

AIV1 - TP07  
Segmentation par classification d'attributs

Suliac Lavenant et Antoine Nollet

4 March 2022

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Histogramme des niveaux de gris</b>	<b>3</b>
<b>3</b>	<b>Histogramme des variances locales</b>	<b>7</b>
<b>4</b>	<b>Histogramme conjoint et classifieur 2D</b>	<b>11</b>
4.1	Histogramme conjoint . . . . .	11
4.2	Classifieur 2D . . . . .	13

# 1 Introduction

Lors de ces travaux de traitements d'images, notre objectif consistera à segmenter nos images de manière à reconnaître correctement son fond et ses formes. Pour cela, il choisira quel(s) attribut(s) prendre en compte et comment les classifier. Nos classifications seront ici des binarisations, car il n'y a que 2 classes possibles, les pixels de l'image appartiennent soit au fond soit à une forme.

La binarisation consiste donc à affecter à chaque pixel une étiquette indiquant s'il appartient à un objet (True) ou au fond (False).

Pour le code, considérons un tableau numpy, on applique l'opération  $b0 + b1 * p$  à chaque valeur d'attribut  $p$  du tableau ( par exemple niveau de gris des pixels de l'image).

```
1 def binarization( im1, b0, b1 ):
2     return ( b0 + b1 * im1 >= 0 )
```

La valeur retournée sera un tableau deux dimensions, de même dimension que l'image initiale, dont chaque valeur à la position (x,y) sera affecté au booléen correspondant à si la valeur du pixel  $I(x,y)$  multipliée par un nombre  $b1$  et additionnée par un nombre  $b0$  est supérieure ou égale à 0.

Autrement dit, le tableau deux dimensions retournée correspondra aux étiquettes des pixels : valeur à True si appartenant à un objet (selon définition de nos  $b0$  et  $b1$ ), valeur à False sinon.

La binarisation est donc défini selon une valeur attribuée par pixels. Mais il faut choisir quel attribut à prendre en compte pour affecter ces valeurs de pixels. Dans un premier temps, la lecture consistera à ouvrir l'image et à la transformer en tableau numpy puis si elle n'est pas d'une seule composante récupère la première en tant qu'image. Ainsi, nous aurons une image dont les valeurs des pixels sont des nuances de gris.

```
1 name = 'aiv1-2-classes-texture-0'
2 image = plt.imread( 'images/' + name + '.png' )
3 if image.ndim > 2:
4     image = image[:, :, 0]
```

## 2 Histogramme des niveaux de gris

Nous considérons donc les niveaux de gris de l'image pour effectuer notre binarisation. Ainsi, nous utilisons les histogrammes des niveaux de gris de nos images afin de trouver les seuils auxquelles déterminer si le pixel appartient au fond ou à une forme.

Voici quelques exemples d'images avec leurs histogrammes de niveaux de gris :

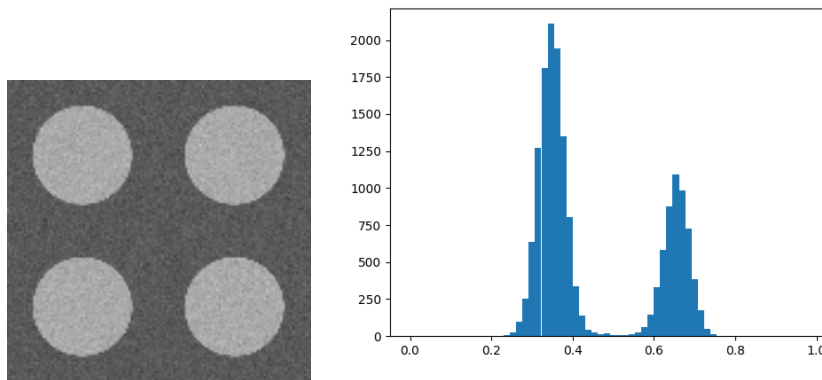


FIGURE 1 – Exemple 0 et son histogramme

En constatant l'histogramme, on choisira un seuil à 0.5 (correspondant au creux) afin de dissocier les pixels de fonds des pixels de formes. Nous obtenons ces résultats :

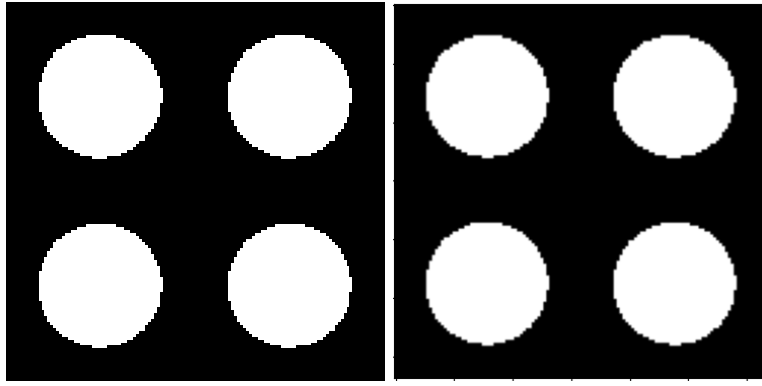


FIGURE 2 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 99.88%. La ressemblance visuelle de ces deux binarisation est très ressemblante et cette ressemblance est confirmée par ce pourcentage de 99.88%.

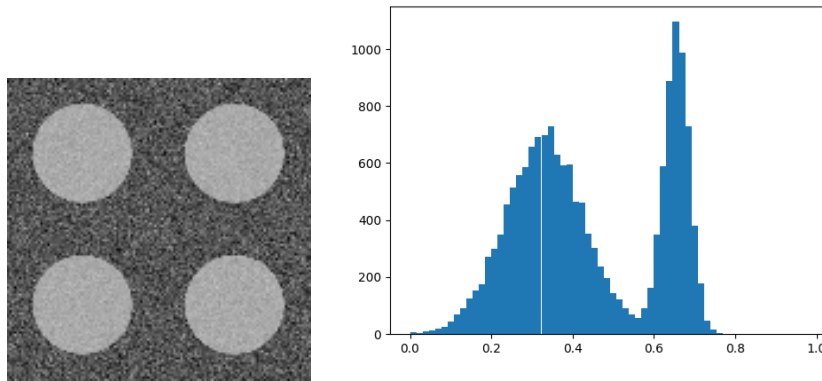


FIGURE 3 – Exemple 1 et son histogramme

En constatant l'histogramme, on choisira un seuil à 0.575 (correspondant au creux) afin de dissocier les pixels de fonds des pixels de formes. Nous obtenons ces résultats :

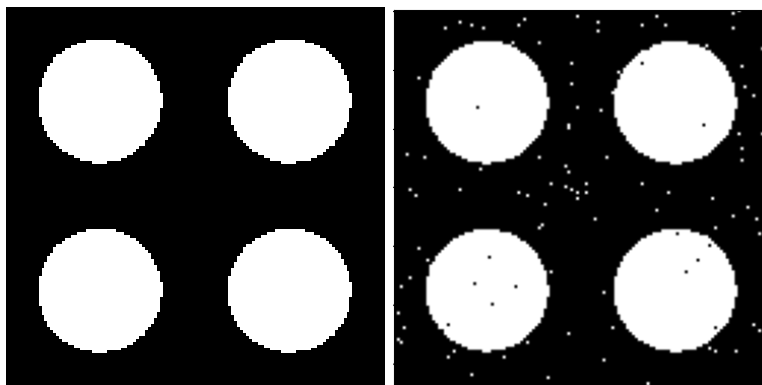


FIGURE 4 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 99.09 %. Bien que le pourcentage reste très élevé, on constate assez vite les pixels parasites à l'oeil nu.

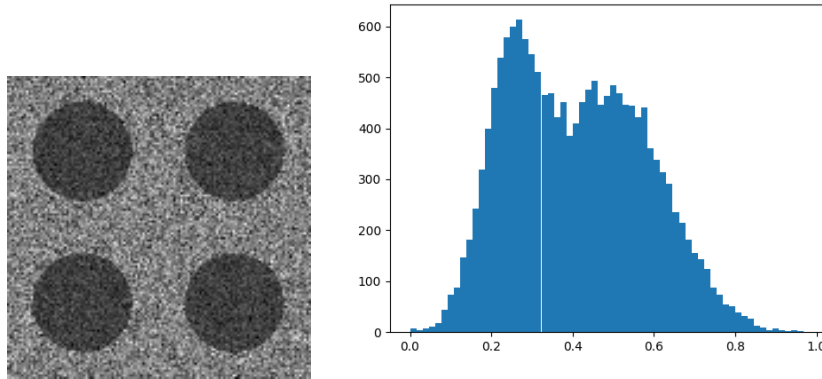


FIGURE 5 – Exemple 2 et son histogramme

En constatant l'histogramme, on choisira un seuil à 0.375 afin de dissocier les pixels de fonds des pixels de formes. Nous obtenons ces résultats :

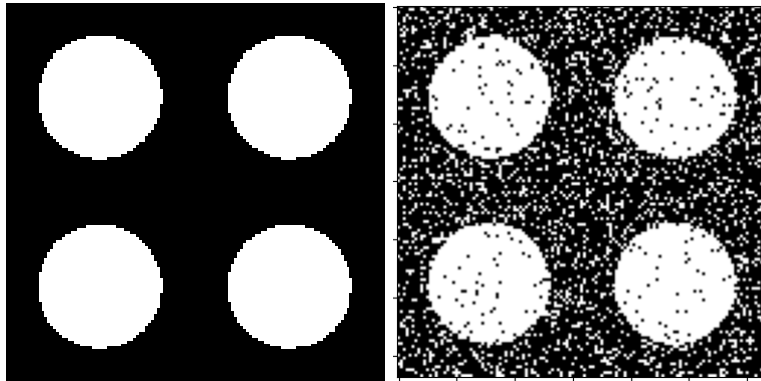


FIGURE 6 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 85.95 %. L'image commence à être parasitée de manière significative, on retrouve un nombre important de pixels noirs dans les formes.

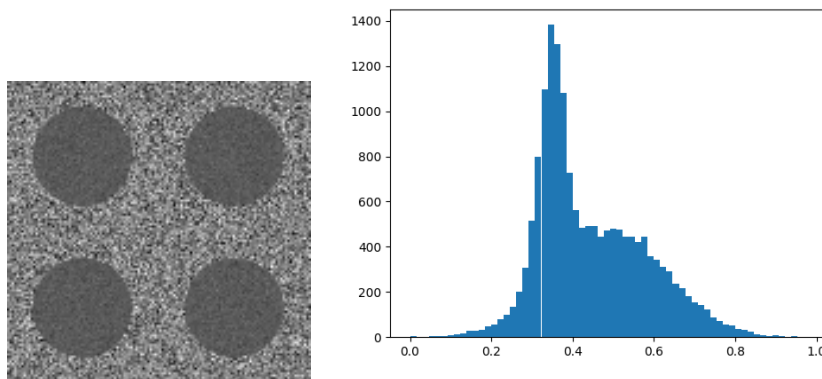


FIGURE 7 – Exemple 3 et son histogramme

En constatant l'histogramme, on choisira un seuil à 0.4 afin de dissocier les pixels de fonds des pixels de formes. Nous obtenons ces résultats :

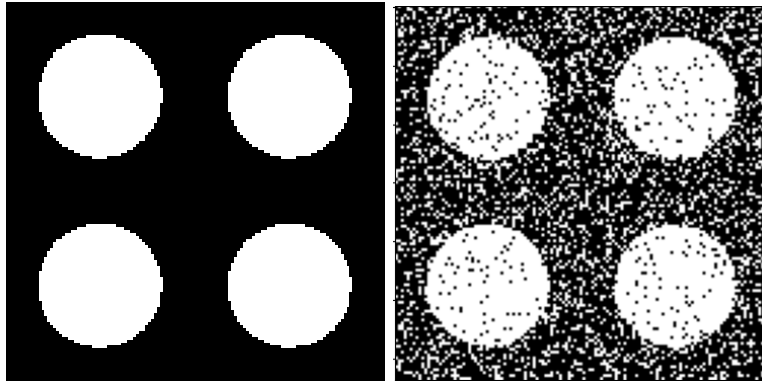


FIGURE 8 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 81.72 %. L'image ici ressemble fortement au cas précédent, la binarisation n'est pas robuste.

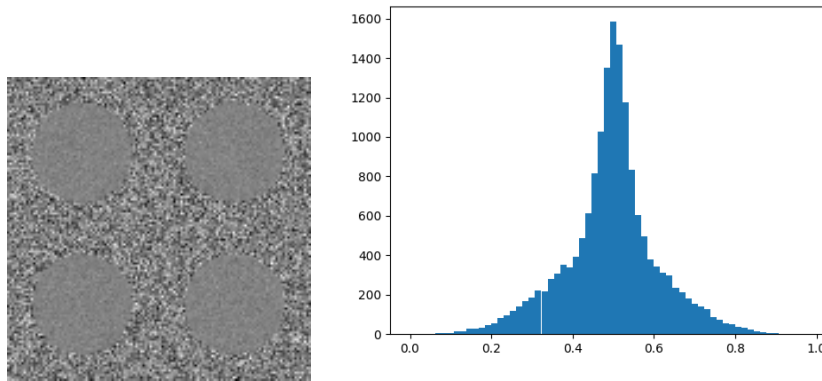


FIGURE 9 – Exemple 4 et son histogramme

En constatant l'histogramme, on choisira un seuil à 0.5 afin de dissocier les pixels de fonds des pixels de formes. Nous obtenons ces résultats :

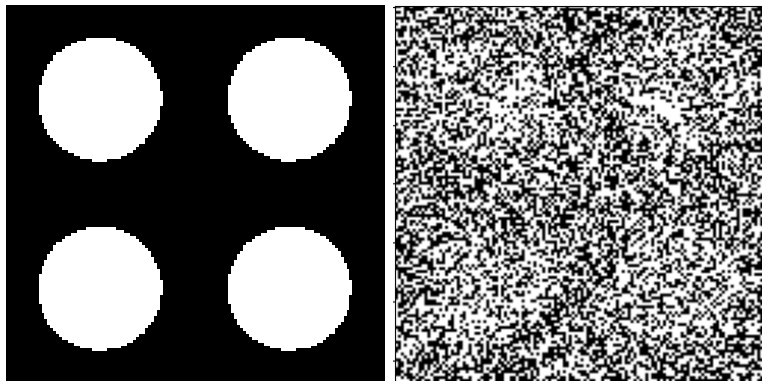


FIGURE 10 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 54.10 %. On ne distingue même plus les 4 formes dans l'image binarisée, ce qui la rend inexploitable.

Pour conclure sur la binarisation par seuil sur les niveaux de gris des pixels, on peut affirmer que l'efficacité de cette méthode dépend de l'image de base utilisée. Plus elle va être dégradée plus la binarisation obtenue ne sera pas d'une qualité acceptable.

L'efficacité de cet attribut (le niveau de gris des pixels) pour segmenter par binarisation le fond et les formes d'une image est donc très dépendant selon les cas, il nous faudrait en utiliser un autre...

### 3 Histogramme des variances locales

Pour espérer effectuer une meilleure segmentation, nous allons toujours utiliser une classification par binarisation (2 classes, soit fond soit forme) mais un nouvel attribut : le niveau de texture des pixels. Le voisinage carré de chaque pixel sera utilisé pour calculer la variance des niveaux de gris aux alentours du pixel auquel on calcul le niveau de texture.

Voici comment sera calculé la valeur  $V_{(x,y)}$  de texture d'un pixel situé en (x,y) :

$$V_{(x,y)} = |I_{(x,y)} - C_{(x,y)}|$$
$$C_{(x,y)} = \frac{1}{(2 * width + 1)^2} \sum_{i=x-width}^{x+width} \sum_{j=y-width}^{y+width} I_{(i,j)}$$

Ici la fonction  $I_{(x,y)}$  retourne la valeur de nuance de gris au pixel (x,y), la fonction  $C_{(x,y)}$  retourne la valeur de la moyenne des niveaux de gris des pixels se trouvant au plus à  $width$  pixels de distance du pixel (x,y). Ainsi, la valeur  $V_{(x,y)}$  de texture correspondra à la variance de niveau de gris au sein du voisinage carré du pixel (x,y).

Voici comment nous allons récupérer cette valeur via algorithme :

```
1 def mask( width ):  
2     width = 2 * width + 1  
3     return np.ones(( width, width )) / width / width
```

Cette fonction mask crée un mask de dimension  $width*2+1$  par  $width*2+1$  possédant la même valeur à chaque endroit, cette valeur est égale à  $1/(width*2+1)^2$ . Ici width prend la valeur  $2 * width + 1$  car le premier width correspond à la distance du voisinage au pixel central, alors que le nouveau correspond à la dimension de ce voisinage carré (width pixels à gauche et width pixels à droite, plus le pixel central). Et la valeur  $1/(width*2+1)^2$  est affectée à chaque "case" du mask car cela correspond à l'inverse du nombre total de "cases" du mask.

```
1 def average( im, width ):  
2     return sig.convolve2d( im, mask( width ), mode='same' )
```

Applique la convolution 2D de notre image im avec le mask défini précédemment. Cela revient à notre définition de  $C_{(x,y)}$ .

```
1 def texture( im, width ):  
2     diff2 = np.power( im - average( im, width ), 2 )  
3     std = np.sqrt( average( diff2, width ) )  
4     return std / std.max()
```

Finalement, cette fonction texture renverra une image, de même dimension que l'image de base, qui aura pour valeurs à chaque pixel la valeur absolue (la racine carré du carré) de la soustraction de leur niveau de gris et de leur variance à leur voisinage. Ce qui fait écho au calcul de  $V_{(x,y)}$  défini précédemment.

Nous allons maintenant appliquer cette texture aux mêmes images que précédemment pour comparer son efficacité. La texture sera appliqué avec une distance de 2.

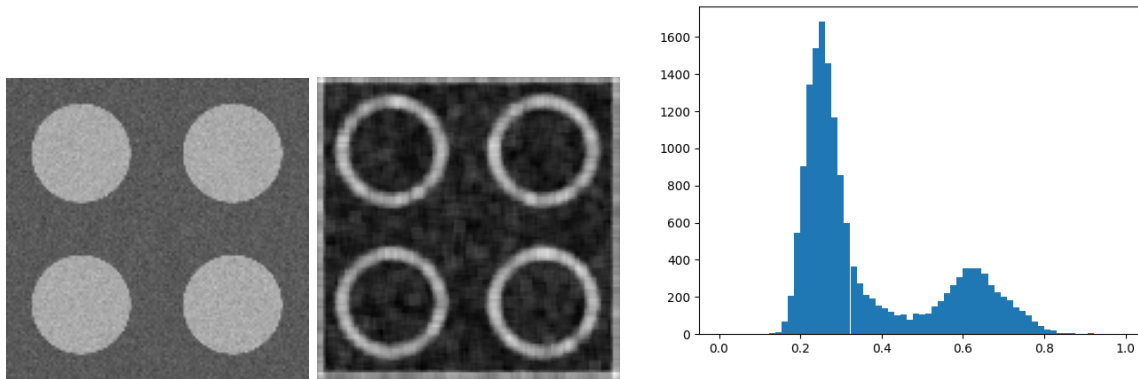


FIGURE 11 – Exemple 0, sa texture et l'histogramme de sa texture

En constatant l'histogramme, on choisira un seuil à 0.5. Nous obtenons ce résultat :

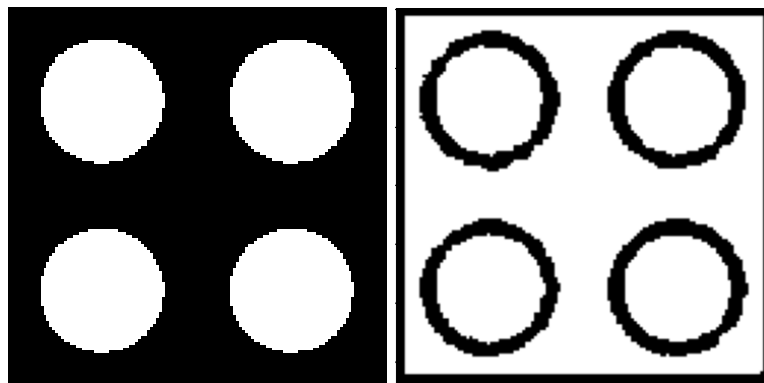


FIGURE 12 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 43.04 %.

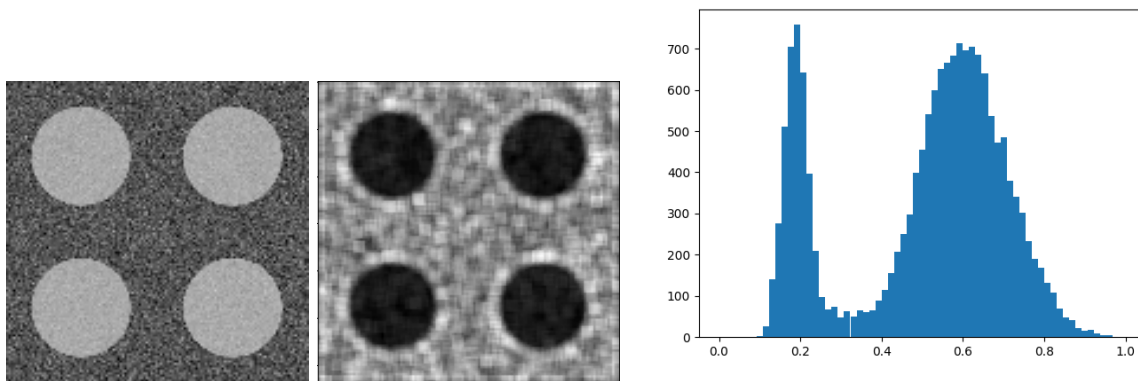


FIGURE 13 – Exemple 1, sa texture et l'histogramme de sa texture

En constatant l'histogramme, on choisira un seuil à 0.325. Nous obtenons ce résultat :



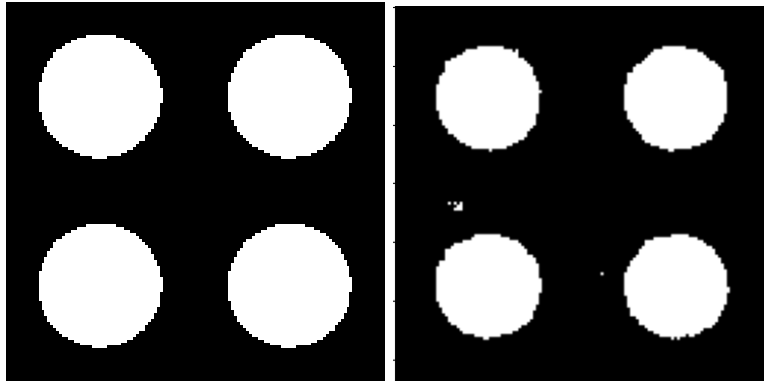


FIGURE 14 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 90.80 %. Les cercles sont bien trouvés mais leur taille est plus petite.

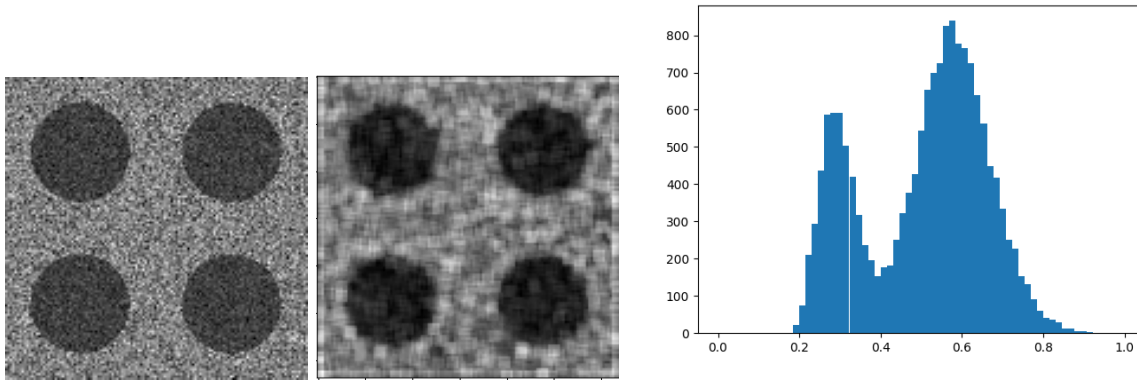


FIGURE 15 – Exemple 2, sa texture et l'histogramme de sa texture

En constatant l'histogramme, on choisira un seuil à 0.4. Nous obtenons ce résultat :

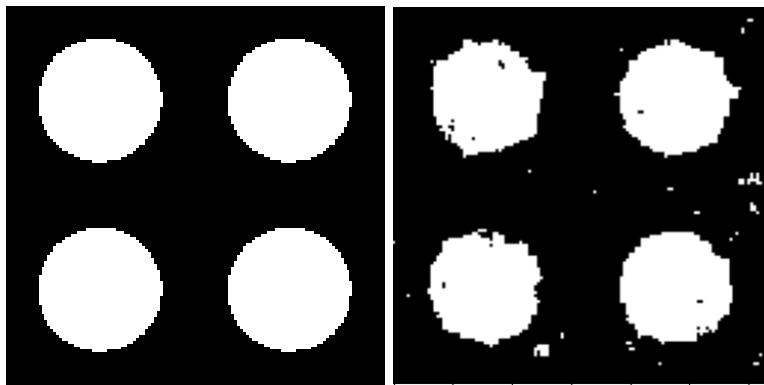


FIGURE 16 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 93.53 %. On voit sur l'image les cercles attendus, à quelques tâches parasites près.

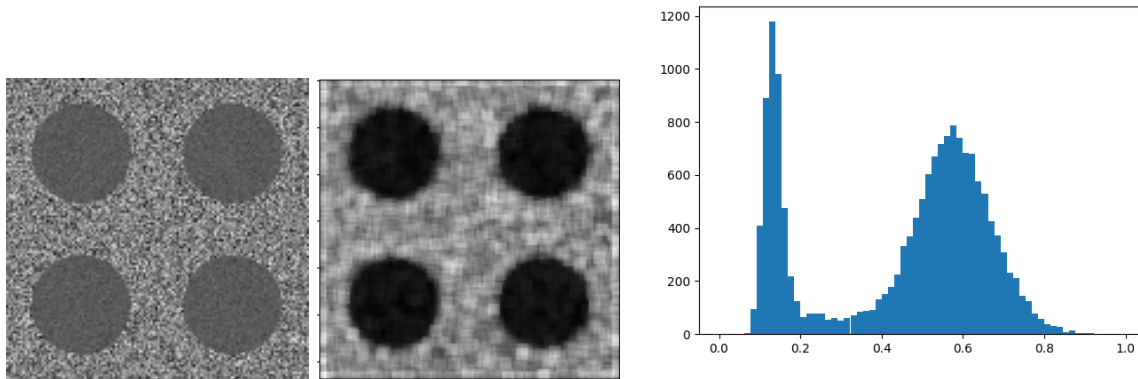


FIGURE 17 – Exemple 3, sa texture et l'histogramme de sa texture

En constatant l'histogramme, on choisira un seuil à 0.3. Nous obtenons ce résultat :

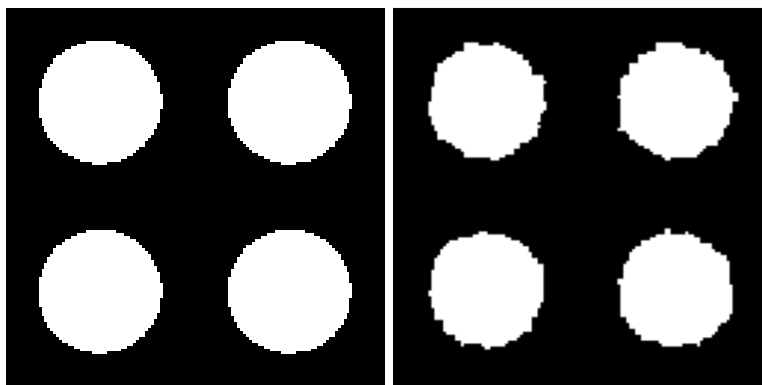


FIGURE 18 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 95.73 %. On voit que les cercles ne sont pas très ronds mais que les formes sont bien devinées.

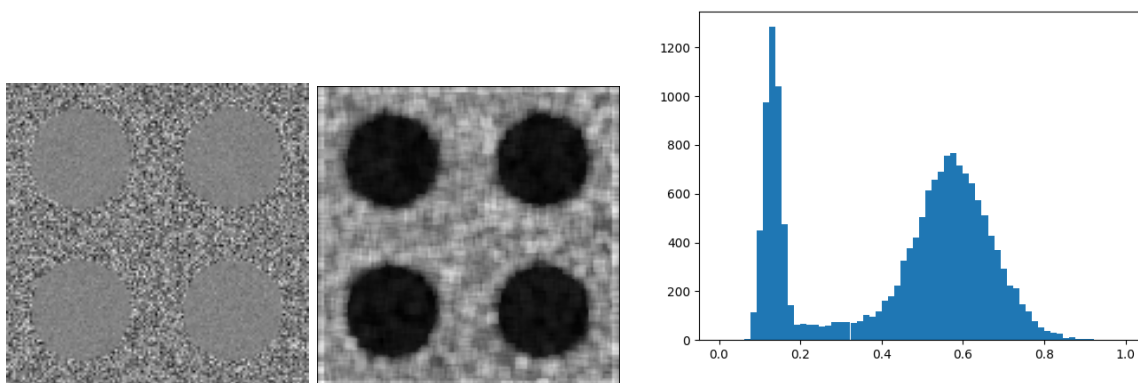


FIGURE 19 – Exemple 4, sa texture et l'histogramme de sa texture

En constatant l'histogramme, on choisira un seuil à 0.3. Nous obtenons ce résultat :

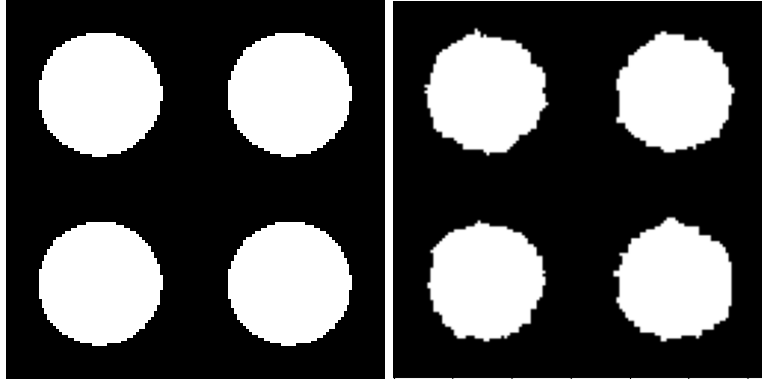


FIGURE 20 – Résultat attendu, Résultat obtenu

La segmentation est ici valide à 96.50 %. On voit que les cercles ont quelques imperfections mais que les formes sont toutefois bien devinées.

Ces résultats nous montre que grâce aux textures, on arrive à segmenter correctement des images assez bruitées mais difficilement des non ou peu bruitées. Son efficacité est "complémentaire" de la binarisation par l'utilisation des niveaux de gris de l'image.

Pour segmenter efficacement une image (peu importe son état) il nous faudrait donc un mélange des deux méthodes abordées.

## 4 Histogramme conjoint et classifieur 2D

Dans cette partie nous allons donc utiliser les deux attributs que sont les valeurs de nuances de gris des pixels et les variances locales des pixels pour effectuer notre binarisation. Cependant, avec un nombre différent d'attributs à prendre en compte, la méthode de binarisation sera elle aussi différente : nous classerons en 2 dimensions et non plus en 1, et l'hyperplan ne sera plus de degré 0 (une valeur de seuil, un point) mais de degré 1 (une droite).

Pour déterminer cette classification, il va falloir regrouper en un seul histogramme les valeurs de nuance de gris et les valeurs de variances locales : nous utiliserons un histogramme conjoint 2D.

### 4.1 Histogramme conjoint

Sur les histogramme 2d suivants, faits à partir de l'image de niveau de gris et l'image de texture, on retrouve en x la texture et en y le niveau de gris. Attention, l'origine du y se trouvera en haut de l'axe et non en bas. Les zones colorées (jaune vert) correspondent à des pics communs pour les deux images.

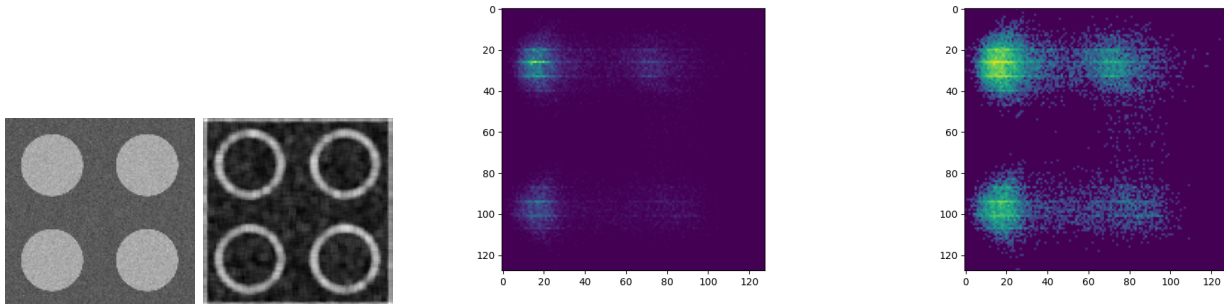


FIGURE 21 – Exemple 0, sa texture et l'histogramme 2d résultant sans et avec log10

En constatant l'histogramme conjoint, on constate qu'il y a beaucoup de pixel avec peu de texture et avec soit un haut niveau de gris soit un faible niveau de gris. Ainsi, la texture n'est ici pas un attribut qui nous permettra de dissocier le fond des formes : l'hyperplan sera donc une droite horizontale centrale à l'histogramme conjoint d'équation  $y - 0.5 = 0$ .

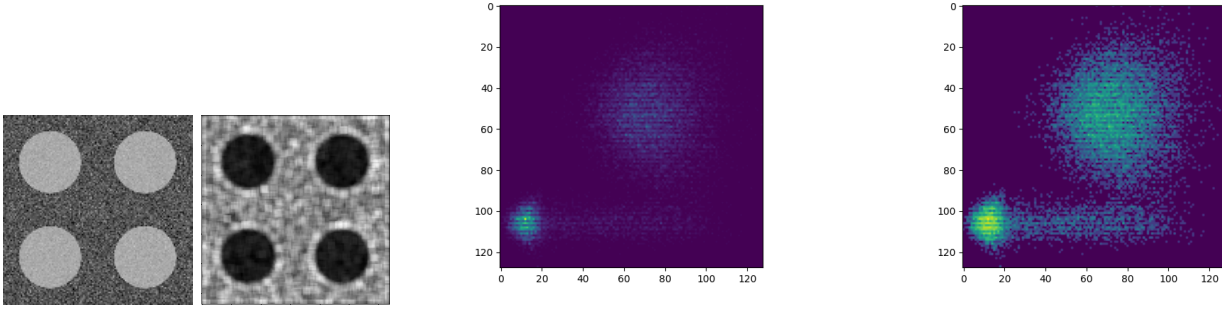


FIGURE 22 – Exemple 1, sa texture et l'histogramme 2d résultant sans et avec log10

Avec cet histogramme ci, on constate deux disques principaux de points. Il va falloir les séparer, et cela sera possible avec une droite diagonale allant du haut gauche jusqu'au bas droit. Cette droite aura comme équation la suivante :  $-0.75x + y - 0.27 = 0$

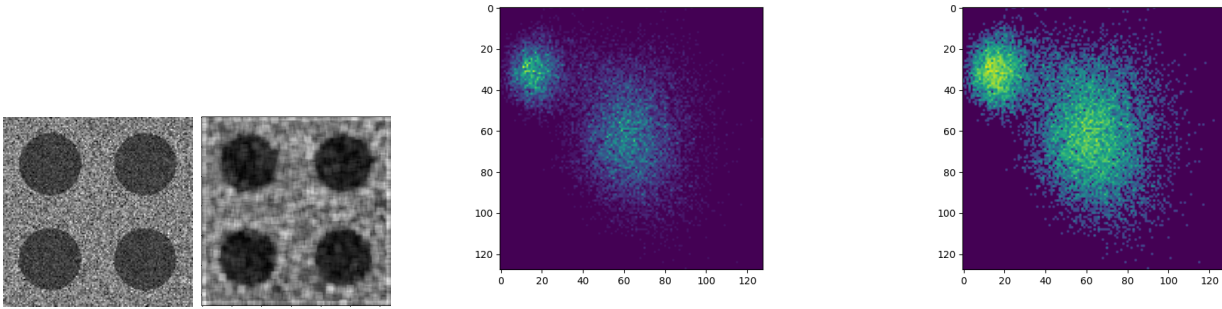


FIGURE 23 – Exemple 2, sa texture et l'histogramme 2d résultant sans et avec log10

Avec cet histogramme ci, on constate également deux disques principaux de points. Il va falloir les séparer, et cela sera possible avec une droite diagonale allant du bas gauche jusqu'au haut droit. Cette droite aura comme équation la suivante :  $-x - 0.6y + 0.66 = 0$

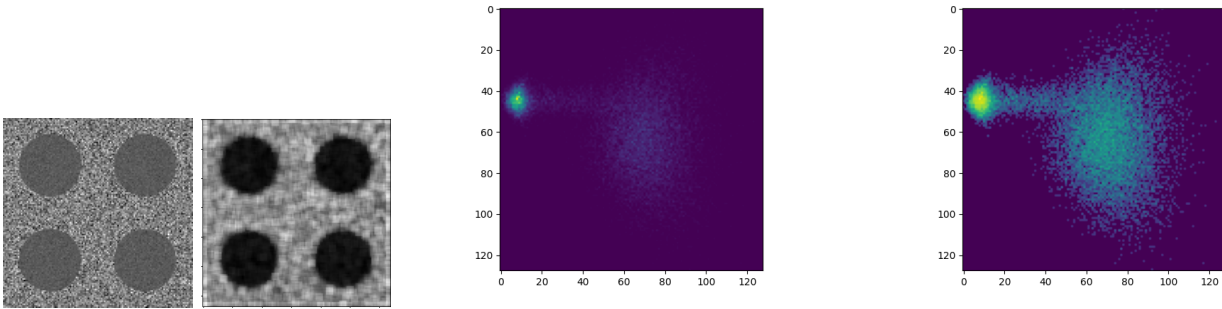


FIGURE 24 – Exemple 3, sa texture et l'histogramme 2d résultant sans et avec log10

Avec cet histogramme ci, on constate deux zones importantes : le petit disque à gauche de forte densité, et le grand disque à plus basse densité. Nous allons les séparer via une droite verticale d'équation :  $-x + 0.3 = 0$

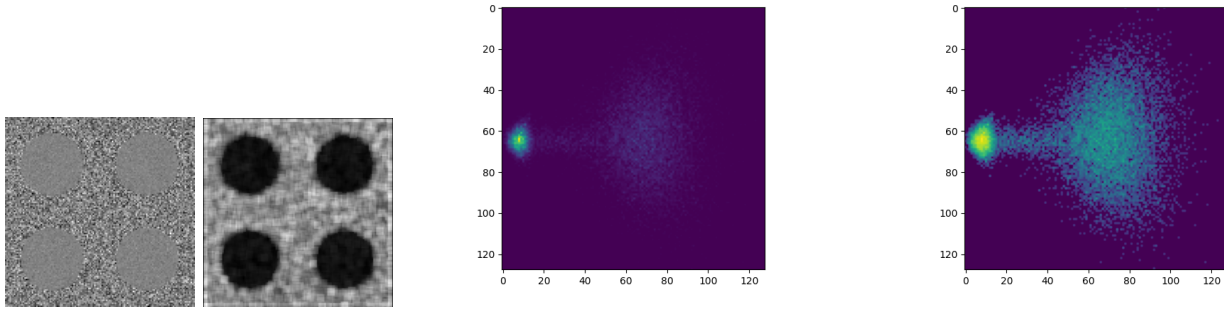


FIGURE 25 – Exemple 4, sa texture et l’histogramme 2d résultant sans et avec log10

Avec cet histogramme ci, on constate deux zones importantes mais similaires à l’histogramme précédent. Nous allons donc également les séparer via une droite verticale d’équation :  $-x + 0.3 = 0$

## 4.2 Classifieur 2D

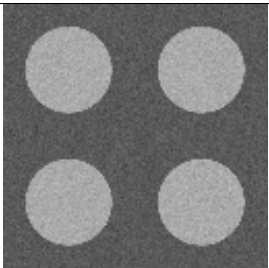
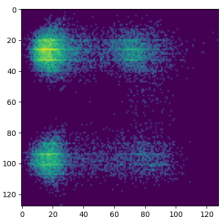
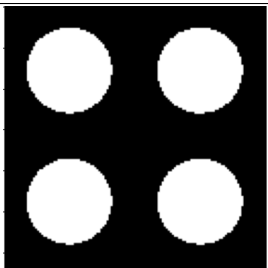
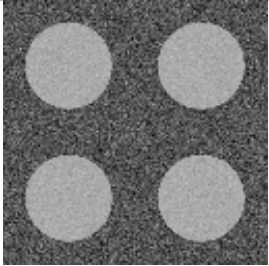
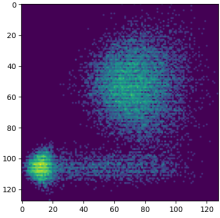
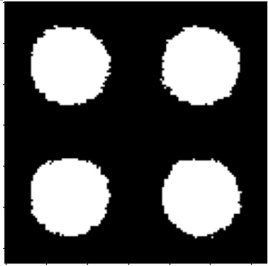
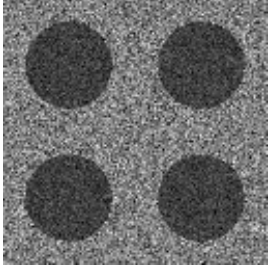
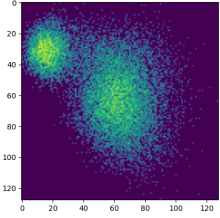
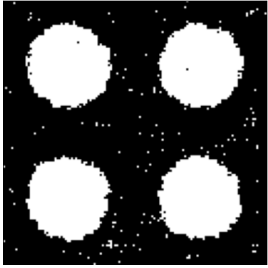
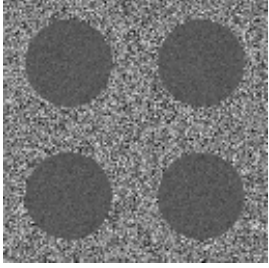
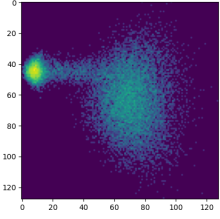
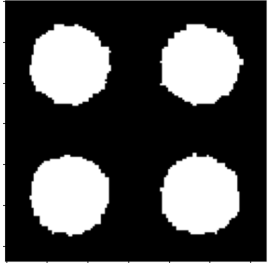
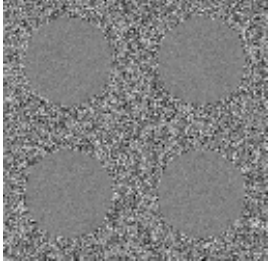
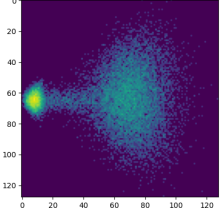
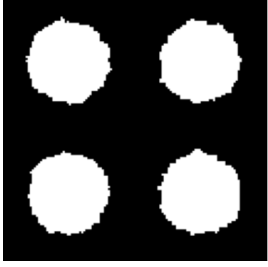
Grâce aux histogrammes 2d obtenues et aux équations de droites définies empiriquement en fonctions de ceux-ci, nous allons pouvoir segmenter nos images avec binarisation. Comme nous avons dorénavant un vecteur d’attribut à deux dimensions (niveaux de gris et niveau de texture), voici notre nouvelle méthode de binarisation :

```

1 def binarize2d(im1, im2, a, b, c):
2     return (c + a * im1 + b * im2 >= 0)

```

Nous utiliserons cette fonction avec les équations de droites définis précédemment : dans l’équation  $ax+by+c = 0$ ,  $b0$  sera le  $c$ ,  $b1$  sera le  $a$  et  $b2$  sera le  $b$ . On notera que dans les cas où  $x$  ou  $y$  valent 0, cela équivaut à faire la binarisation simple vu précédemment (seuillage). Voici donc l’application de nos binarisations sur les images d’exemples :

image	histogramme binarisé	équation de droite	segmentation	pourcentage ressemblance
		$y - 0.5 = 0$		99.88%
		$-0.75x + y - 0.27 = 0$		94.91%
		$-x - 0.6y + 0.66 = 0$		96.48%
		$-x + 0.3 = 0$		95.73%
		$-x + 0.3 = 0$		96.50%

On peut conclure de ces résultats que prendre en compte le niveau de gris et le niveau de texture est une méthode plus acceptable et plus adaptable à toute sorte de situation que de ne considérer qu'un seul de ces deux attributs.