

TP 3 : Segmentation par classification (binarisation)

Introduction

Dans ce TP, nous allons voir deux méthodes pour classer les éléments d'une image par binarisation. La première méthode consiste à séparer les éléments via les niveau de gris des pixels qui les composent. La seconde méthode consiste à les séparer via leur attribut de texture. Nous allons voir les qualités et les défauts de ces deux méthodes et voir si il est possible de traiter toute les images avec une seule. Enfin, nous utiliserons ces deux méthodes combinées pour voir si le résultat est meilleur.

1. Code R

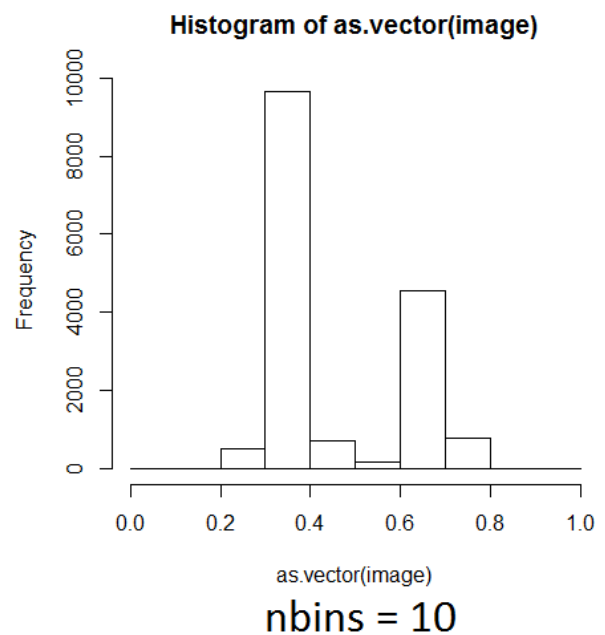
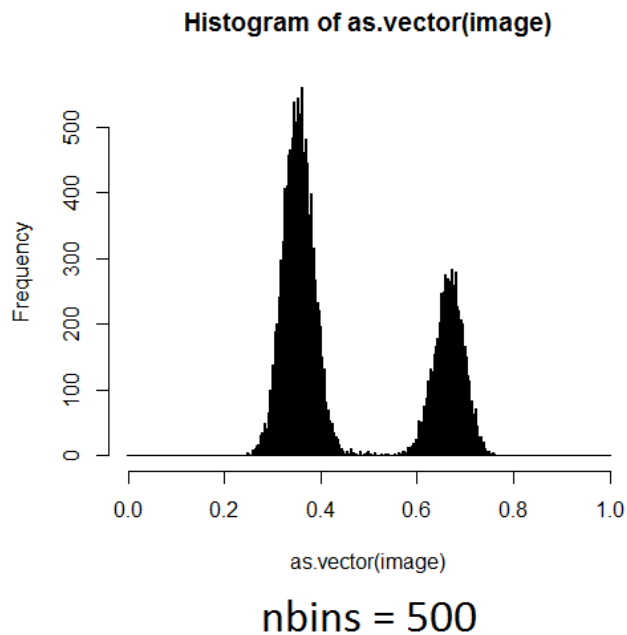
Dans cette partie, nous allons analyser le code R pour comprendre les fonctions à utiliser.

1.1. A quoi correspond l'argument `nbins` utilisé pour calculer l'histogramme des niveaux de gris? Modifier sa valeur (autre que 256) et commenter les résultats.

`bins` permet de définir le pas de l'histogramme, c'est à dire la différence entre chaque valeur de niveau de gris possible de l'histogramme. Donc, plus `bins` est grand, plus $1/\text{bins}$ est petit, plus il y a de valeurs possible et donc plus l'histogramme est précis, c'est à dire, proche de l'image originale. A l'inverse, plus `bins` est petit, plus $1/\text{bins}$ est grand, plus il y aura plus de valeurs qui seront groupés en une seule donc moins l'histogramme sera précis:

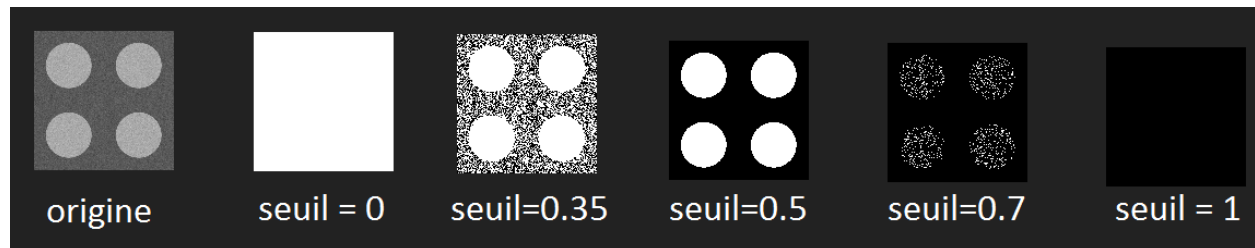
Exemple :

- `bins = 500` $\Rightarrow 1/\text{bins} = 0.002 \Rightarrow$ 500 valeurs possible entre 0 et 1.
- `bins = 10` \Rightarrow 10 valeurs possible \Rightarrow ce qui est entre 0 et 0.1 est regroupé en une seule valeur. On perd de l'information.



1.2. Modifier la valeur de la variable `seuil` et commenter les résultats.

Il faut rappeler que nous sommes en train de faire de la binarisation, c'est à dire faire en sorte de n'avoir que 2 valeurs sur l'image : blanc ou noir. La variable seuil sert à définir la limite entre les deux valeurs. Voici ce qu'il se passe quand on la modifie :



(Rappel) sur une image, plus la valeur d'un pixel est proche de 0 plus sa couleur est proche du noir, plus celle-ci est proche de 1 plus sa couleur est proche du blanc.

Pour être plus précis et comprendre vraiment ce qu'il se passe, la valeur "seuil" définit la limite pour laquelle :

- Tout les points avec une valeur **inférieure** à celle du **seuil** deviendront **noir**.
- Tout les points avec une valeur **supérieure** à celle du **seuil** deviendront **blanc**.

Donc finalement on a :

- Pour seuil = 0, tout les points de l'image sont > 0 donc l'image devient blanche.
- Pour seuil = 1, tout les points de l'image sont < 1 donc l'image devient noire.
- Pour seuil = 0.35, le fond gris foncé ayant des points < 0.35 et > 0.35 , celui-ci devient noir et blanc, les ronds étant nettement > 0.35 ceux-ci deviennent blanc.
- Pour seuil = 0.7, le fond gris foncé ayant des points nettement < 0.7 , celui-ci devient noir, les ronds ayant des points < 0.7 et > 0.7 ceux-ci deviennent noirs et blancs.
- Pour **seuil = 0.5**, tous les points du fond gris foncé sont < 0.5 et tout les points des ronds sont > 0.5 donc le fond devient noir et les ronds deviennent blancs, on a la **binarisation parfaite**.

2. Histogramme des niveaux de gris

Dans cette partie nous allons tester la binarisation grâce au niveau de gris sur plusieurs images pour voir son efficacité et si il est possible de le faire sur toute les images avec une bonne qualité.

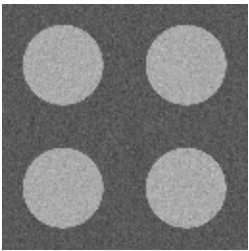
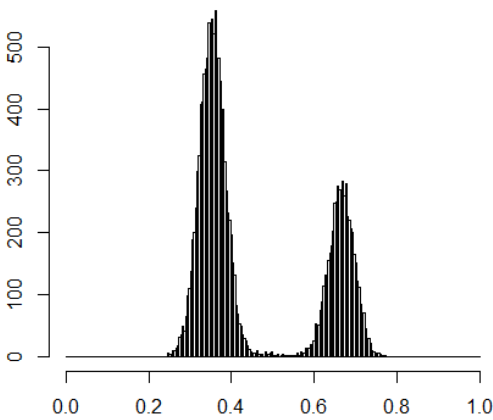
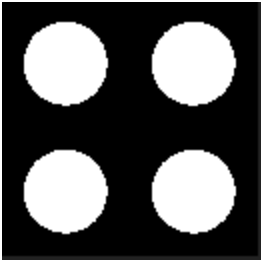
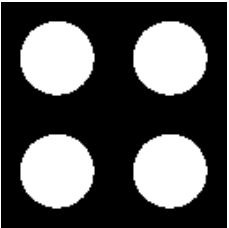
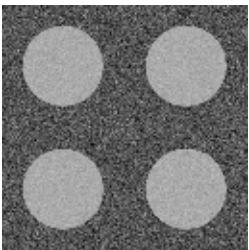
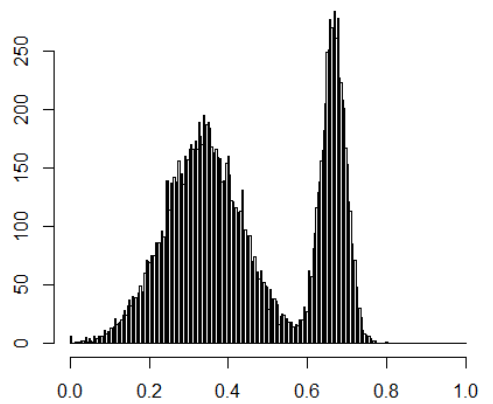
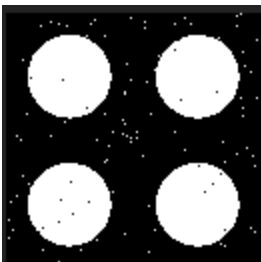
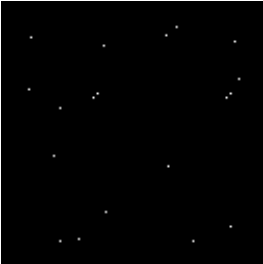
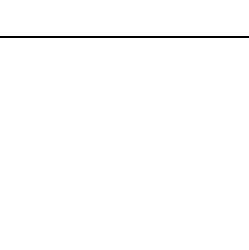


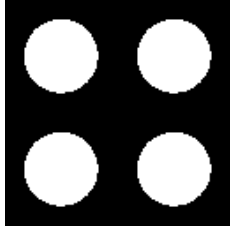
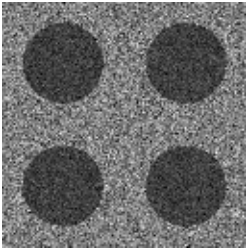
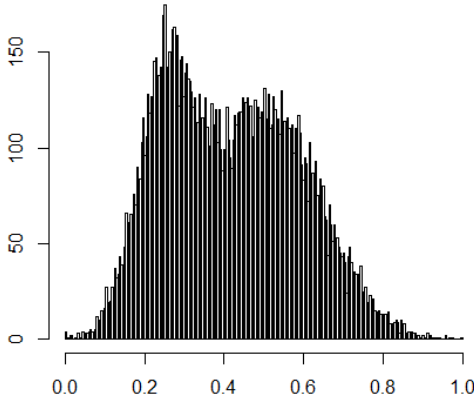
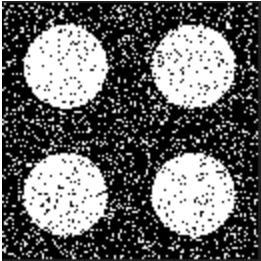
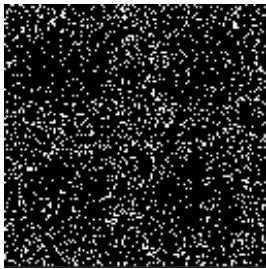
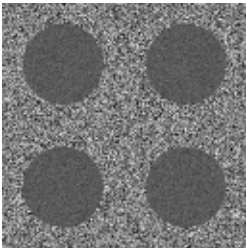
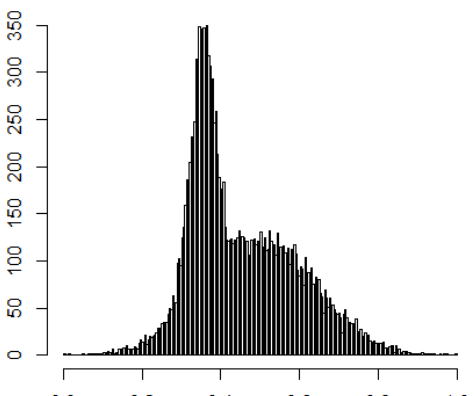
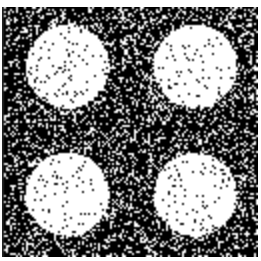

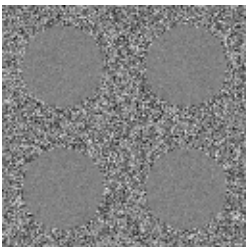
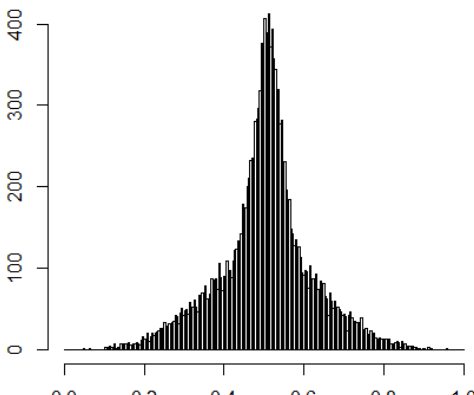
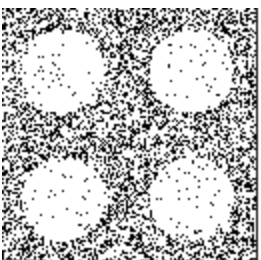
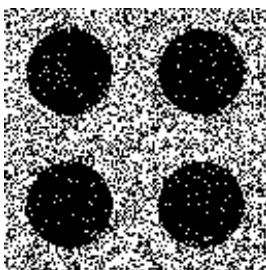
image	histogramme (nbins = 256)	seuil	binarisation	% différence /comparaison
				
		0.5		0.11% 
		0.58		0.88% 

image	histogramme (nbins = 256)	seuil	binarisation	% différence
				
		0.35 (seuil - image car ronds plus foncés que le fond)		12.5% 
		0.4 (seuil - image car ronds plus foncés que le fond)		18% 
		0.45		43.8% 

Conclure sur la possibilité ou pas de binariser toutes les images fournies en associant à chaque pixel son niveau de gris comme **seul attribut** pour la segmentation.

Nous voyons grâce à ces tests que la binarisation avec comme seul attribut le niveau de gris est très efficace sur les images comme la première et la deuxième. En effet la particularité de ces images est que le niveau de gris des pixels du fond est beaucoup plus foncé que le niveau de gris des pixels des ronds.

Par contre, pour les trois autres images, on voit qu'il y a de plus en plus de pixels du fond ayant le même niveau de gris que les pixels des ronds, il est **impossible de les différencier uniquement avec le niveau de gris**. Par conséquent, la qualité de la binarisation est de plus en plus médiocre.

Il est donc **impossible** de **binariser** n'importe quel image avec comme **seul attribut le niveau de gris**.

3. Histogramme des niveaux de texture

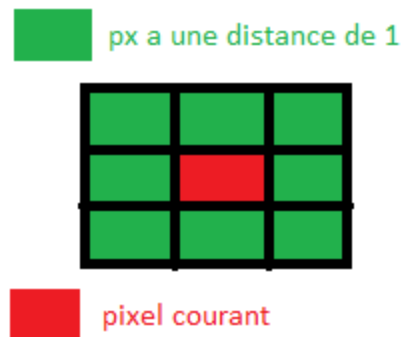
3.1. Expliquer comment la fonction `rdfTextureEcartType` détermine le niveau de texture pour chaque pixel de l'image.

- La fonction `rdfTextureEcartType` calcule d'abord la moyenne pour chaque pixel de l'image grâce à la fonction `rdfMoyenneImage`.
- La fonction `rdfMoyenneImage` a pour but de créer une matrice qui servira de filtre puis l'applique à chacun des pixels de l'image passée en paramètre.
- Ensuite la fonction `rdfTextureEcartType` calcule la variance suivie de l'écart type.

3.2. Comment fixe-t-on la dimension du voisinage de calcul grâce à l'argument taille passé à la fonction?

La taille correspond au nombre de pixels voisins que l'on doit prendre en considération. Grâce à ce chiffre on on calcul la dimension de la matrice pour déterminer l'ensemble de pixel autour du pixel courant.

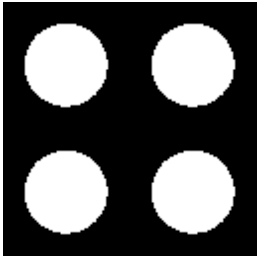
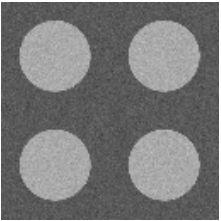
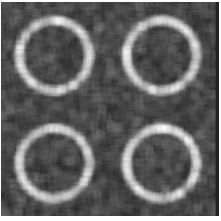
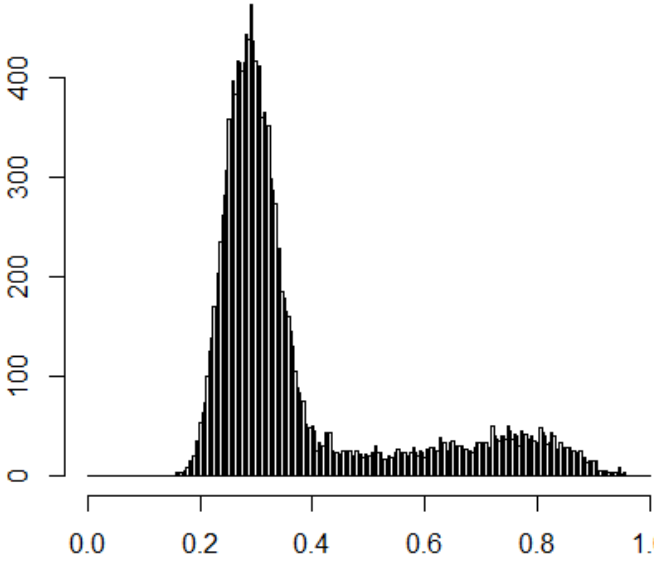

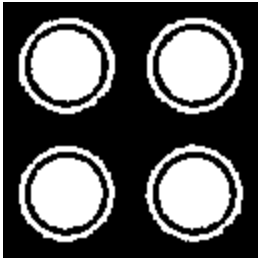
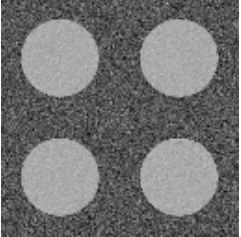
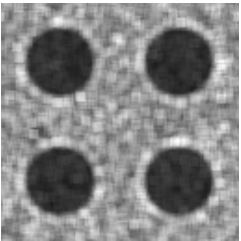
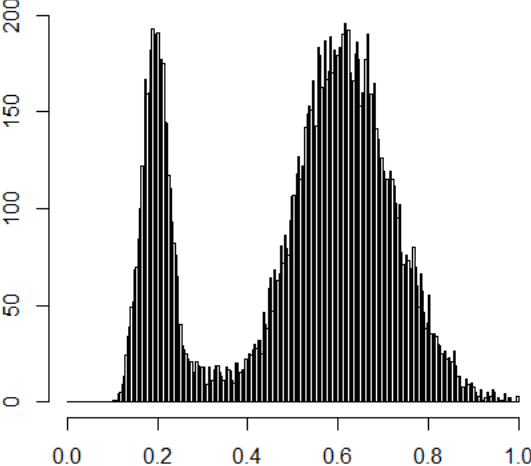
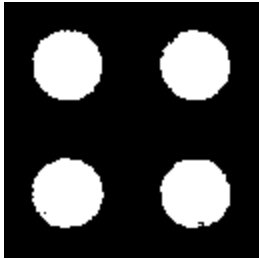
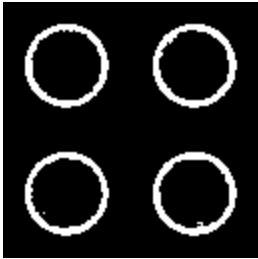
une taille de 1 donne $2*1+1$, on obtient une matrice de dimension $3*3$. Ce schéma montre bien la forme de la matrice $3*3$ pour les pixels a une distance de 1.

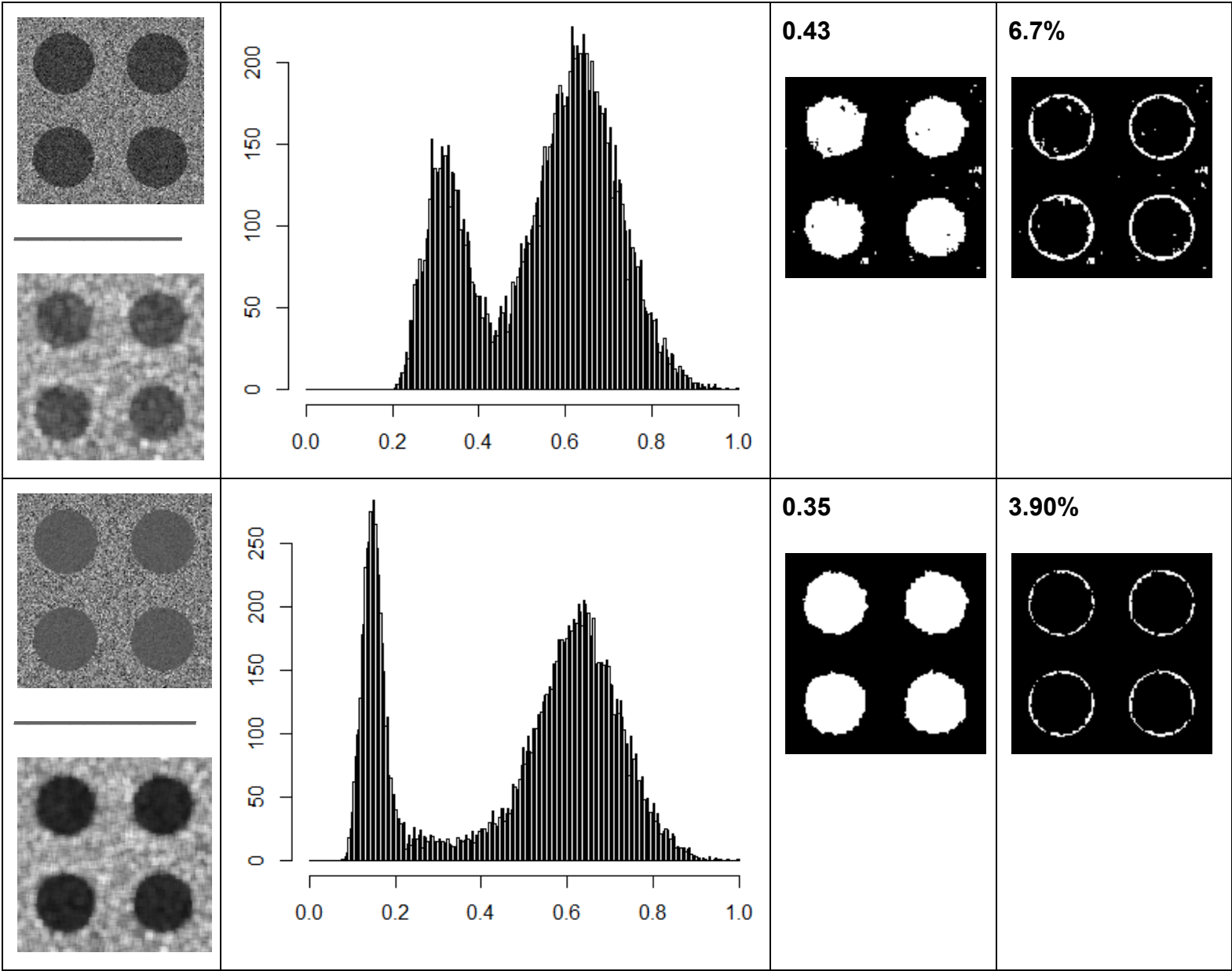


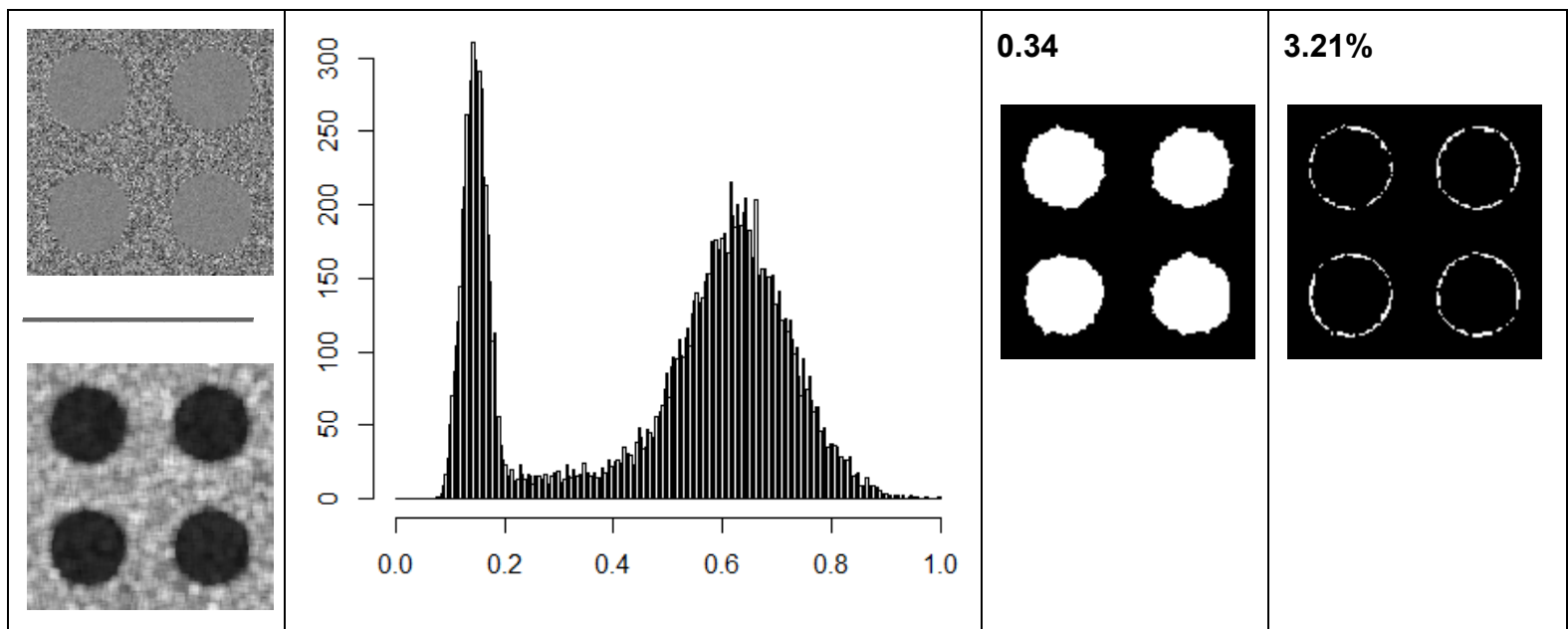
3.3. Pourquoi l'image écart-type est-elle normalisée?

La fonction `rdfTextureEcartType` normalise la matrice puisque le passage au carré modifie grandement les valeurs calculées. En la normalisant, on assure que le résultat retourné possède une fourchette de valeurs similaire à celle de l'image passée en paramètre.

3.4. Pour les 5 images traitées précédemment, déterminer les images de niveau de texture en utilisant un voisinage carré de dimension **5x5** pour chaque pixel. Calculer l'histogramme de chacune de ces images de niveau de texture et en déduire le seuil qui permet de la binariser au mieux. En utilisant l'image de référence, calculer également le pourcentage de pixels mal segmentés dans chaque cas.

<p>image originale</p> <hr/> <p>texture</p>	<p>histogramme de textures (nbins = 256)</p>	<p>seuil / binarisation</p>	<p>% difference</p> <hr/> <p>comparaison avec :</p> 
 <hr/> 		<p>0.46</p> 	<p>34.46%</p> 
 		<p>0.285</p> 	<p>10.18%</p> 





Conclure sur la possibilité de binariser toutes les images fournies en associant à chaque pixel son niveau de texture comme **seul attribut** pour la classification.

Nous voyons grâce à ces tests que la binarisation avec comme seul attribut le niveau de texture est très efficace sur les images comme les deux dernières. En effet la particularité de ces images est que, même si le niveau de gris est proche, le niveau de texture du fond est très différent du niveau de texture des ronds, les ronds sont “lisse” et le fond possède beaucoup de bruit. C’est ce qui fait qu’on passe de 43% d’erreur à 3% d’erreur, ce qui est énorme.

Par contre, pour les trois autres images, on voit qu’il y a du bruit aussi bien sur les pixels que sur les ronds, il est **impossible de les différencier parfaitement uniquement avec le niveau de texture**. Par conséquent, la qualité de la binarisation est beaucoup plus médiocre qu’avec le niveau de gris.

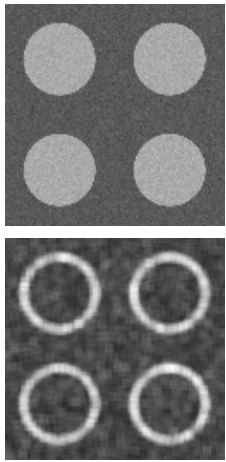
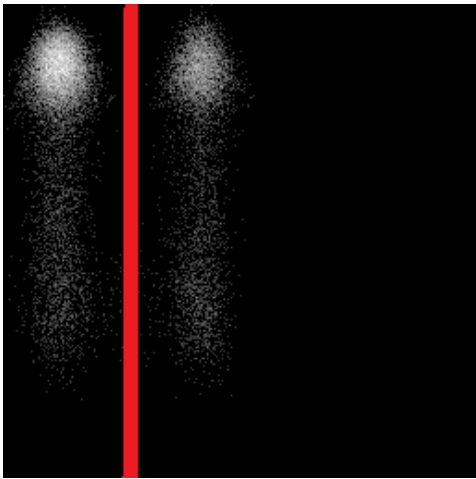
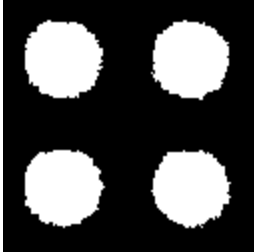

Il est donc **impossible de binariser** n’importe quel image avec comme **seul attribut le niveau de gris**.

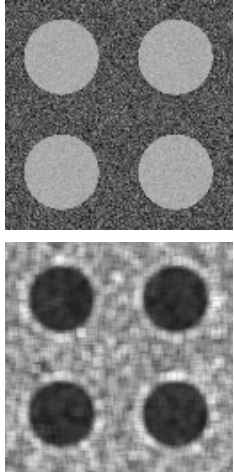
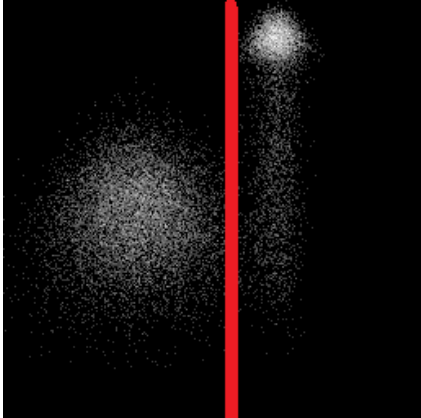
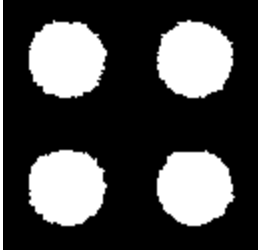
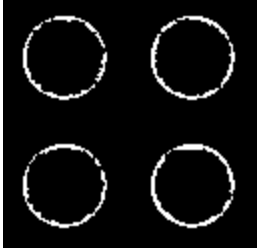
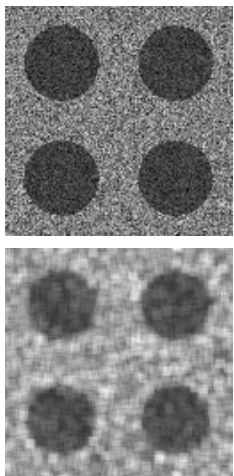
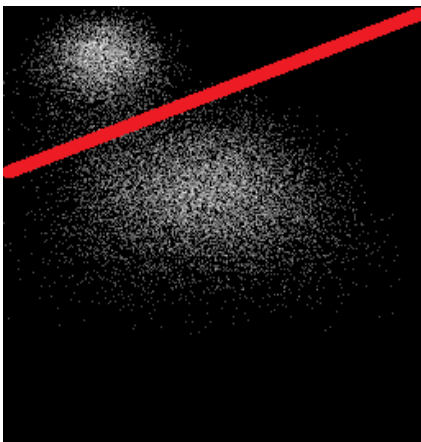
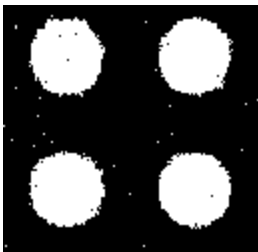
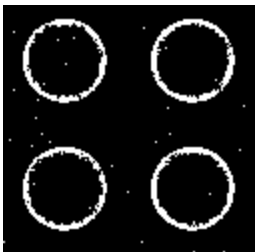
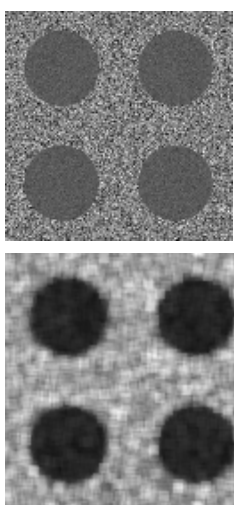
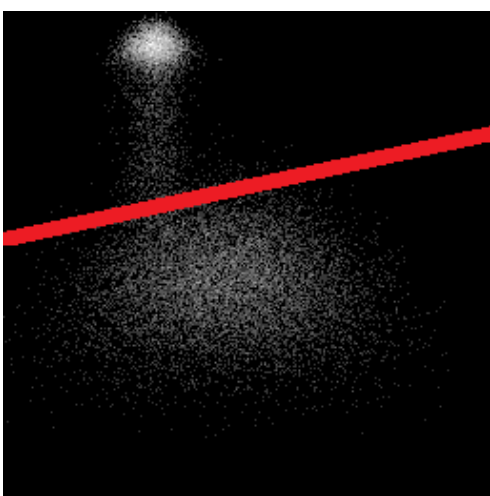

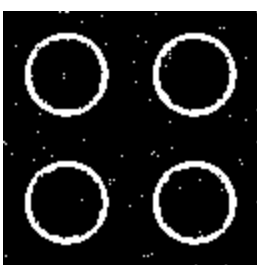
4. Histogramme conjoint

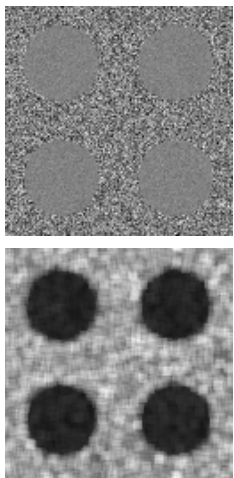
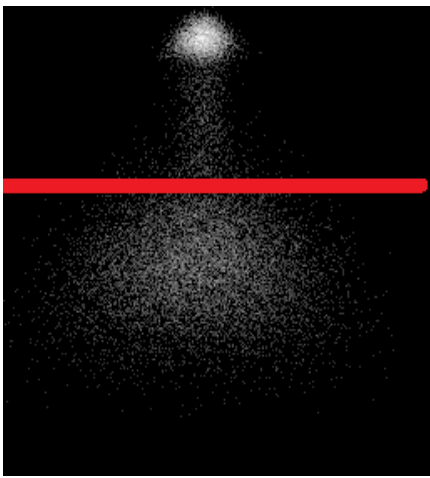
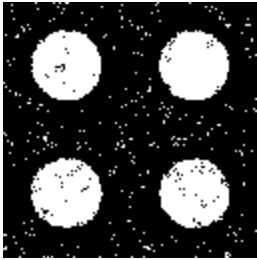
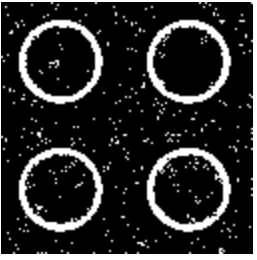
4.1. Analyser le code de la fonction `rdfCalculeHistogramme2D` qui permet de calculer l'**histogramme conjoint** des deux attributs

Cette méthode consiste principalement a récupérer les valeurs des deux images afin de construire les points de l'histogramme vu comme une image 2D.

4.2. Classification linéaire à deux dimensions

image gris / texture	histogramme 2D	Binarisation via classifieur 2D	% erreur
		<p>-0.1</p> 	<p>4.55%</p> 

		<p>-0.139</p> 	<p>5.34%</p> 
		<p>0.65</p> 	<p>8.0%</p> 
		<p>0.68</p> 	<p>9.62%</p> 

		<p>0.79</p> 	<p>12.76%</p> 
--	---	---	--

conclure sur l'intérêt d'utiliser un classifieur 2D sur ce type d'images.

Après avoir combiné les deux méthodes de classification vues précédemment sur les 5 images grâce au classifieur 2D, nous pouvons voir que celui-ci est plutôt performant sur toutes les images.

En effet lorsque nous utilisons qu'une seule méthode (niveau de gris / niveau de texture), celle-ci était très efficace mais sur un seul type d'images, celles dont les ronds ont un niveau de gris très différent de celui du fond, et celles dont le niveau de texture des ronds est très différent du niveau de texture du fond.

Avec le classifieur 2D, nous pouvons classifier les éléments (rond et fond) sur n'importe quel type d'image, peu importe le niveau de gris et le niveau de texture, avec une qualité convenable, mais l'efficacité sur les extrêmes n'est pas autant au rendez-vous qu'avec les méthodes seules. En effet, sur l'image 1, lorsqu'on hérite de la qualité du niveau de gris, on hérite aussi de la médiocrité du niveau de texture, et sur l'image 5, lorsqu'on hérite de la qualité du niveau de texture, on hérite des problèmes du niveau de gris.

Tout ces éléments font de la méthode du classifieur 2D une méthode plutôt performante dans l'ensemble avec quelques petites erreurs.

Conclusion

Les 2 démarches vues lors de ce TP pour classifier les éléments d'une image sont intéressantes. La méthode du niveau de gris est très utile pour des images dont le niveau de gris des différents éléments est bien distinct. Cependant pour les images dont les niveaux de gris sont proches la marge d'erreur est énorme.

Par contre la méthode des niveaux de textures est très efficace sur les images dont la texture des éléments distincts est différente cependant pour les images dont les textures sont proches la marge d'erreur est énorme.

C'est pourquoi on ne peut pas classifier n'importe quelles images avec un seul attribut qu'il soit le niveau de gris ou la texture, il faut combiner ces deux attributs et binariser via le classifieur 2D pour avoir un résultat convenable sur n'importe quelle image.