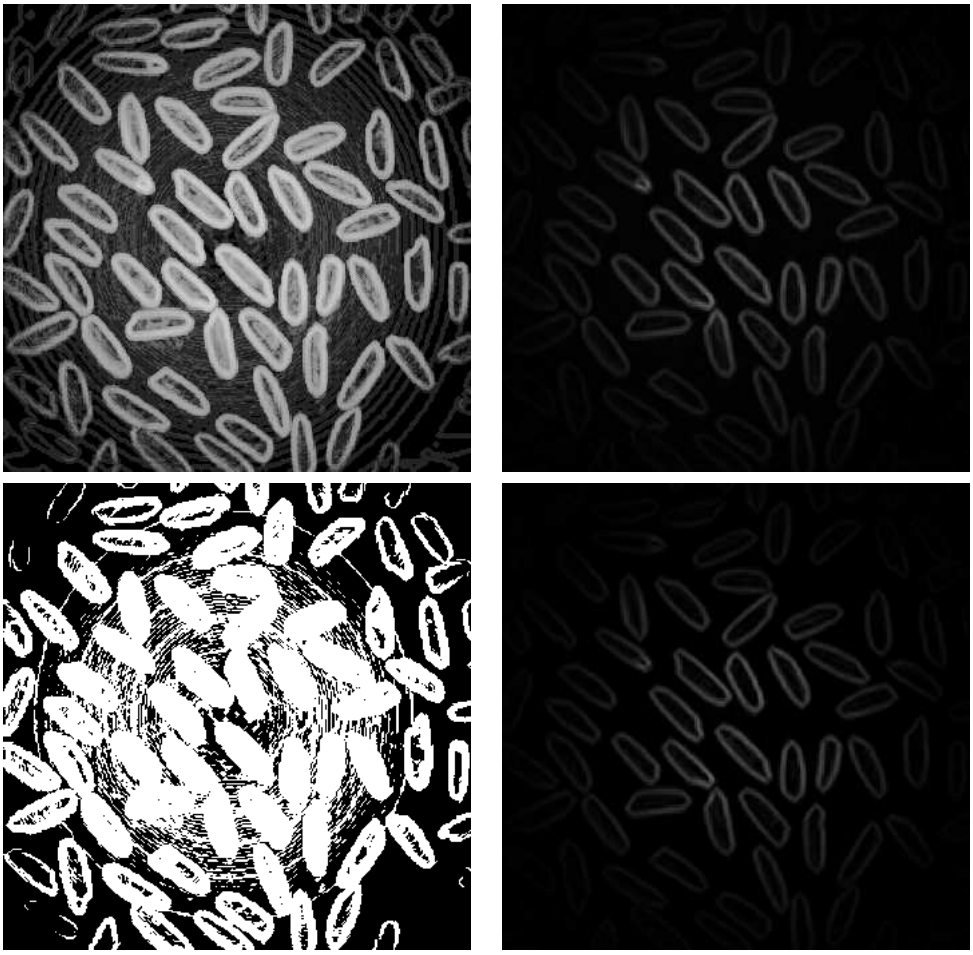

Figure 16: De gauche à droite et de haut en bas : gradient de l'image riz.pgm (rizgrad), gradient sur celle sans halo (riz_grad_5), puis avec seuil (riz_grad_5_seuil_3), et enfin avec débruitage riz_correct_35_debruit_grad_5_seuil

Finalement on se retrouve avec riz_final.png qui montre bien le dilemme entre avoir les coins et ne pas connecter les grains de riz du centre. (image créée à partir des bords seuillés, et de 2 images avec fond corrigé et debruité mais 2 valeurs de seuillage différentes), ainsi tout ce qui est en gris foncé possède la même couleur et apparaitra en même temps au seuillage.

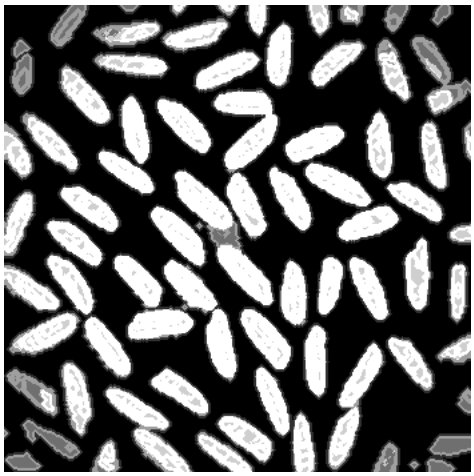

Figure 17: Mise en évidence du compromis (riz_final) : avoir les grains dans les coins versus déconnecter les grains au centre

Enfin, on applique notre algorithme d'étiquettage des composantes connexes :

Figure 18: Comptage des composantes connexes

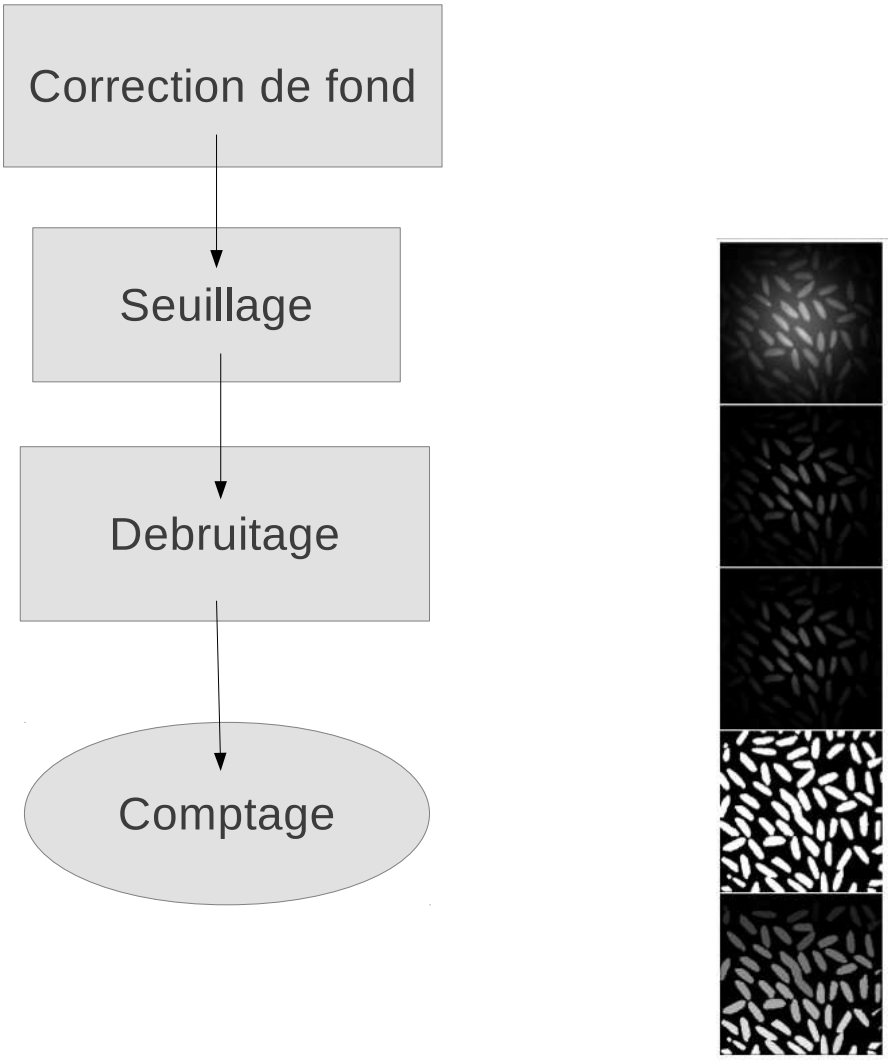

Figure 19: Schéma synoptique