

Rendu tp: segmentation par binarisation

Durant ce TP, nous allons aborder différentes manière de segmenter une image par binarisation. La première méthode étudiée va être la classification des pixels d'une image par l'analyse de son histogramme afin de reconnaître ou non l'objet du fond. On utilisera le niveau de gris d'un pixel comme attribut pour la classification. Ensuite la deuxième méthode sera la segmentation par l'étude de l'histogramme des niveaux de texture d'une image: on va utiliser le niveau de texture associé à chaque pixels.

Avant toute chose, nous allons manipuler les histogrammes d'une image et certaines fonctions permettant de segmenter une image avec le seuil.

Code R

Nous allons analyser les deux fichiers « `rdfSegmentation.R` » et « `rdfTestSegmentations.R` ».

Le fichier « `rdfSegmentation.R` » contient les fonctions permettant d'extraire les attributs de pixels pour la classification d'une image. La fonction « `rdfReadGreyImage` » permet de charger une image en niveau de gris (la même fonction que dans les précédents TP). Le fichier contient trois autres fonctions « `rdfMoyenneImage` », calculant la moyenne d'une image en fonction de la taille du rayon du carré des voisins de l'image, « `rdfTextureEcartType` » permet de calculer l'écart type normalisé à partir des voisins carré d'une image et enfin `rdfCalculeHistogramme2D` calcul l'histogramme 2D de deux images.

Le fichier « `rdfTestSegmentations.R` » correspond à l'exécution des fonctions du fichier précédent. C'est ici que l'on charge un fichier puis qu'on affiche la segmentation par binarisation.

L'argument « `nbins` » utilisé pour calculer l'histogramme des niveaux de gris correspond au nombre de niveau de gris pris en compte dans cet histogramme. Plus on augmente ce nombre, plus la précision de l'histogramme sera élevé.

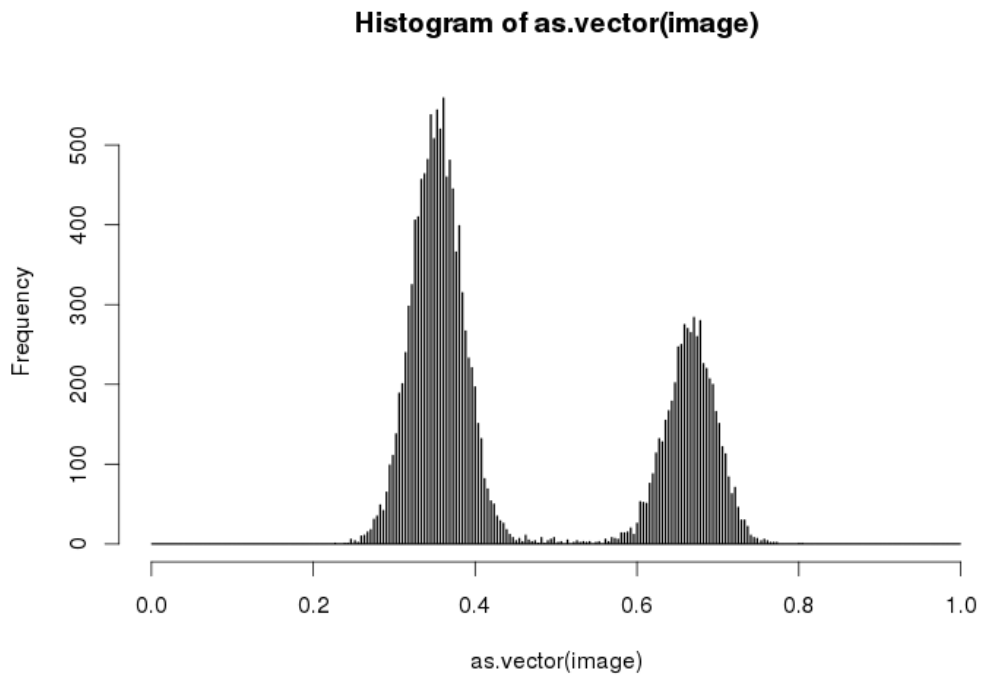


Illustration 1: Histogramme de l'image "2 classes texture-0" avec nbins = 1024

Au contraire, en utilisant une valeur très faible, notre histogramme ne pourra pas être utilisé car il ne sera pas assez précis. Il nous indiquera vaguement la proportion de pixel clairs ou foncés.

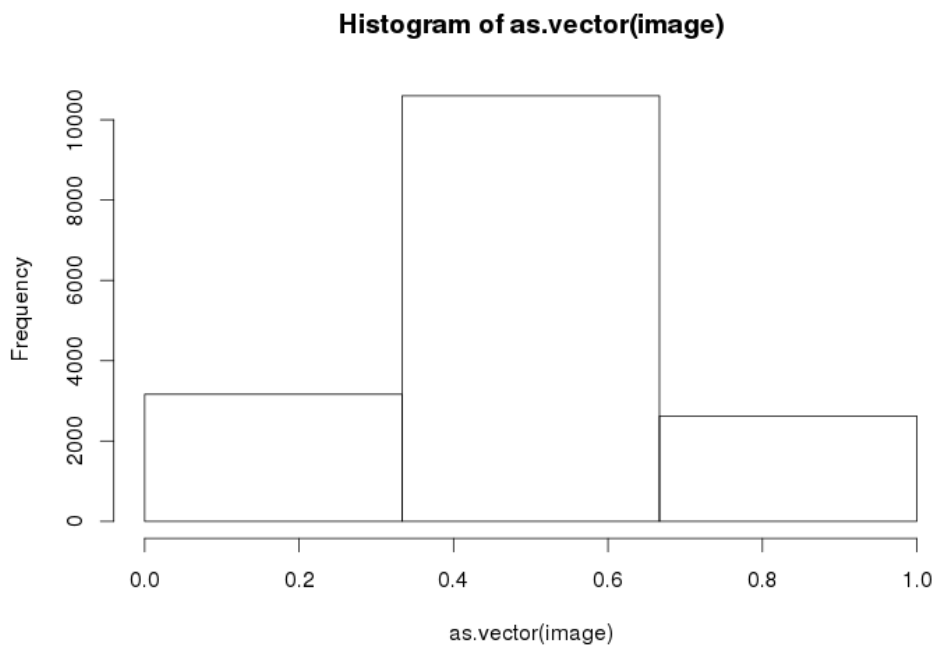


Illustration 2: Histogramme de l'image "2 classes texture-0" avec nbins = 3

L'argument seuil correspond au seuil de binarisation de l'image. Cet argument permet de transformer une image en niveau de gris, en image en noir et blanc. Ce seuil fonctionne de la manière suivante : les pixels dont le niveau de gris est supérieur au seuil sont mis à 1 et ceux inférieur à 0. Si le seuil est à 0, nous obtiendrons une image toute blanche. Par contre, si il est à 1, l'image obtenu sera noire. Dans notre cas, le seuil idéal est de 0,5 car le creux entre les deux pics de l'histogramme est situé au alentour de 0,5.

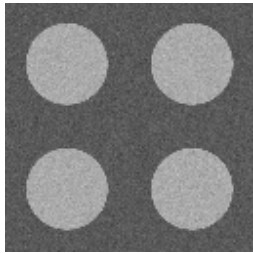


Illustration 3: l'image "2 classes texture-0" sans binarisation

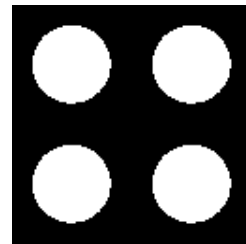


Illustration 4: Binarisation de l'image "2 classes texture-0" avec un seuil de 0,5

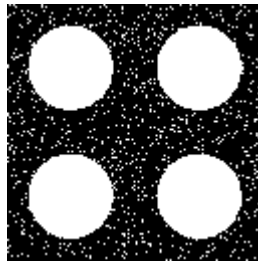
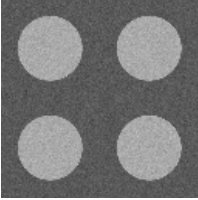
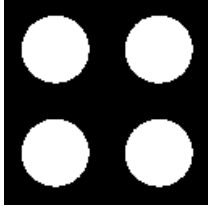
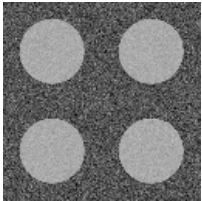
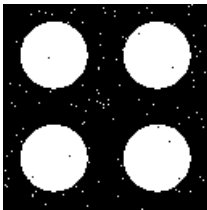
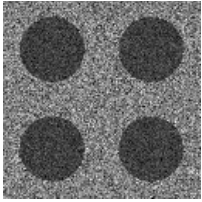
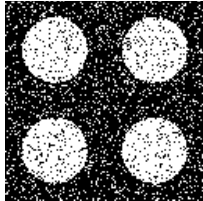
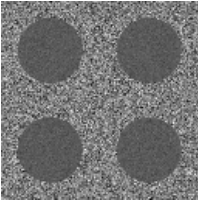
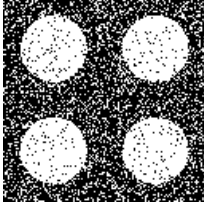


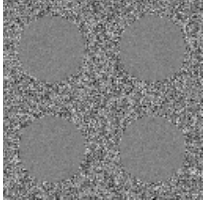
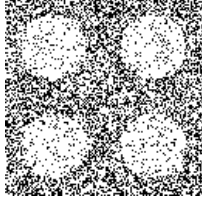
Illustration 5: Binarisation de l'image "2 classes texture-0" avec un seuil de 0,4

L'illustration 5 ci-dessus montre bien qu'avec un seuil trop bas, on obtient du bruit sur la segmentation de l'image.

Maintenant que nous avons vu comment fonctionne la binarisation d'une image en fonction du niveau de gris des pixels, nous allons tester cette méthode sur différentes images pour voir si elle est fiable.

Histogramme des niveaux de gris

 <p><i>Illustration 3: image texture 0</i></p>	 <p><i>Illustration 5: image binarisé avec un seuil de 0,5</i></p>	<div>0.12 % d'erreur</div> <p>avec un seuil de 0.5</p>
 <p><i>Illustration 6: image texture 1</i></p>	 <p><i>Illustration 4: image binarisé avec un seuil de 0,58</i></p>	<div>0.88 % d'erreur</div> <p>avec un seuil de 0.58</p>
 <p><i>Illustration 7: image texture 2</i></p>	 <p><i>Illustration 8: image binarisé avec un seuil de 0,34</i></p>	<div>12.32 % d'erreur</div> <p>avec un seuil de 0.34</p>
 <p><i>Illustration 10: image texture 3</i></p>	 <p><i>Illustration 9: image binarisé avec un seuil de 0,4</i></p>	<div>18,27 % d'erreur</div> <p>avec un seuil de 0.40</p>

 <p><i>Illustration 12: image texture 4</i></p>	 <p><i>Illustration 11: image binarisé avec un seuil de 0,47</i></p>	<div data-bbox="1019 159 1436 197" style="background-color: #cccccc; padding: 2px;">42,51 % d'erreur</div> <p>avec un seuil de 0.47</p>
--	---	---

Grâce à ces résultats, nous pouvons constater que toutes les images ne peuvent pas être binarisées en utilisant uniquement le seuil de binarisation. Pour l'image n°4, on voit très clairement que la binarisation avec le seuil n'est pas fiable. En effet, comme on le voit sur l'histogramme ci-dessous, les points du cercle et ceux du fond ont un niveau de gris équivalent, la binarisation fait apparaître des points un peu partout. Il faut donc trouver d'autres méthodes pour binariser au mieux une image.

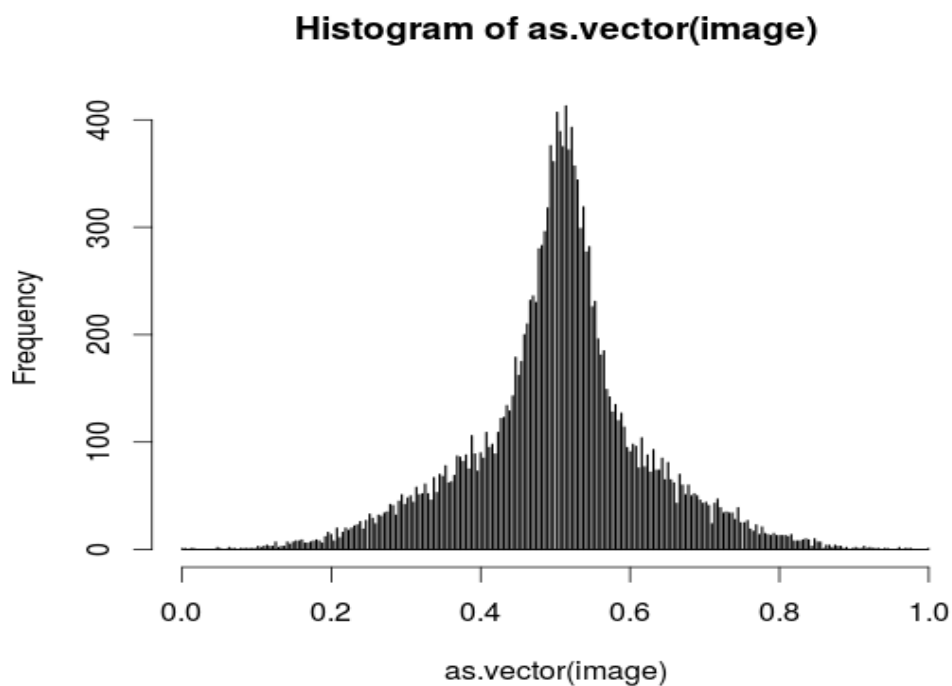


Illustration 13: Histogramme du niveau de gris de la texture n°4

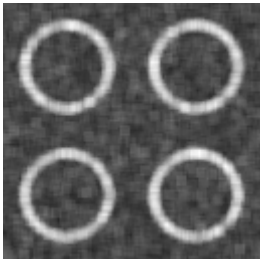
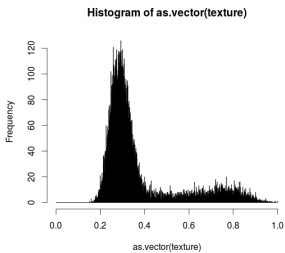

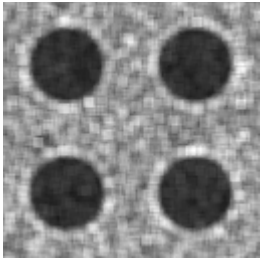
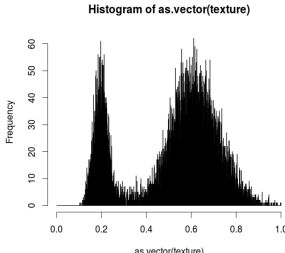
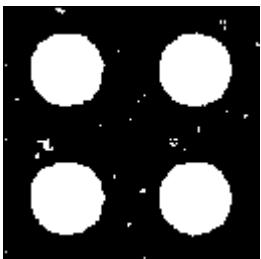
Maintenant que l'on sait que la binarisation grâce au niveau de gris des pixels n'est pas suffisante pour toutes les images, nous allons tester avec les niveaux de textures pour voir si nous allons avoir un meilleur rendu.

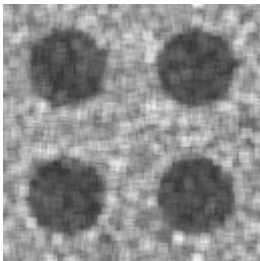
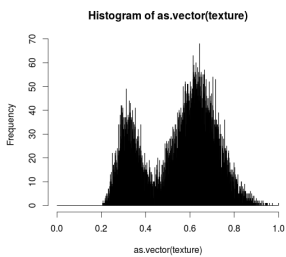

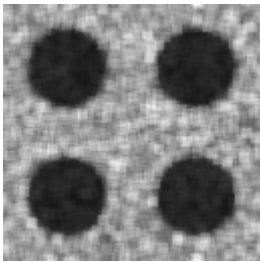
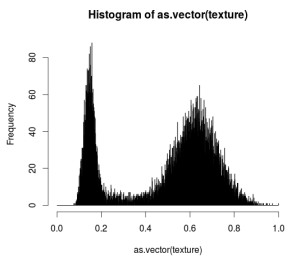

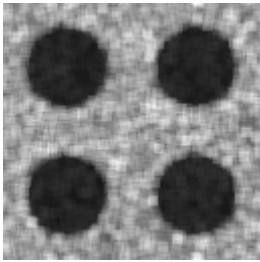
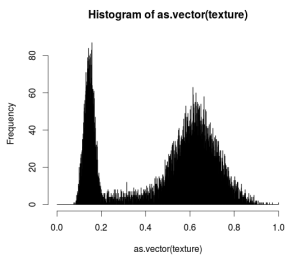

Histogramme des niveaux de texture

La fonction « `rdfTextureEcartType` » détermine le niveau de texture pour chaque pixel d'une image. Pour ce faire, La fonction calcule, pour chaque pixel, l'écart type entre ce pixel et les voisins dans un rayon de « `taille` » pixels. L'écart type est calculé en faisant la moyenne des pixels concernés, Une fois la moyenne obtenue, on calcule le carré de l'image moins sa moyenne pour obtenir la variance de l'image. Ensuite, il ne reste plus qu'à faire la racine carré de la variance pour obtenir l'écart type du voisinage de pixels. Pour déterminer le voisinage d'un pixel en fonction de la taille du rayon voulu, on crée un tableau de taille « `taille*2 + 1` », ainsi pour un rayon de 2 pixels, le tableau sera de taille 5x5, ce qui correspond au pixel et ses voisins situés à une distance de 2.

`Filter2` permet de filtrer une image en utilisant la transformation de Fourier. `Filter2` peut être utile pour enlever le bruit sur une image ou pour détecter les contours. On va alors passer dans ce tp l'image et le masque calculé en fonction de la taille. C'est le filtre de convolution.

Ensuite, pour utiliser l'écart type afin d'obtenir une binarisation de l'image, il ne reste qu'à le normaliser pour obtenir une image en niveau de gris que nous pourrions binariser comme dans la partie précédente. Cette normalisation s'obtient en divisant chaque écart type par l'écart type le plus grand. On obtiendra ainsi uniquement des valeurs situées entre 0 et 1 ce qui correspond à une image en niveau de gris.

			34,45 % d'erreur avec un seuil de 0.45
			8,40 % d'erreur avec un seuil de 0.40

			6,27 % d'erreur avec un seuil de 0.46
			3,09 % d'erreur avec un seuil de 0.40
			2,47 % d'erreur avec un seuil de 0.41

Au vu de ces résultats, on constate que toutes les images ne peuvent pas être binarisées en utilisant uniquement l'histogramme des niveaux de texture. On le voit d'une part car les formes ne sont pas très précises et on constate du bruit dans la binarisation. Et d'autre part car, comme on le voit avec l'image 0, la binarisation s'effectue parfois uniquement sur le contour et non pas sur toutes la forme.

Nous possédons deux méthodes de binarisation qui ne fonctionne pas sur toutes les images. Nous allons donc, dans la partie suivante, essayer de les combiner pour voir si l'on obtient un meilleur résultat.

Histogramme conjoint et classifieur 2D

Pour essayer d'obtenir une meilleur segmentation de notre image, nous allons créer un histogramme conjoint aux deux attributs utilisés précédemment. Pour ce faire, la méthode « `rdfCalculeHistogramme2D` » va calculer le tableau de contingence entre le tableau des niveaux de gris et celui des niveaux de texture. On applique tout d'abord la fonction « `findInterval(images1, Seq(0,1,1/nbins1))` » pour mettre les deux tableaux sous la même forme. Ensuite, cette méthode va utiliser la fonction logarithme pour faire apparaître les zones de faible surface. Finalement, cette

fonction va normaliser les valeurs pour pouvoir les afficher dans une nouvelle image.

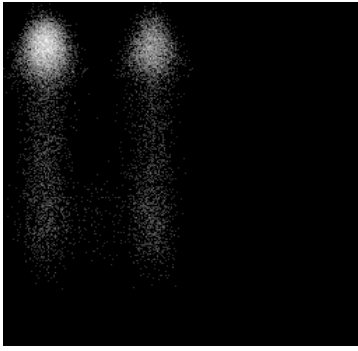


Illustration 14: Histogramme conjoint de la texture 0

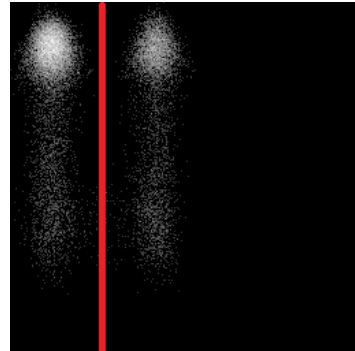


Illustration 15: Séparation de l'histogramme conjoint de la texture 0

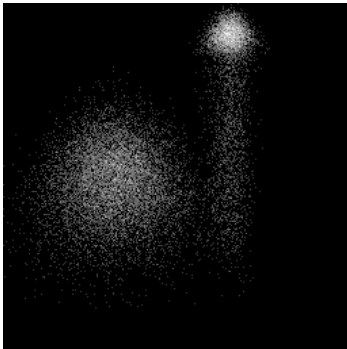


Illustration 16: Histogramme conjoint de la texture 1

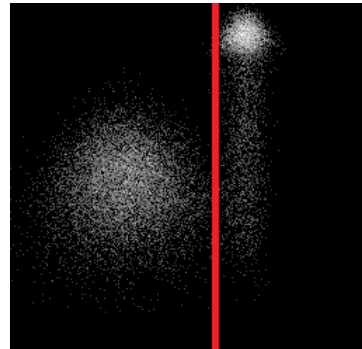


Illustration 17: Séparation de l'histogramme conjoint de la texture 1

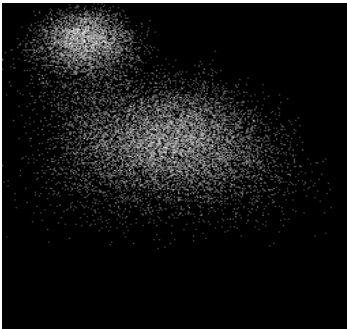


Illustration 18: Histogramme conjoint de la texture 2

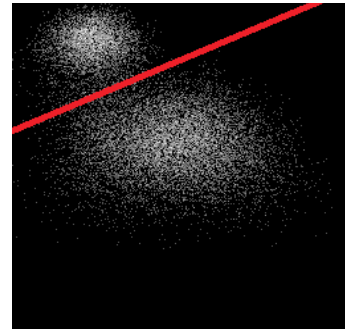


Illustration 19: Séparation de l'histogramme conjoint de la texture 2

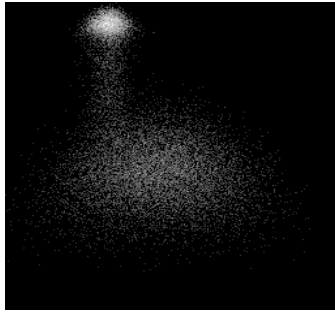


Illustration 20: Histogramme conjoint de la texture 3

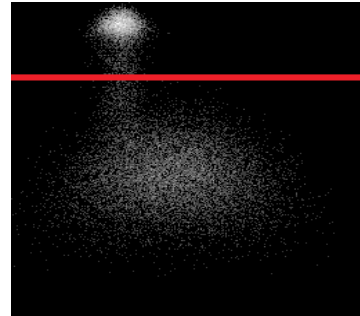


Illustration 21: Séparation de l'histogramme conjoint de la texture 3

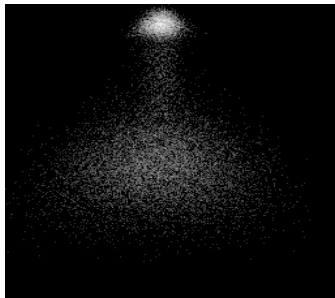


Illustration 22: Histogramme conjoint de la texture 4

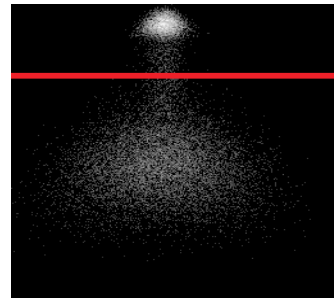


Illustration 23: Séparation de l'histogramme conjoint de la texture 4

Dans le tableau ci-dessus, on voit que la séparation du fond et de la forme est de trois types, une séparation verticale, une horizontale et une quelconque.

On constate que lorsque la séparation est verticale, la meilleur binarisation obtenue l'est par les niveaux de gris. Lorsque cette séparation est horizontale, on obtient cette fois-ci une binarisation optimale avec les niveaux de textures. Enfin, quand la séparation est ni horizontale, ni verticale, une association des deux attributs permet une meilleur binarisation.

Conclusion

Nous avons vu dans ce TP deux méthodes différentes pour binariser une image afin de séparer une forme du fond. Les deux méthodes vues, celle utilisant les niveaux de gris et l'autre utilisant les niveaux de textures, obtiennent des résultats différents en fonction des images. Par exemple, pour la texture 0, la méthode avec les niveaux de gris obtient un résultat excellent alors que la binarisation avec les niveaux de textures est mauvaise. Nous avons également vu qu'il est possible de combiner plusieurs méthodes pour obtenir une meilleur binarisation.