

AIV1 - TP09

## Segmentation supervisée par analyse d'histogramme

Suliac Lavenant et Antoine Nollet

18 March 2022

## Table des matières

1	Introduction	3
2	Seuillage fixe	3
3	Seuillage automatique (Bayes)	4
4	Classification en 3 classes de l'image	5
5	Extraction de la région représentant le chiffre par seuillage supervisé (Bayes)	7
6	Seuillage du chiffre 1 à partir du seuil extrait du chiffre 0	7
A	Annexe 1 - Question 1 et 2	8
B	Annexe 2 - Question 3	10
C	Annexe 3 - Question 4 et 5	11

# 1 Introduction

Lors de cet écrit, l'objectif sera d'utiliser l'apprentissage supervisé via le seuillage supervisé de niveaux de gris afin de segmenter au mieux des images, dans le but de reconnaître les différentes parties de ces images. Nous verrons pour quelles raisons cette méthode est intéressante à utiliser.

## 2 Seuillage fixe

Nous commençons donc pour utiliser une segmentation par seuillage fixe, c'est à dire sans apprentissage, sur cette exemple d'image :

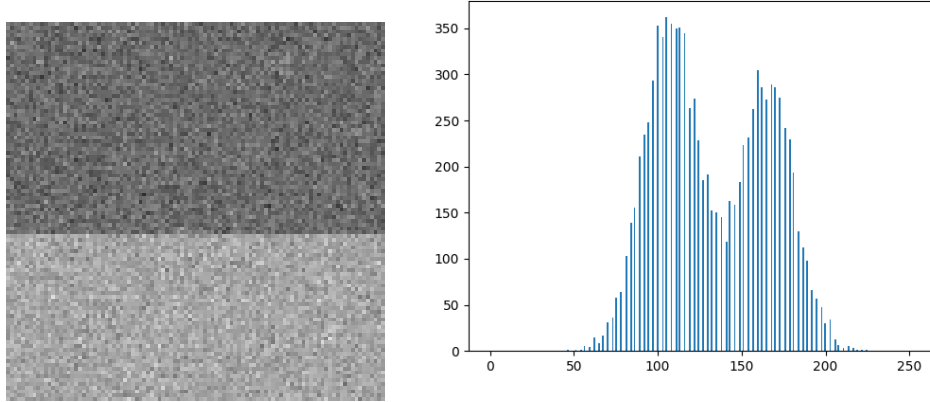


FIGURE 1 – Image 2classes\_100\_100\_8bits et son histogramme

Nous allons empiriquement utiliser des seuils  $\hat{X}$  de 130, 140 et 150 (140 étant la moyenne) pour segmenter l'image en les classes  $\hat{w}_1$  et  $\hat{w}_2$ .

Le seuil  $\hat{X}$  définit donc les 2 classes de pixels  $\hat{w}_1$  et  $\hat{w}_2$  :

$$\hat{\omega}_1 = P \in I | I(P) > \hat{X}$$

$$\hat{\omega}_2 = P \in I | I(P) \leq \hat{X}$$

Ces deux valeurs  $w_1$  et  $w_2$  correspondent à deux classes représentant le fond et la forme séparés par le seuil présumé  $\hat{X}$ .

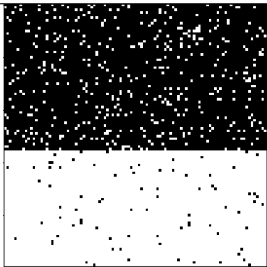
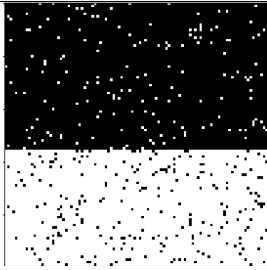
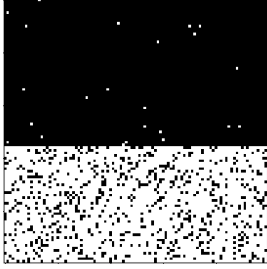
Nous connaissons le résultat attendu de la segmentation, la vérité terrain :



FIGURE 2 – Image de Vérité Terrain

Grâce à la matrice de confusion de l'image de vérité terrain avec l'image seuillée, nous pourrions déterminer le taux de bonne classification selon les seuils.

Voici donc le tableau contenant les résultats obtenues selon les seuils choisis :

seuillage	image segmentée	matrice de confusion	taux bonne classification
130		$\begin{bmatrix} 5147 & 453 \\ 79 & 4321 \end{bmatrix}$	94.67
140		$\begin{bmatrix} 5435 & 165 \\ 238 & 4162 \end{bmatrix}$	95.97
150		$\begin{bmatrix} 5577 & 23 \\ 719 & 3681 \end{bmatrix}$	92.58

En constatant les résultats, on remarque que le seuil de 140 obtient le meilleur taux de bonne classification, c'est donc celui-ci qu'on préférera utiliser.

### 3 Seuillage automatique (Bayes)

Nous allons maintenant introduire de l'apprentissage dans la détermination du seuil à utiliser. Ainsi, le seuil  $\hat{t}$  ne sera plus déterminé empiriquement mais sera déterminé avec la formule suivante :

$$\hat{t} = \operatorname{argmin}(\epsilon(t))$$

où  $\epsilon(t)$  est défini ainsi :

$$\epsilon(t) = \sum_{i=0}^t \frac{h_2(i)}{h(i)} \cdot P(\omega_2) + \sum_{i=t+1}^{255} \frac{h_1(i)}{h(i)} \cdot P(\omega_1)$$

Ici les deux classes  $\omega_1$  et  $\omega_2$  sont connus avant traitement et définition du seuil, voici les images de ces classes :

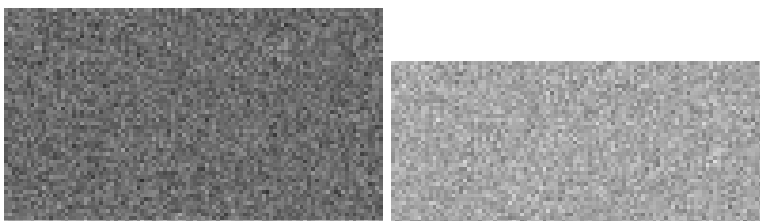


FIGURE 3 –  $\omega_1$  et  $\omega_2$

Avec celles-ci nous allons pouvoir expliquer les valeurs  $P(\omega_1)$  et  $P(\omega_2)$  : il s'agit de la probabilité que lorsqu'on prend un pixel de l'image de base, il appartienne soit à la classe  $\omega_1$  soit à la classe  $\omega_2$ . Ensuite, les valeurs  $h(i)$  correspondent aux occurrences de la valeur  $i$  dans l'histogramme de niveaux de gris de l'image de base. Ainsi, il en va de même pour les valeurs  $h_1(i)$  et  $h_2(i)$  pour les histogrammes respectifs de  $\omega_1$  et  $\omega_2$ .

En utilisant donc la formule précédemment citée, en utilisant les données connues de  $\omega_1$  et  $\omega_2$ , nous déterminons un seuil  $\hat{t}$  égal à 141 donnant le résultat suivant (retrouvez l'implémentation en annexe 1) :

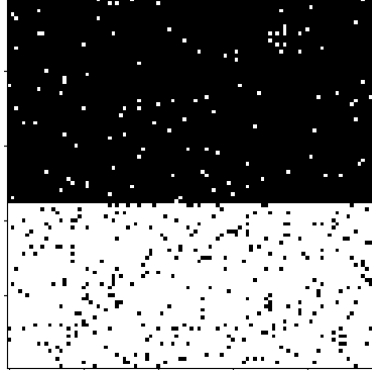


FIGURE 4 – Image segmentée selon le seuil  $t$  obtenu précédemment (141)

Cela nous donne donc un taux de bonne classification de 95,78%, ce qui est cohérent avec ce qui est trouvé précédemment.

## 4 Classification en 3 classes de l'image

Bien que que l'apprentissage nous ait donné une approximation du seuil optimal, qui était trouvable de manière empirique, l'apprentissage nous sera très pratique dans la détermination du ou des seuils. En effet, il devient tout de suite plus difficile de déterminer empiriquement les deux seuils qu'il faudrait pour segmenter une image en 3 classes.

Nous traiterons cette image pour présenter la segmentation en 3 classes :

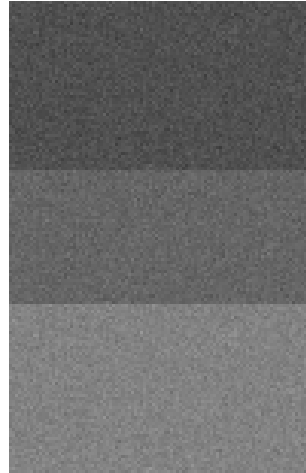


FIGURE 5 – Image de base contenant 3 classes

Les 3 classes  $\omega_1$ ,  $\omega_2$  et  $\omega_3$  seront définis selon deux seuils optimaux  $(\hat{t}_1, \hat{t}_2)$  comme ceci :

$$\begin{aligned}\hat{\omega}_1 &= P \in I | I(P) \leq \hat{t}_1 \\ \hat{\omega}_2 &= P \in I | \hat{t}_1 < I(P) \leq \hat{t}_2 \\ \hat{\omega}_3 &= P \in I | I(P) > \hat{t}_2\end{aligned}$$

Là où précédemment nous cherchions à minimiser les erreurs (minimiser le nombre de pixels mals segmentés : au dessus du seuil mais dans la classe  $\omega_1$  ou en dessous du seuil mais dans la classe  $\omega_2$ ), nous allons maintenant maximiser le nombre de bonne segmentation avec des seuils ( $\hat{t}_1, \hat{t}_2$ ) optimaux :

$$(\hat{t}_1, \hat{t}_2) = \operatorname{argmax}(B(t_1, t_2))$$

où  $B(t_1, t_2)$  est défini ainsi :

$$B(t_1, t_2) = \sum_{i=0}^{t_1} \frac{h_1(i)}{h(i)} \cdot P(\omega_1) + \sum_{i=t_1+1}^{t_2} \frac{h_2(i)}{h(i)} \cdot P(\omega_2) + \sum_{i=t_2+1}^{255} \frac{h_3(i)}{h(i)} \cdot P(\omega_3)$$

Ici également, nous connaissons les 3 classes l'avance :



FIGURE 6 –  $\omega_1, \omega_2$  et  $\omega_3$

Et de plus, nous connaissons également en amont le résultat attendu, la vérité terrain, qui est celle ci :



FIGURE 7 – Image de Vérité Terrain

Avec l'implémentation que vous pouvez retrouver en annexe 2, nous obtenons les seuils optimaux  $t_1 = 91$  et  $t_2 = 115$ . Nous donnant l'image segmentée suivante :

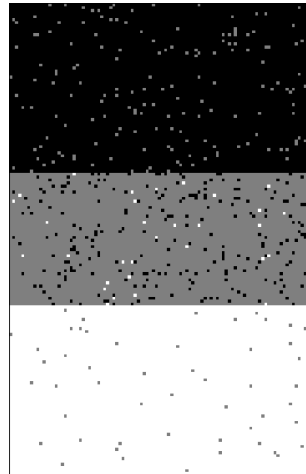


FIGURE 8 – Image classes\_100\_156\_8bits segmentée avec  $t_1 = 91$  et  $t_2 = 115$

Ce résultat a un taux de bonne assignation de 95.95%, résultat qu'il aurait été difficile d'obtenir de manière empirique.

## 5 Extraction de la région représentant le chiffre par seuillage supervisé (Bayes)

Nous allons maintenant traiter des formes plus complexes, prenons l'exemple du chiffre 0 comme ce qui suit (nous considérons 2 classes, soit la forme, soit le fond) :

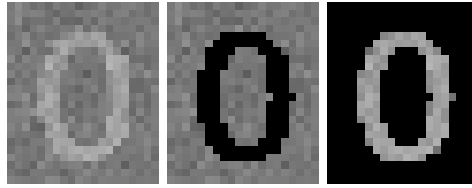


FIGURE 9 – Image du 0, suivi du  $\omega_1$  et du  $\omega_2$  correspondants

La méthode pour définir les histogrammes de niveaux de gris des  $\omega_1$  et  $\omega_2$  sera différente que précédemment, nous ne considérons simplement pas la valeur 0 (car cela représente la découpe de l'image de base pour obtenir l'image de classe). La méthode, mis à part ce point ci, sera très ressemblante à ce qui a été expliquée précédemment (voir Annexe 3 pour le détail de l'implémentation).

On obtient ainsi un seuil de 141, nous pourrions espérer réutiliser ce seuil pour une image du même type, pourquoi pas un 1 à la place du 0...

## 6 Seuillage du chiffre 1 à partir du seuil extrait du chiffre 0

Considérons donc l'image de 1 et sa vérité terrain qui suivent :

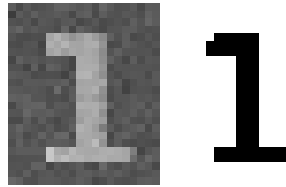


FIGURE 10 – Image du 1, suivi de sa vérité terrain

En réutilisant le seuil de 141 précédemment déterminé grâce au 0, nous obtenons le résultat suivant :

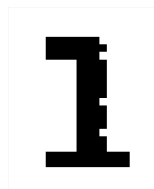


FIGURE 11 – Image du 1 segmentée

Ce résultat a un taux de bonne classification de 97,08%.

Nous pouvons conclure de ces travaux que l'apprentissage peut être intéressant pour définir des seuils optimaux pour segmenter des images en un nombre choisi de classes.

## A Annexe 1 - Question 1 et 2

```
1 #Antoine Nollet et Suliac Lavenant
2
3 import numpy as np
4 import cv2
5 import matplotlib.pyplot as plt
6 from sklearn.metrics import confusion_matrix
7
8 #ouvre une image en niveau de gris et v rifie qu'elle est bien de dimension 2
9 def openImg(name):
10     image = plt.imread( 'images/' + name + '.png' )
11     if image.ndim > 2:
12         image = image[:, :, 0]
13
14     image[:, :] = (image[:, :] * 255).astype(np.uint8)
15
16     return image
17
18 #affiche une image en niveau de gris
19 def showGreyImage(image):
20     plt.imshow(image, cmap='gray' )
21     plt.show()
22
23 #affichage de l'histogramme de image
24 def showHistogramOf(image):
25     nbins=255
26     countsg, edgesg = np.histogram(image, bins=nbins, range=(0,255))
27     plt.hist(edgesg[:-1], nbins, weights=countsg)
28     plt.show()
29
30 #seuille l'image image selon le seuil seuil
31 def seuillage(image, seuil):
32     return np.where(image > seuil, 255, 0)
33
34 #calcule la matrice de confusion de l'image image
35 def calculateConfusionMatrice(image):
36     ImageSeuil_1D = image.flatten()
37     GT_1D = (openImg('2classes_100_100_8bits_GT')).flatten()
38     cm = confusion_matrix(GT_1D, ImageSeuil_1D)
39     return cm
40
41 #operation de la question 1
42 def question1():
43     name='2classes_100_100_8bits'
44
45     image=openImg(name)
46
47     #####affichage image
48     showGreyImage(image)
49
50     #####affichage histogramme
51     showHistogramOf(image)
52
53     #####calcul et affichage de l'image seuill e
54     imageBin = seuillage(image, 150)
55     showGreyImage(imageBin)
56
57     # calcul matrice de confusion et interpretation pour le taux de bonne classification des pixel
58     confusionMatrice = calculateConfusionMatrice(imageBin)
59     print(confusionMatrice)
60     print("taux_de_bonne_classification_des_pixels_de:__"+str((matrice[0,0]+matrice[1,1])/(matrice[0,0]+matrice[0,1]+matrice[1,0]+matrice[1,1])))
61
62 #operation de la question 2
63 def question2():
64
65     name='2classes_100_100_8bits'
66
67     #ouvre les images utilis et les passent en une dimention
```



```

68 image=openImg(name)
69 imageFlat=image.flatten()
70 omega1Flat=openImg(name+"_omega1").flatten()
71 omega2Flat=openImg(name+"_omega2").flatten()
72
73 # on termine les probabilit P1 et P2
74 p1 = len(omega1Flat)/len(imageFlat)
75 p2 = len(omega2Flat)/len(imageFlat)
76
77 #on calcule les histogrammes de omega1 et omega2
78 h1 = [np.count_nonzero(omega1Flat==i) for i in range(256)]
79 h2 = [np.count_nonzero(omega2Flat==i) for i in range(256)]
80
81 minError = float("inf")
82 tmin = 0
83 for t in range(256):#calcule les epsilon pour chaque valeurs possible
84     currentError = 0
85     for i in range(t+1):
86         currentError += p2 * h2[i]
87     for i in range(t+1,255):
88         currentError += p1 * h1[i]
89     if currentError < minError: #retient la valeur de t amenant a la plus petite erreur
90         minError = currentError
91         tmin=t
92
93 print("Valeur_de_seuil_optimale_:_"+str(tmin))
94
95 #affiche l'image avec le seuil d duit
96 imageBin = seuillage(image, tmin)
97 showGreyImage(imageBin)
98
99 #question1()
100 question2()

```

## B Annexe 2 - Question 3

```
1 #Antoine Nollet et Suliac Lavenant
2
3 import numpy as np
4 import cv2
5 import matplotlib.pyplot as plt
6 from sklearn.metrics import confusion_matrix
7
8 #ouvre une image en niveau de gris et v rifie qu'elle est bien de dimension 2
9 def openImg(name):
10     image = plt.imread( 'images/' + name + '.png' )
11     if image.ndim > 2:
12         image = image[:, :, 0]
13
14     image[:, :] = (image[:, :] * 255).astype(np.uint8)
15
16     return image
17
18 #affiche une image en niveau de gris
19 def showGreyImage(image):
20     plt.imshow(image, cmap='gray' )
21     plt.show()
22
23 #affichage de l'histogramme de image
24 def showHistogrameOf(image):
25     nbins=255
26     counttsg, edgesg = np.histogram(image, bins=nbins, range=(0,255))
27     plt.hist(edgesg[:-1], nbins, weights=counttsg)
28     plt.show()
29
30 #seuille l'image image selon le seuil seuil
31 def seuillage(image, seuil):
32     return np.where(image > seuil, 255, 0)
33
34 #seuille l'image image selon les seuils seuil1 et seuil2
35 def doubleSeuillage(image, seuil1, seuil2):
36     imageSeuil1 = np.where(image > seuil1, 127, 0)
37     imageSeuil2 = np.where(image > seuil2, 255, 0)
38     return np.maximum(imageSeuil1, imageSeuil2)
39
40 #calcule la matrice de confusion de l'image image
41 def calculateConfusionMatrice(image):
42     ImageSeuil_1D = image.flatten()
43     GT_ID = (openImg( '3classes_100_156_8bits_GT' )).flatten()
44     cm = confusion_matrix(GT_ID, ImageSeuil_1D)
45     return cm
46
47 def question3():
48     name="3classes_100_156_8bits"
49     image = openImg(name)
50
51     #showGreyImage(image)
52     #showHistogrameOf(image)
53
54
55     imageFlat=image.flatten()
56     omega1Flat=openImg(name+"_omega1").flatten()
57     omega2Flat=openImg(name+"_omega2").flatten()
58     omega3Flat=openImg(name+"_omega3").flatten()
59
60     # on d termine les probabilit P1 et P2
61     p1 = len(omega1Flat)/len(imageFlat)
62     p2 = len(omega2Flat)/len(imageFlat)
63     p3 = len(omega3Flat)/len(imageFlat)
64
65     #on calcule les histogrammes de omega1 et omega2
66     h1 = [np.count_nonzero(omega1Flat==i) for i in range(256)]
67     h2 = [np.count_nonzero(omega2Flat==i) for i in range(256)]
```

```

68     h3 = [np.count_nonzero(omega3Flat==i) for i in range(256)]
69
70     betterAssignment = -1
71     t1Max = 0
72     t2Max = 0
73     for t1 in range(255):
74         for t2 in range(t1+1,256):
75             goodAssignment = 0
76             for i in range(t1+1):
77                 goodAssignment += p1 * h1[i]
78             for i in range(t1+1,t2):
79                 goodAssignment += p2 * h2[i]
80             for i in range(t2+1,255):
81                 goodAssignment += p3 * h3[i]
82             if goodAssignment > betterAssignment:
83                 betterAssignment = goodAssignment
84                 t1Max=t1
85                 t2Max=t2
86
87     print("Valeur_de_seuil_optimale_t1:_"+str(t1Max)+"_et_t2:_"+str(t2Max))
88
89     imageSegmente = doubleSeuillage(image,t1Max,t2Max)
90     confusionMatrice = calculateConfusionMatrice(imageSegmente)
91     print("taux_de_bonne_classification_des_pixels_de:_"+str((confusionMatrice[0,0]+confusionMatrice[1,1])/(
92     showGreyImage(imageSegmente)
93
94
95 question3()

```

## C Annexe 3 - Question 4 et 5

```

1  #Antoine Nollet et Suliac Lavenant
2
3  import numpy as np
4  import cv2
5  import matplotlib.pyplot as plt
6  from sklearn.metrics import confusion_matrix
7
8  #ouvre une image en niveau de gris et v rifie qu'elle est bien de dimension 2
9  def openImg(name):
10     image = plt.imread( 'images/' + name + '.png' )
11     if image.ndim > 2:
12         image = image[:, :, 0]
13
14     image[:, :] = (image[:, :] * 255).astype(np.uint8)
15
16     return image
17
18  #affiche une image en niveau de gris
19  def showGreyImage(image):
20     plt.imshow(image, cmap='gray' )
21     plt.show()
22
23  #affichage de l'histogramme de image
24  def showHistogrammeOf(image):
25     nbins=255
26     counttsg, edgesg = np.histogram(image, bins=nbins, range=(0,255))
27     plt.hist(edgesg[:-1], nbins, weights=counttsg)
28     plt.show()
29
30  #seuille l'image image selon le seuil seuil
31  def seuillage(image, seuil):
32     return np.where(image > seuil, 255, 0)
33
34  #seuille l'image image selon le seuil seuil
35  def seuillageInv(image, seuil):

```

```

36     return np.where(image > seuil, 0, 255)
37
38 #seuille l'image image selon les seuils seuil1 et seuil2
39 def doubleSeuillage(image, seuil1, seuil2):
40     imageSeuil1 = np.where(image > seuil1, 127, 0)
41     imageSeuil2 = np.where(image > seuil2, 255, 0)
42     return np.maximum(imageSeuil1, imageSeuil2)
43
44 #calcule la matrice de confusion de l'image image
45 def calculateConfusionMatrice(image, name):
46     ImageSeuil_1D = image.flatten()
47     GT_1D = (openImg(name+'_GT')).flatten()
48     cm = confusion_matrix(GT_1D, ImageSeuil_1D)
49     return cm
50
51
52 #operation de la question 4
53 def question4():
54
55     name='rdf-chiffre-0-8bits'
56
57     #ouvre les images utilis es et les passe en une dimention
58     imageFlat=openImg(name).flatten()
59     omega1Flat=openImg(name+"_omega1").flatten()
60     omega2Flat=openImg(name+"_omega2").flatten()
61
62     # on d termine les probabilit s P1 et P2
63     p1 = len(omega1Flat)/len(imageFlat)
64     p2 = len(omega2Flat)/len(imageFlat)
65
66     #on calcule les histogrammes de omega1 et omega2
67     h1 = [np.count_nonzero(omega1Flat==i) for i in range(256)]
68     h1[0]=0
69     h2 = [np.count_nonzero(omega2Flat==i) for i in range(256)]
70     h2[0]=0
71
72     minError = float("inf")
73     tmin = 0
74     for t in range(256):#calcule les epsilon pour chaque valeurs possible
75         currentError = 0
76         for i in range(t+1):
77             currentError += p2 * h2[i]
78         for i in range(t+1,255):
79             currentError += p1 * h1[i]
80         if currentError < minError: #retient la valeur de t amenant a la plus petite erreur
81             minError = currentError
82             tmin=t
83
84     print("Valeur_de_seuil_optimale:_"+str(tmin))
85
86     return tmin
87
88
89
90
91 #operation de la question 5
92 def question5():
93     name='rdf-chiffre-1-8bits'
94     #ouvre les images utilis es et les passent en une dimention
95     image=openImg(name)
96
97
98     #recupere le seuil recuperer avec la question pr c dente sur l'image 0
99     seuil = question4()
100
101
102     #affiche l'image avec le seuil
103     imageBin = seuillageInv(image, seuil)# on seuille de cette facon pour inverser la couleur du fond et de la
104     confusionMatrice = calculateConfusionMatrice(imageBin, name)
105     print("taux_de_bonne_classification_des_pixels_de:_"+str((confusionMatrice[0,0]+confusionMatrice[1,1])/
106

```

```
107     showGreyImage (imageBin)
108
109
110 question5 ()
```