ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

Université de Rennes

TP2 TIA

Morphologies mathématiques

14 NOVEMBRE 2024

Axel PLESSIS
Damien VAILLAND

Classe : ESIR 2 IN

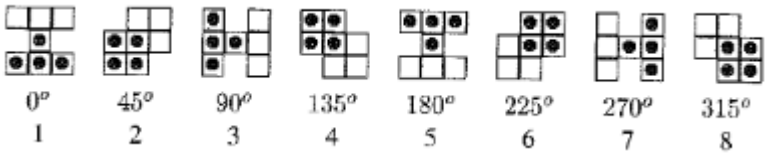
Introduction

Dans le cadre des cours de traitement de l'image avancé, nous réalisons un travail pratique. L'objectif de ce TP est de réaliser des algorithmes avancés de morphologie mathématique en langage C++.

Dans un premier temps nous nous pencherons sur la squelettisation par amincissement d'éléments binaires. Puis nous évoquerons dans un second temps la transformation en distance. Enfin, dans une dernière partie, nous utiliserons ces algorithmes dans l'optique de résoudre un problème.

1. Squelettisation

Le squelette par amincissement consiste en une succession d'amincissement par une liste d'éléments structurants. En l'occurrence, nous utiliserons la lettre L de l'alphabet de Golay qui définit des listes d'éléments structurants pour chaque lettre de l'alphabet tels que :



Cette opération détermine le squelette d'une structure binaire de largeur minimale qui suit la forme de la structure. Elle est définit selon cette formule :

$$Skel(S) = (S \odot T)^{\infty} = (S - (S \circledast T))^{\infty}$$

sachant que l'amincissement est défini tel que :

$$S \odot T = S - (S \circledast T)$$

Nous commençons par implémenter un algorithme squelettisation par amincissement par la lettre L de golay. Nous créons une boucle qui, à chaque tour, appelle la fonction `toutourien` qui ramène à 0 les pixels correspondant aux différents éléments structurants. Lorsque l'image n'est plus modifiée par le contenu de la boucle, l'algorithme s'arrête.

Pour un premier essai, nous entrons l'image d'une structure carré pour lequel l'algorithme est supposé retourner une croix composée des deux diagonales.

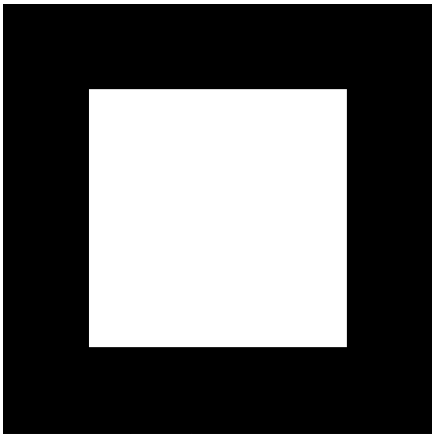


Image originale

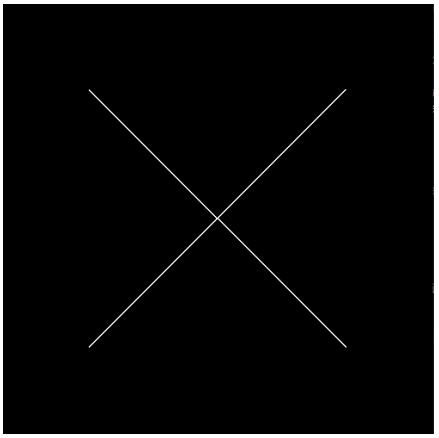


Image squelettisée

Comme attendu, l'image obtenue est une croix de largeur de 1 pixel.
Nous portons attention au nombre d'itérations qu'il a fallu à l'algorithme pour obtenir le squelette final. Celui-ci a dû faire 150 fois la boucle des 8 éléments structurants.

Notre algorithme est fonctionnel pour un élément carré, il s'agit désormais d'appliquer notre méthode sur une image binaire composées d'éléments divers :

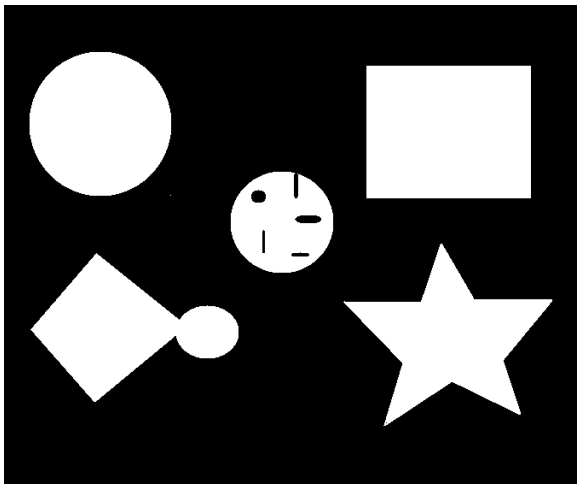


Image originale

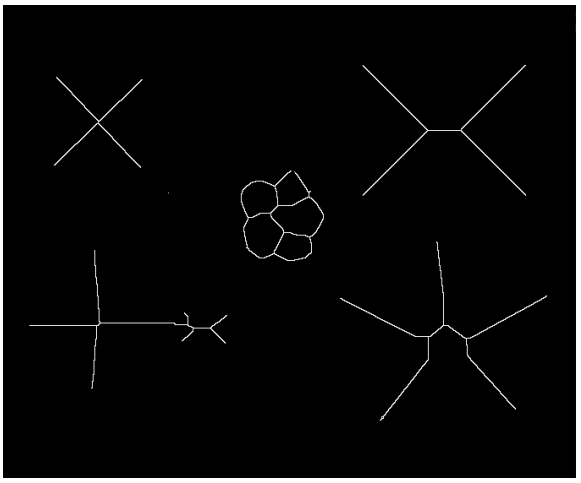
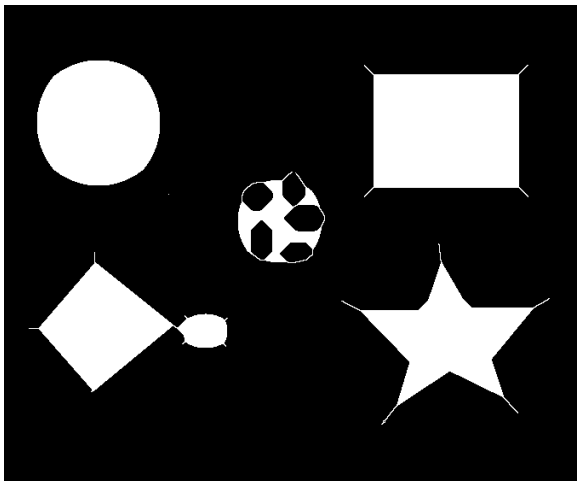


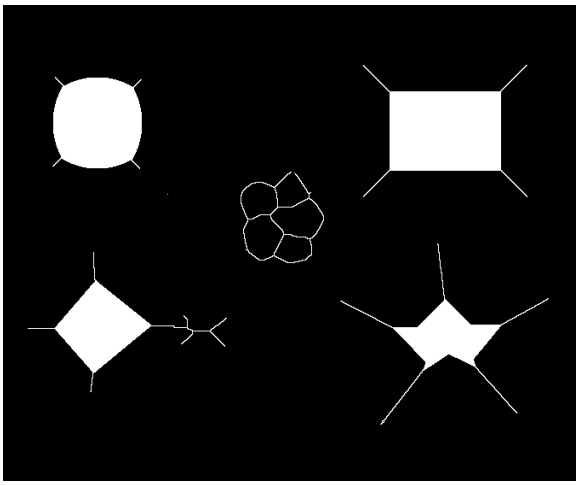
Image squelettisée

Cette fois-ci, il a fallu uniquement 80 itérations pour obtenir les squelettes finaux. Cette différence s'explique par la taille des éléments. En effet, malgré le fait qu'ils soient plus nombreux pour la deuxième image, le carré est bien plus grand sur la première, l'algorithme a alors besoin de plus d'itération pour venir à bout de la squelettisation.

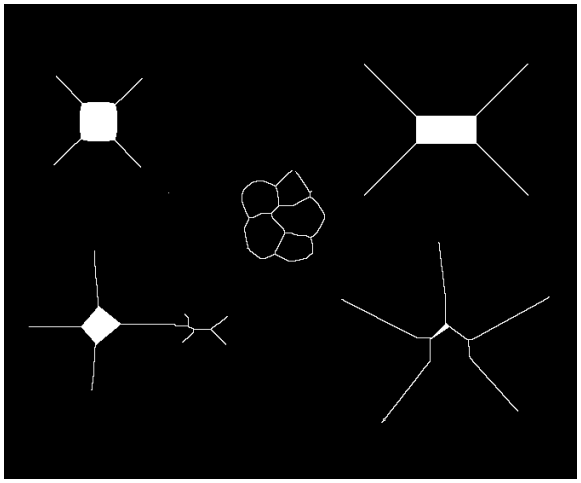
Enfin, nous constatons l'avancement de notre algorithme pour un nombre donné d'itérations avant le résultat final :



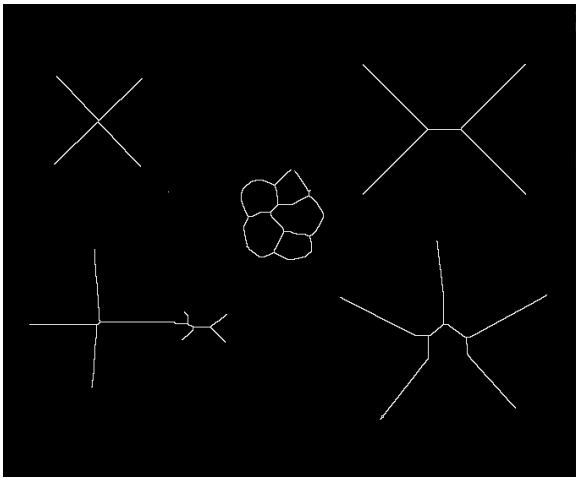
10 itérations



30 itérations



60 itérations



84 itérations

Il est intéressant de noter que les squelettes composés d'angles droits nécessitent davantage d'itération que les autres éléments. Ceci est dû à la complexité qu'un angle droit représente pour l'algorithme d'amincissement.

2. Transformation en distance

La transformation en distance consiste en la création d'une matrice dont les points sont calculés en fonction de la distance par rapport à un élément de l'image d'entrée. Il existe plusieurs transformée en distance.

2.1 Transformée en distance L1

La transformée en distance L1 appuie son calcul de la distance à partir d'une connexité à 4 voisins. Nous implémentons un premier algorithme de transformée en distance à partir d'un masque permettant d'appliquer la connexité à 4 voisins tel que :

```
0 1 0
1 1 1
0 1 0
```

L'algorithme de transformée en distance L1 applique un premier balayage direct avec la partie supérieure gauche du masque, puis un second balayage rétrograde, en ajoutant 1 à tous les éléments détectés par cette première partie.

```
0 1 0
1 1 0
0 0 0
```

Masque balayage direct

```
0 0 0
0 1 1
0 1 0
```

Masque balayage rétrograde

Nous appliquons ce premier algorithme de transformation en distance au même rectangle que dans la partie précédente. L'image attendue est alors un rectangle noir à la place de l'élément blanc initial, puis un dégradé de plus en plus clair vers les bords. La connexité étant à 4 voisins, les coins apparaîtront 2 fois plus clairs que les bords uniques puisque qu'il faudra 2 fois plus d'itérations pour accéder à ces pixels.

Nous obtenons le résultat suivant :

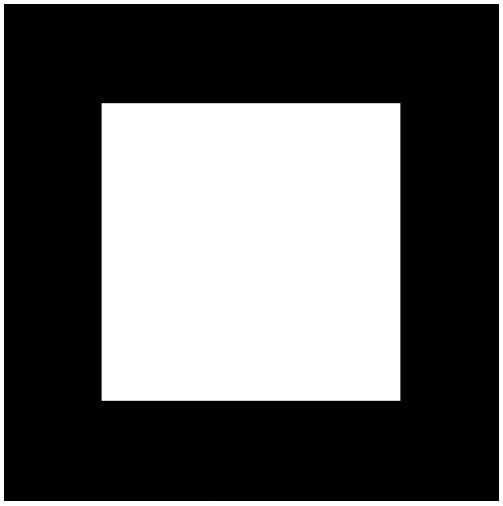
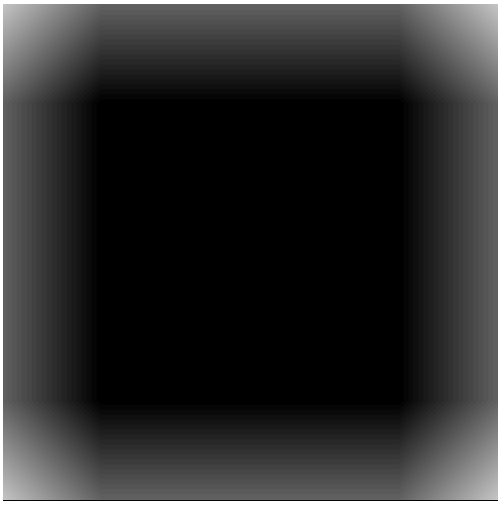


Image originale



Transformée en distance L1 (voisin 4)

Le résultat est cohérent et en concordance avec nos attentes. Nous retrouvons l'élément de base à 0, les bords de plus en plus clairs et les coins qui se démarquent.

L'algorithme étant fonctionnel, nous l'appliquons à la seconde image présentant plusieurs éléments plus complexes :

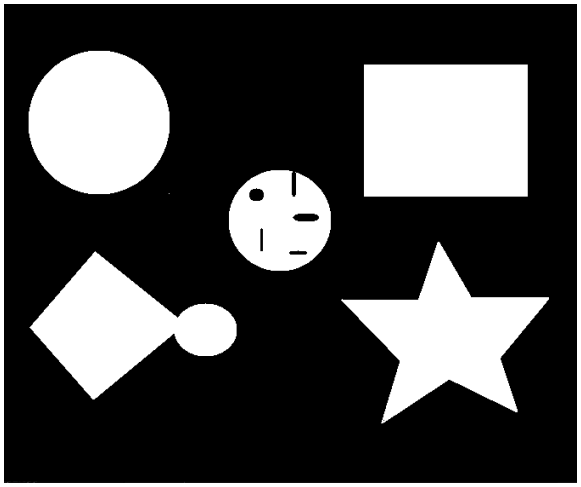
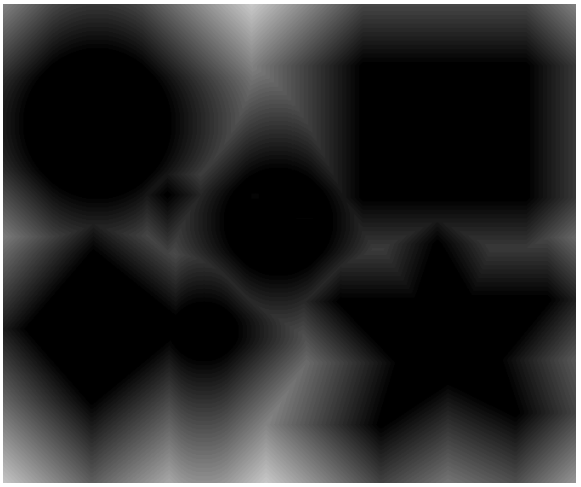


Image originale



Transformée en distance L1 (voisin 4)

Le résultat est conforme. Plus les éléments sont écartés, plus la frontière est marquée.

2.2 Transformée en distance max

La transformée en distance par max, utilise une connexité 8 voisins au contraire de la première transformation. Nous adaptons à nouveau cette connexité sous forme d'un masque :

```
1 1 1
1 1 1
1 1 1
```

Décomposé en 2 balayages appliqués l'un après l'autre tels que :

```
1 1 0
1 1 0
1 0 0
```

Masque balayage direct

```
0 0 1
0 1 1
0 1 1
```

Masque balayage rétrograde

Nous lançons ce nouvel algorithme à nouveau sur le carré. Cette fois-ci nous nous attendons au même carré noir, mais à des bordures et des coins en dégradé.

Nous obtenons le résultat suivant :

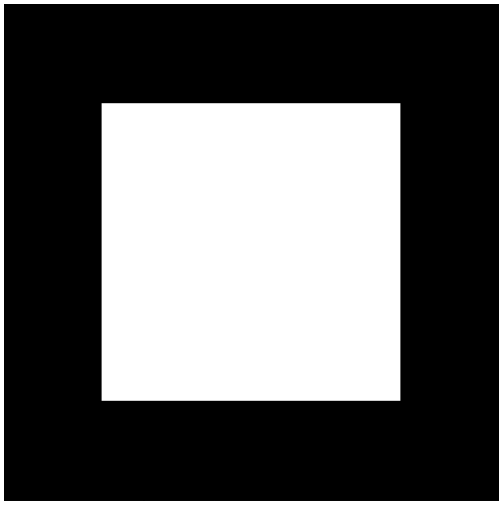
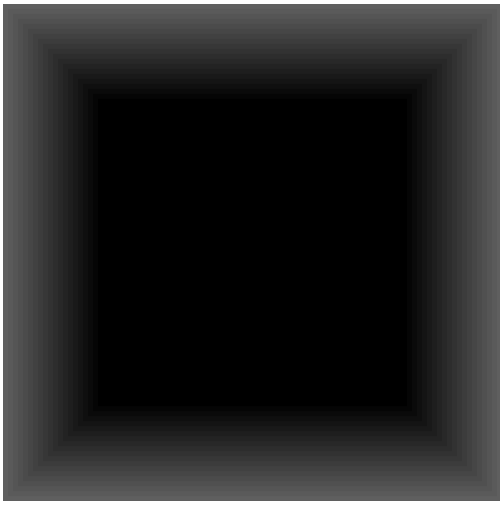


Image originale



Transformée en distance max (voisin 8)

L'image rendue est conforme aux attentes. Et nous appliquons notre algorithme de transformée en distance max à nouveau notre image composée d'éléments plus complexe :

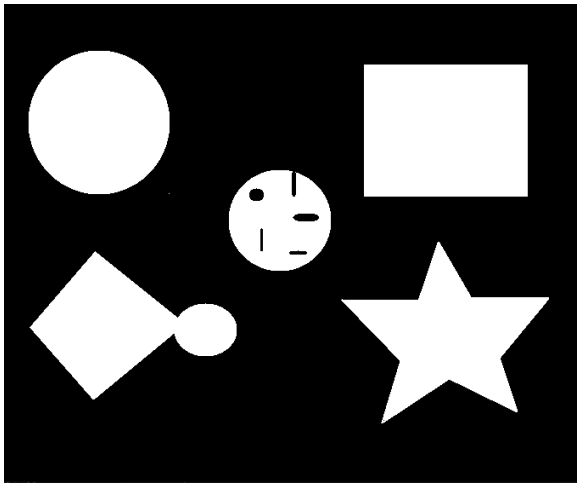
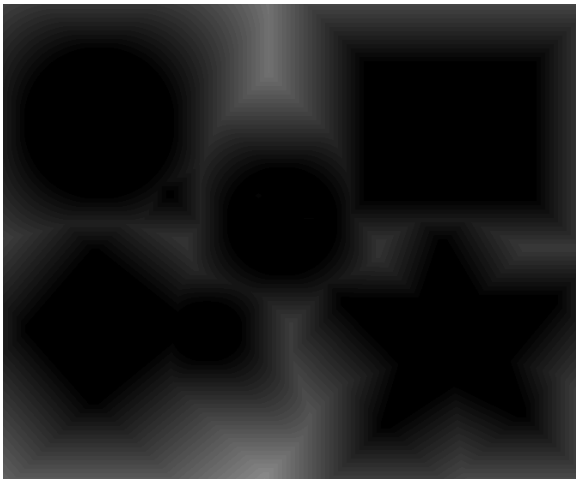


Image originale



Transformée en distance max (voisin 8)

Comparé à la première méthode à 4 voisins, les angles sont plus adoucis et les frontières sont moins marquées et angulaires.

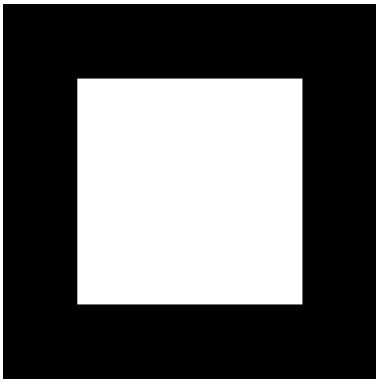
2.3 Érosion à base de transformée de distance

Nous cherchons désormais à créer un algorithme de calcul d'une érosion, en utilisant la transformée en distance.

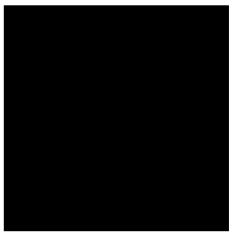
La transformée à distance conserve les éléments blancs et applique un dégradé à leur bordure. Ainsi nous relevons que la binarisation d'une transformée à distance selon un seuil défini permettrait de réaliser une dilatation.

Réciproquement, en inversant d'abord notre image binaire, puis en appliquant notre transformée en distance, et une même binarisation, nous obtiendrons une opération semblable à une érosion.

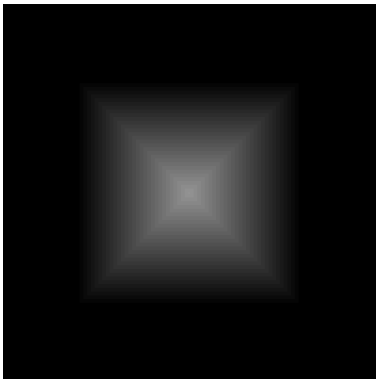
Nous implémentons une méthode d'inversion d'une image binaire, puis une méthode de binarisation selon une variable seuil en entrée. Nous essayons notre algorithme selon différentes valeurs :



Étape 1 : Image originale

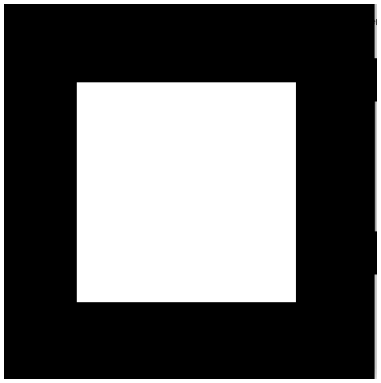


Étape 2 : Inversion



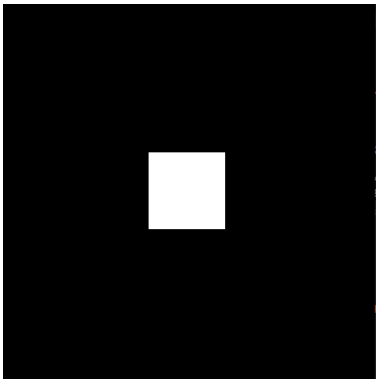
Étape 3 : Transformée en distance de l'étape 2

Étape 4 : binarisation de l'étape 3 selon un seuil S différent



Binarisation S=5

Binarisation S=50



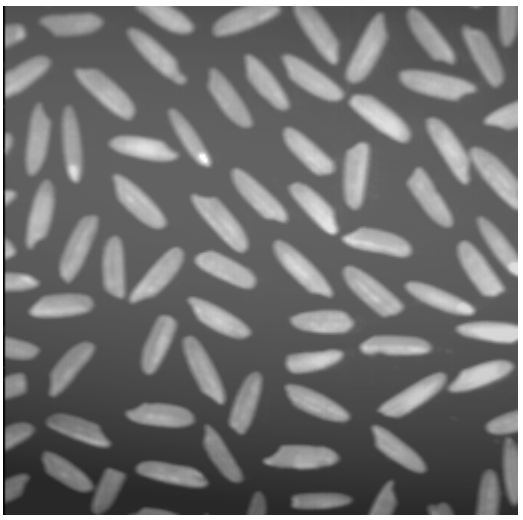
Binarisation S=100

L'évolution du dégradé allant de 1 en 1, nous retrouvons exactement un algorithme d'érosion avec en paramètre la taille d'un élément structurant.

3. Problème

Nous avons développé des algorithmes avancés de morphologie mathématique. Ces différentes méthodes ont de multiples utilisations dans de nombreux domaines. Dans le cadre de ce travail, nous chercherons à utiliser ces algorithmes dans l'optique de discerner et compter une quantité d'éléments sur une image en niveaux de gris.

L'image considérée est la photographie de graine suivante :



Nos méthodes définies précédemment étant effectives sur des images binaires, nous commençons par binariser notre image de graine. Par tâtonnement nous déterminons que le seuil permettant de conserver le plus d'information est S=124

Notre hypothèse pour répondre à notre problème est de réaliser une ouverture sur notre image binaire, dans l'optique de lisser les éléments et de supprimer le bruit potentiel. Puis d'appliquer une squelettisation avant d'implémenter un algorithme qui ne garde qu'un seul pixel par élément. Il suffit par la suite de compter le nombre de pixels blancs restants.

Notre méthode de décompte d'élément fonctionne grâce à une boucle qui parcourt tous les pixels éloignés d'au moins 9 pixels des bords de l'image et qui ne conserve que les pixels qui sont éloignés de tout pixels blancs à droite et en bas dans un rayon de 9 pixels.

En comptant à la main les graines entières (qui ne touchent pas les bords), nous trouvons un total de 54 graines.

Après de multiples tentatives d'implémentation, nous remarquons que notre algorithme rend le même résultat avec ou sans l'ouverture. Ainsi le fonctionnement final est le suivant :

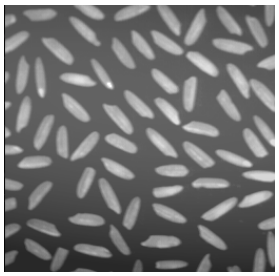
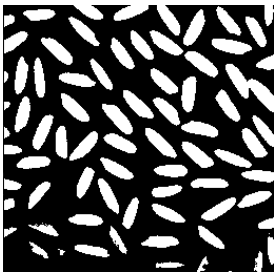
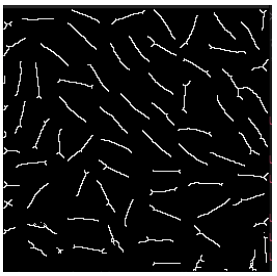


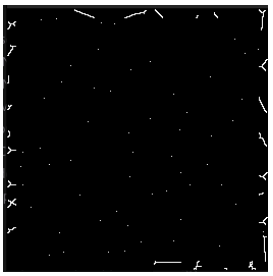
Image originale



Binarisation



Squelettisation



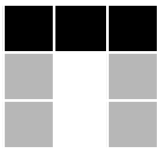
Comptage d'éléments connexes

Une fois la manipulation terminée, l'algorithme retourne 55 graines. Notre algorithme est relativement fragile, et ce pour diverses raisons :

- La première binarisation est déduite par tâtonnement et induit des erreurs (disparition des certaines graines dans la partie sombre de l'image, fusion d'éléments dans la partie claire de l'image)
- Le squelette de certains éléments qui touchent le bord ne touchent plus le bord et risque d'être comptabilisés
- Le décompte des éléments dépend d'un rayon d'écart défini arbitrairement de 9 pixels

Pour toutes ces raisons, nous ne trouvons pas le résultat exact et notre algorithme est spécifique à cette image uniquement, il est donc peu efficace.

En cherchant sur internet nous avons trouvé la méthode idéale qui permet de facilement compter des éléments entiers dans une image. Celle-ci consiste à conserver d'abord uniquement les éléments qui ne touchent pas les bords, puis d'appliquer une squelettisation par amincissement avec le masque suivant :



Le squelette de chaque élément est alors 1 seul pixel, il suffit ensuite de compter le nombre de pixel blanc de l'image rendue.

Conclusion

Ce travail pratique nous a permis de mettre en œuvre différentes techniques de morphologie mathématique. Nous avons implémenté et appliqué un premier algorithme de squelettisation par amincissement, puis 2 autres algorithmes de transformée en distance avant de les appliquer dans un problème de décompte d'éléments connexes. Ce TP a permis d'approfondir notre compréhension des concepts de morphologie mathématique tout en nous confrontant aux défis pratiques liés à leur implémentation et à leur adaptation à des images complexes.