Python et traitement d'images

L'objectif de ce court document est de présenter quelques aspects simples du traitement d'images sous Python, et de donner quelques pistes pour aller plus avant.

Plusieurs modules existent pour faire du traitement d'images sous Python, dont PIL et skimage.

Pour rester dans la continuité de la formation proposée l'an dernier, nous nous appuierons ici sur Numpy, Scipy et matplotlib (même si certaines fonctions de Scipy impliquent que PIL soit chargé).

1 Pour commencer

1.1 Ouvrir et écrire des images sur disque

Le code 1 présente une manière d'ouvrir et d'afficher des images présentes sur le disque.

```
# -*- coding: utf-8 -*-
from scipy import misc
#module d'affichage
import matplotlib.pyplot as plt

# ouverture d'une image couleur sur le disque
i = misc.imread('isima.png')

#Affichage en utilisant une figure matplotlib
plt.imshow(i)
#Les axes sont supprimés
plt.axis('off')
plt.show()

# ouverture d'une image en niveaux de gris sur le disque
j = misc.imread('hand.jpg')
#La palette est mise à la palette en niveaux de gris
plt.imshow(j,cmap=plt.cm.gray)
plt.axis('off') #Ote les axes
plt.show()
```

Listing 1 – Ouverture et affichage d'une image. Deux exemples.

Ces images étant alors stockées comme des tableaux, le module Numpy peut naturellement être utilisé pour les traiter.

1.2 Ecrire dans le tableau

Le code 2 montre comment créer une image directement en écrivant des valeurs dans les pixels du tableau (figure 1)

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np
import math

#Création d'image en parcourant les pixels
image = np.zeros((256, 256))
nl, nc = image.shape

for l in range(nl):
        for c in range(nc):
                        image[l][c] = l*math.cos(c)*math.sin(l)

plt.imshow(image,cmap=plt.cm.gray)
plt.title('Exemple de creation d une image')
plt.axis('off') #Ote les axes
plt.show()
```

Listing 2 – Ouverture et affichage d'une image. Deux exemples.

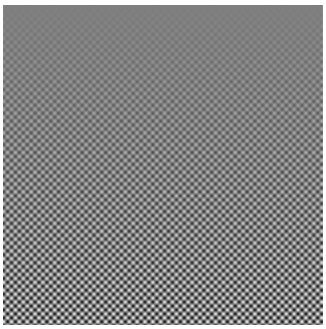


FIGURE 1 – Application du code 2

1.3 Accès à la grille des pixels

Il est possible d'avoir facilement accès à la grille des pixels de l'image, sous forme de tableau, et de faire des traitements matriciels directs sur les images. Le code 3 décrit comment :

- appliquer un masque binaire sur une image donnée, à partir de la définition de ce masque sous forme d'une formule calculée sur la grille des pixels.
- accéder à une série de pixels pour afficher des valeurs prédéfinies

```
# -*- coding: utf-8 -*-
from scipy import misc
import matplotlib.pyplot as plt
import numpy as np

# ouverture d'une image couleur sur le disque
j = misc.imread('hand.jpg')

#récuperation des dimensions
nl, nc = j.shape

#recuperation d'une grille ayant la dimension de l'image
X, Y = np.ogrid[0:nl, 0:nc]

#création d'un masque circulaire
masque = (X - nl / 2) ** 2 + (Y - nc / 2) ** 2 > nl * nc / 8

plt.imshow(masque,cmap=plt.cm.gray)
plt.title('masque')
plt.axis('off')
plt.show()

# Application du masque sur l'image
j[masque]=0

# Accès a une zone de la grille
j[160:210,530:565]=255


#La palette est mise à la palette en niveaux de gris
plt.imshow(j,cmap=plt.cm.gray)
plt.title('image avec masques')
plt.axis('off')

plt.show()
```

Listing 3 – Accès aux pixels.

2 Manipulations de base

2.1 Transformations géométriques élémentaires

Il est très aisé d'appliquer des transformations géométriques élémentaires (linéaires) sur des images. Le code 4 et la figure 2 présentent quelques exemples (région d'intérêt, rotations, symétrie, zoom) et introduit également un exemple de présentation sous la forme de tableaux d'images. les fonctions utilisées proviennent à la fois de numpy et de scipy (import de ndimage, ensemble de fonctions pour le traitement d'images multidimensionnelles). Dans cette série de fonctions, on trouvera également par exemple la définition de transformations affines ou géométriques génériques,

```
# -*- coding: utf-8 -*-
from scipy import misc
from scipy import ndimage
import matplotlib.pyplot as plt
import numpy as np

# ouverture d'une image couleur sur le disque
j = misc.imread('hand.jpg')

#recuperation des dimensions
nl, nc = j.shape

# Définition d'une zone d intérêt
f, tab = plt.subplots(2,  sharey=True)
j_roi = j[160:210,530:565]
tab[0].imshow(j_roi,cmap=plt.cm.gray)
tab[0].set_title('region d interet')
tab[0].axis('off')

# visualisation précise de la zone d'intérêt
tab[1].imshow(j_roi, cmap=plt.cm.gray, interpolation='nearest')
tab[1].set_title('pixels')
tab[1].axis('off')
plt.show()


# symétrie haut-bas
f, ax = plt.subplots(2,2)
ax[0,0].imshow(j,cmap=plt.cm.gray)
ax[0,0].axis('off')
ax[0,0].set_title('originale')
j_flip = np.flipud(j)
ax[0,1].imshow(j_flip,cmap=plt.cm.gray)
ax[0,1].axis('off')
ax[0,1].set_title('symetrie')

 # rotations
j_tournee = ndimage.rotate(j, 30)
ax[1,0].imshow(j_tournee,cmap=plt.cm.gray)
ax[1,0].axis('off')
ax[1,0].set_title('rotation1')
j_tournee2 = ndimage.rotate(j, 30, reshape=False)
ax[1,1].imshow(j_tournee2,cmap=plt.cm.gray)
ax[1,1].axis('off')
ax[1,1].set_title('rotation2')
plt.show()

#zoom avec différents facteurs sur chaque axe
j_zoom = ndimage.zoom(j,[3,5])
plt.imshow(j_zoom,cmap=plt.cm.gray)
plt.title('homothetie')
plt.axis('off')
plt.show()
```

Listing 4 – Quelques transformations géométriques

2.2 Mesures radiométriques

Des mesures statistiques peuvent facilement être effectuées sur les images, en utilisant des fonctions définies à cet effet. Le code 5 présente quelques exemples de ces fonctions. En particulier, il décrit le calcul et l'affichage de l'histogramme. L'histogramme d'une image numérisée comportant n pixels, dont les niveaux de gris sont compris dans [0,L-1] est la fonction discrète :

$$h : [0, L-1] \to \mathbb{R}$$
$$i \mapsto \frac{i}{n}$$

$h(i)$ donne une approximation de la probabilité d'occurrence de i. Le tracé de $h(i)$ fournit une description globale de l'apparence de l'image (figure 3) , qui bien que globale, permet quelques possibilités en traitement d'images. A partir de cet histogramme, de nombreux outils de traitement d'images peuvent être définis (segmentation par seuillage, réhaussement de contraste, modélisation de la distribution des niveaux de gris par mélange de gaussiennes...).

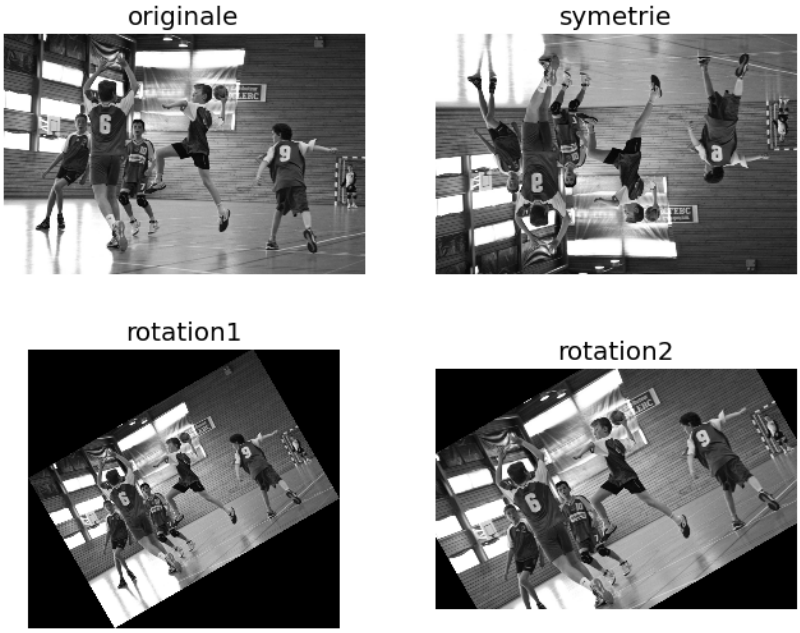


FIGURE 2 – Exemple de résultat du code 4

```
# -*- coding: utf-8 -*-
from scipy import misc
from scipy import ndimage
import matplotlib.pyplot as plt
import numpy as np


j = misc.imread('hand.jpg')

#histogramme de l'image
nbins = 256
h = ndimage.histogram(j,0,255,nbins)
plt.plot(h)
plt.title('histogramme de l image')
plt.xlabel('niveau de gris')
plt.ylabel('occurrences')
plt.show()

# statistiques simples sur l'image
print "la moyenne des niveaux de gris de l'image est ",j.mean()
print "le max des niveaux de gris de l'image est ",j.max()
print "le max des niveaux de gris de l'image est ",j.min()

g = ndimage.center_of_mass(j)
print "le centre de masse des niveaux de gris est en ", g
```

Listing 5 – Quelques mesures radiométriques sur les images.

2.3 Filtrage

Le filtrage est un domaine important du traitement d'images. Il peut être réalisé dans le domaine spatial (des pixels) ou fréquentiel (via une transformée de Fourier).

Le terme spatial se réfère à l'ensemble des pixels composant l'image. Tout traitement spatial peut s'exprimer comme

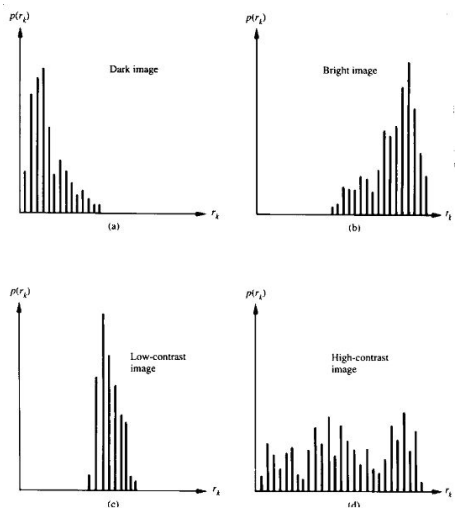


FIGURE 3 – Quelques exemples d'histogrammes

$$J(x, y) = T[I(x, y)]$$

où I est l'image originale, J l'image traitée, et T est un opérateur sur I, défini sur un voisinage de (x, y). Si ce voisinage est de taille 1, T agit pixel par pixel, et on parle alors d'analyse point à point. On récupère alors entre autres les méthodes d'analyse d'histogramme, auxquelles on adjoint toute transformation mathématique $i \mapsto T(i)$ (logarithmique, exponentielle, ...).
Nous envisageons dans la suite un voisinage de taille strictement supérieure à 1, centré en (x, y). et dans ce cadre il existe des filtres linéaires et non linéaires.

Les filtres linéaires sont fondés sur l'hypothèse de linéarité du système d'acquisition. Les filtres dits passe-bas atténuent ou éliminent les hautes fréquences spatiales dans l'image, en laissant les basses fréquences intactes. Au contraire, un filtre passe haut laisse intactes les hautes fréquences, et atténue les basses. Enfin, les filtres passe-bande atténuent ou éliminent une bande de fréquences donnée.

Quel que soit le type de filtrage spatial linéaire envisagé, la démarche est identique. Il s'agit de définir autour d'un pixel (x, y) un voisinage W de taille N impaire, centré en (x, y), dont les coefficients sont par exemple pour $N = 3$:

$w_{-1,-1}$	$w_{-1,0}$	$w_{-1,1}$
$w_{0,-1}$	$w_{0,0}$	$w_{0,1}$
$w_{1,-1}$	$w_{1,0}$	$w_{1,1}$

Le filtrage consiste alors simplement en le balayage de l'image par le masque, et en l'affectation au pixel (x, y) du niveau de gris résultant de la combinaison linéaire des pixels voisins pondérés par les coefficients du filtre :

$$\sum_{i=-1}^{1} \sum_{j=-1}^{1} w_{i,j} I(x - i, y - j)$$

Le filtrage non linéaire, quant à lui, est également réalisé à partir de voisinages centrés, mais agit directement à partir des niveaux de gris $I(x+i, y+j)$ par une transformation non linéaire (max,min, médiane...) ou issue du traitement du signal par exemple (cf. 7 pour un exemple, basé sur le module scipy.signal).

Les figures 4 et 5 donnent deux exemples de filtrage passe haut (détection de contour : Sobel) et passe bas (filtrage moyenneur). Les codes 7 et **??** présentent quelques exemples de filtrage. A noter que bon nombre de fonctions implémentent déjà le filtrage sans qu'il soit nécessaire de définir explicitement le masque ou les opérations.

1	2	1	-1	0	1
0	0	0	-2	0	1
-1	-2	-1	-1	0	1

FIGURE 4 – Filtres de Sobel vertical et horizontal

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

FIGURE 5 – Filtre moyenneur

```
# -*- coding: utf-8 -*-
import scipy as sp
from scipy import misc
from scipy import signal
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt

j = misc.imread('lenagray.png')

#filtre linéaire : norme du gradient approche par un filtrage de Sobel (cf. masque dans document)
plt.figure(figsize=(20,5))
dx = ndimage.sobel(j, 0)  # dérivée horizontale
dy = ndimage.sobel(j, 1)  #  dérivée verticale
mag = np.hypot(dx, dy)  # norme du gradient
mag *= 255.0 / np.max(mag)  # normalisation

plt.subplot(141)
plt.imshow(j,cmap=plt.cm.gray)
plt.title('originale')
plt.axis('off')
plt.subplot(142)
plt.imshow(dx,cmap=plt.cm.gray)
plt.title('Sobel en x')
plt.axis('off')
plt.subplot(143)
plt.imshow(dy,cmap=plt.cm.gray)
plt.title('Sobel en y')
plt.axis('off')
plt.subplot(144)
plt.imshow(mag,cmap=plt.cm.gray)
plt.title('norme du gradient')
plt.axis('off')
plt.subplots_adjust(wspace=0.02, hspace=0.02, top=1, bottom=0, left=0, right=0.9)
plt.show()

#filtres passe bas gaussiens
j_lissee = ndimage.gaussian_filter(j, sigma=3)
j_lissee2= ndimage.gaussian_filter(j, sigma=5)
j_lissee3= ndimage.gaussian_filter(j, sigma=11)

f, tab = plt.subplots(2,2,  sharey=True)
tab[0,0].imshow(j,cmap=plt.cm.gray)
tab[0,0].set_title('originale')
tab[0,0].axis('off')
tab[0,1].imshow(j_lissee, cmap=plt.cm.gray, interpolation='nearest')
tab[0,1].set_title('gaussien 3')
tab[0,1].axis('off')
tab[1,0].imshow(j_lissee2, cmap=plt.cm.gray, interpolation='nearest')
tab[1,0].set_title('gaussien 5')
tab[1,0].axis('off')
tab[1,1].imshow(j_lissee3, cmap=plt.cm.gray, interpolation='nearest')
tab[1,1].set_title('gaussien 11')
tab[1,1].axis('off')
plt.show()

#Réduction de bruit : filtrage non linéaire median et filtrage non linéaire de Wiener
j = misc.imread('hand.jpg')
f, ax = plt.subplots(2,2,  sharex=True)
jbruitee = j +  0.9*j.std() * np.random.random(j.shape)
jmedian = ndimage.median_filter(jbruitee, 3)
jdiff = jmedian-jbruitee
jWiener = sp.signal.wiener(jbruitee)
ax[0,0].imshow(jbruitee,cmap=plt.cm.gray)
ax[0,0].set_title('originale bruitee')
ax[0,0].axis('off')
ax[0,1].imshow(jmedian,cmap=plt.cm.gray)
ax[0,1].set_title('filtrage median')
ax[0,1].axis('off')
ax[1,0].imshow(jdiff,cmap=plt.cm.gray)
ax[1,0].set_title('difference')
ax[1,0].axis('off')
ax[1,1].imshow(jWiener,cmap=plt.cm.gray)
ax[1,1].set_title('Filtrage de Wiener')
ax[1,1].axis('off')

plt.show()
```

Listing 6 – Quelques exemples de filtrage.

```
# -*- coding: utf-8 -*-
from scipy import ndimage
import matplotlib.pyplot as plt
import numpy as np
import math

#image de synthèse : rectangle tourné de 30 degrés
im = np.zeros((256, 256))

im[100:-100, 50:-50] = 1
im = ndimage.rotate(im, 30)
im = ndimage.gaussian_filter(im, 8)

sx = ndimage.sobel(im, axis=0)
sy = ndimage.sobel(im, axis=1,)
sob = np.hypot(sx, sy)

plt.figure(figsize=(16, 5))
plt.subplot(151)
plt.imshow(im, cmap=plt.cm.gray)
plt.axis('off')
plt.title('image de synthese')
plt.subplot(152)
plt.imshow(sx,cmap=plt.cm.gray)
plt.axis('off')
plt.title('Sobel en x')
plt.subplot(153)
plt.imshow(sy,cmap=plt.cm.gray)
plt.title('Sobel en y')
plt.axis('off')
plt.subplot(154)
plt.imshow(sob,cmap=plt.cm.gray)
plt.axis('off')
plt.title('filtre de Sobel')

im += 0.1*np.random.random(im.shape)
sx = ndimage.sobel(im, axis=0, mode='constant')
sy = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)

plt.subplot(155)
plt.imshow(sob,cmap=plt.cm.gray)
plt.axis('off')
plt.title('Sobel sur image bruitee',)

plt.subplots_adjust(wspace=0.02, hspace=0.02, top=1, bottom=0, left=0, right=0.9)
plt.show()
```

Listing 7 – Détection de contour par filtrage de Sobel.

2.4 morphologie mathématique

La morphologie mathématique est un outil mathématique permettant au départ d'explorer la structure géométrique des objets dans une image. Le développement de techniques basées sur ces outils a ensuite permis d'élargir le champ de ses applications, par exemple dans le domaine du réhaussement de contraste ou du filtrage.
Vu sous son angle "reconnaissance des formes", le traitement d'images vise à extraire d'une image donnée des informations de type géométrique (localisation, périmètre, aire, orientation), permettant de distinguer certains objets dans une scène. La plus grande partie des traitements de ce type nécessitent le design d'un *opérateur de forme*, possédant un certain nombre de propriétés attendues (invariance par translation,...) et permettant de discriminer un objet particulier.
Plusieurs problèmes se posent alors, et notamment le fait que les objets sont opaques, et que donc l'information de forme n'est pas additive. En fait, les objets dans une scène se combinent principalement sous deux formes :

- par union ensembliste (recouvrement d'objets) : $X = X_1 \cup X_2$
- par intersection ensembliste (occlusion) : $X = X_2 \setminus X_1 = X_1^C \cap X_2$

L'opérateur de forme Ψ à construire doit alors se distribuer sur l'ensemble des unions et des intersections (équivalent de la linéarité) :

– $\Psi_\delta(X_1 \cup X_2) = \Psi_\delta(X_1) \cup \Psi_\delta(X_2)$
– $\Psi_\epsilon(X_2 \setminus X_1) = \Psi_\epsilon(X_1) \cap \Psi_\epsilon(X_2)$

La première opération va être appelée dans la suite *dilatation morphologique*, et la seconde *érosion morphologique*. Ces deux opérations sont à la base de la *morphologie mathématique*, à partir desquelles des *opérateurs morphologiques* plus complexes vont être construits.

2.4.1 Cas des images binaires

Commençons par quelques définitions de base : pour $A, B \subset \mathbb{Z}^2$, dont les composantes sont notées $a = (a_1, a_2)$ et $b = (b_1, b_2)$,

– la *translation* de A par $x = (x_1, x_2)$, notée $(A)_x$ est $(A)_x = \{c = a + x, a \in A\}$
– la réflexion de B, notée $\hat{B}$ est $\hat{B} = \{x = -b, b \in B\}$
– le complément de A, noté A^C est $A^C = \{x, x \notin A\}$
– la différence de A et B est $A \setminus B = A \cap B^C = \{x, x \in A, x \notin B\}$

Ici, A sera une image binaire, et B un opérateur de forme binaire.

Définition 1 *La dilatation de A par B, notée $A \oplus B$ est l'ensemble défini par*

$$A \oplus B = \{x, (\hat{B})_x \cap A \neq \oslash\}$$

Définition 2 *L'érosion de A par B, notée $A \ominus B$ est l'ensemble défini par*

$$A \ominus B = \{x, (B)_x \subseteq A\}$$

B est l'opérateur de forme, et dans le cadre de la morphologie mathématique on l'appelle l'*élément structurant*.

En clair, pour un objet A binaire et un élément structurant B binaire lui aussi et symétrique, les opérations simples de morphologie mathématique consistent à parcourir l'image et à considérer B comme un masque binaire : si, centré en (x, y), B intersecte A, alors la valeur du dilaté de A par B en (x, y) vaut 1, et 0 sinon. De même si B n'est pas tout inclus dans A, la valeur de l'érodé de A par B en (x, y) vaut 0, et 1 (figure 6). Ainsi, l'érosion rapetisse A, et la dilatation l'étend, selon B (comme les noms sont bien choisis !)

Il est facile de montrer que l'érosion est la transformation duale de la dilatation par rapport à la complémentation :$A \ominus B = (A^C \oplus B)^C$. Ainsi, il est équivalent d'éroder un objet ou de dilater son complémentaire.

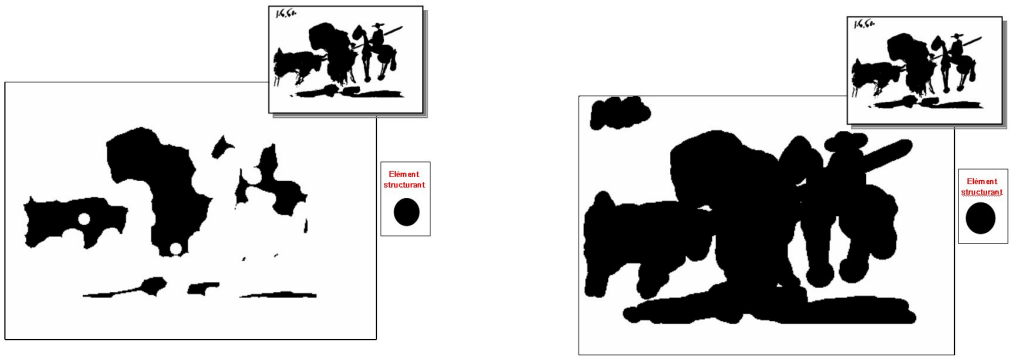
A partir de la dilatation et de l'érosion, on définit deux opérations : l'*ouverture* et la *fermeture*.

Définition 3 *L'ouverture de A par B est définie par*

$$A \circ B = (A \ominus B) \oplus B$$

Définition 4 *La fermeture de A par B est définie par*

$$A \bullet B = (A \oplus B) \ominus B$$



Erosion par un élément structurant circulaire Dilatation par un élément structurant circulaire

FIGURE 6 – Exemple de dilatation et d'érosion

L'ouverture généralement lisse les contours d'une image, casse les liens étroits entre objets (les isthmes), et élimine les petits objets isolés (petits au sens de B). Le lissage et le type de lissage sont déterminés par la forme et la taille de B. La fermeture tend également à lisser les contours, mais rassemble les objets proches (au sens de B), élimine les petits trous (au sens de B) et connecte les contours. La figure 7 présente le résultat de l'ouverture morphologique de l'image précédente, par un élément structurant circulaire.

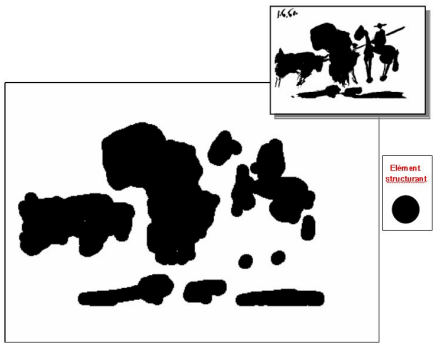


FIGURE 7 – Ouverture morphologique par un élément structurant circulaire

La notion de filtrage prend son importance lorsque l'on considère une forme et une taille adaptée pour B : la figure 8-a présente un tableau d'Henri Matisse (La femme à l'amphore, 1952, un peu de culture), la figure 8-b une version dégradée par par un bruit vertical, et la 8-c l'image restaurée, par ouverture morphologique par un élément structurant adapté.

Le code 8 présente ces opérations élémentaires sur une image binaire, et la figure 9 illustre le résultat

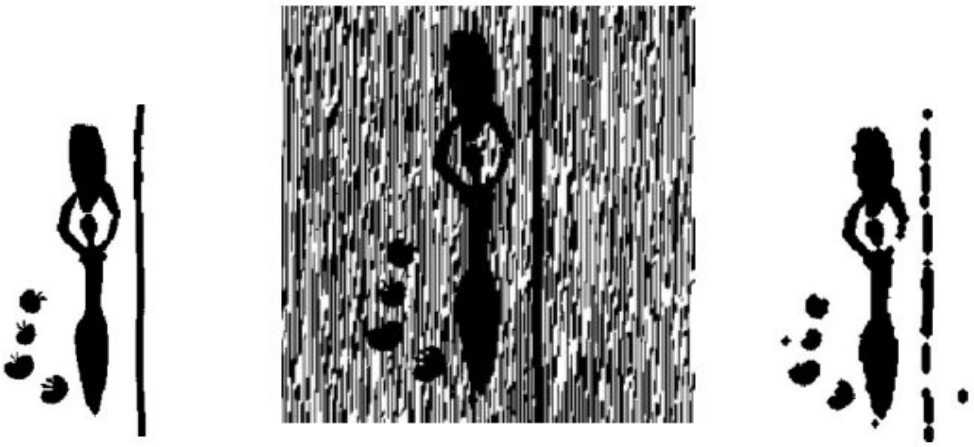


image originale image bruitée image restaurée

FIGURE 8 – Exemple de dilatation et d'érosion

```python
# -*- coding: utf-8 -*-
import scipy as sp
from scipy import misc
from scipy import signal
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt

#élément structurant 2D en croix
el = ndimage.generate_binary_structure(2, 1)

#image de synthèse binaire avec bruit
im= np.zeros((128,128), dtype=np.int)
im[50:-50, 30:-30] = 1
np.random.seed(2)
x, y = (128*np.random.random((2, 100))).astype(np.int)
im[x, y] = 1


erosion = ndimage.binary_erosion(im).astype(im.dtype)
dilatation = ndimage.binary_dilation(im).astype(im.dtype)
ouverture = ndimage.binary_dilation(erosion).astype(im.dtype)
fermeture = ndimage.binary_erosion(dilatation).astype(im.dtype)
plt.figure(figsize=(20,5))
plt.subplot(171)
plt.imshow(im,cmap=plt.cm.gray)
plt.title('originale')
plt.axis('off')
plt.subplot(172)
plt.imshow(erosion,cmap=plt.cm.gray)
plt.title('erosion')
plt.axis('off')
plt.subplot(173)
plt.imshow(im-erosion,cmap=plt.cm.gray)
plt.title('image - erosion')
plt.axis('off')
plt.subplot(174)
plt.imshow(dilatation,cmap=plt.cm.gray)
plt.title('dilatation')
plt.axis('off')
plt.subplot(175)
plt.imshow(dilatation-im,cmap=plt.cm.gray)
plt.title('dilatation - image')
plt.axis('off')
plt.subplot(176)
plt.imshow(ouverture,cmap=plt.cm.gray)
plt.title('ouverture')
plt.axis('off')
plt.subplot(177)
plt.imshow(fermeture,cmap=plt.cm.gray)
plt.title('fermeture')
plt.axis('off')
plt.subplots_adjust(wspace=0.02, hspace=0.02, top=1, bottom=0, left=0, right=0.9)
plt.show()
```

Listing 8 – Morphologie mathématique binaire.

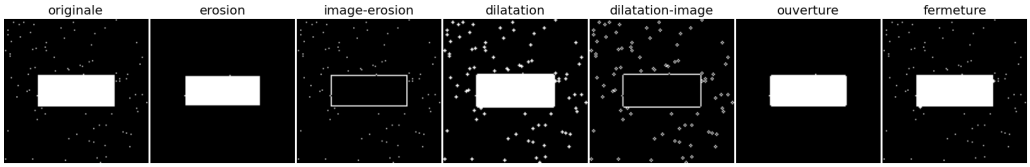


FIGURE 9 – Morphologie mathématique binaire

2.4.2 Les images en niveaux de gris

Ici, $A = I$ est l'image et B est un élément structurant en niveaux de gris (une fonction).

Définition 5 *La dilatation de A par B est*

$$(A \oplus B)(s,t) = \max_{(s-x),(t-y)\in D_I,(x,y)\in D_B} \{I(s-x,t-y) + B(x,y)\}$$

où D_I (resp. D_b) est le domaine de l'image (resp. de l'élément structurant). On a l'habitude d'illustrer cette définition sur des fonctions 1D (figure 10-a, où $I = f$), pour lesquelles la formule précédente se réécrit

$$(A \oplus B)(s,t) = \max_{(s-x)\in D_I,x\in D_B} \{I(s-x) + B(x)\}$$

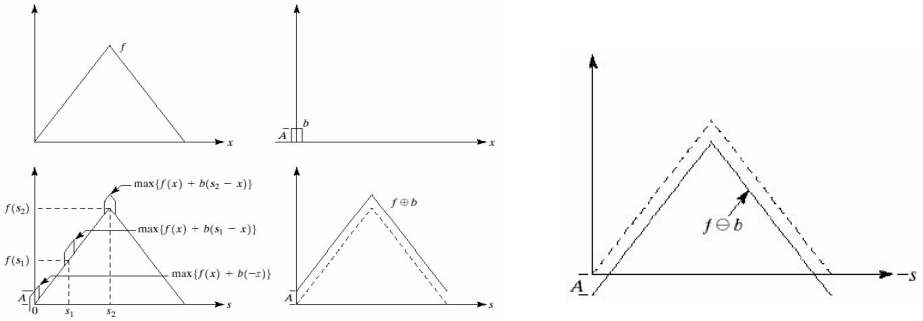


Illustration du processus de dilatation Illustration du processus d'érosion

FIGURE 10 – Illustration de la dilatation en niveaux de gris (d'après [2])

De même, on peut définir l'érosion d'une image en niveaux de gris par un élément structurant en niveaux de gris :

Définition 6 *L'érosion de A par B est*

$$(A \ominus B)(s,t) = \min_{(s+x),(t+y)\in D_I,(x,y)\in D_B} \{I(s+x,t+y) - B(x,y)\}$$

et l'illustration correspondante en 1D est décrite sur la figure 10-b.

Ces deux définitions permettent là encore de développer des opérations de morphologie mathématique plus complexes (ouverture, fermeture, mais aussi squelettisation, transformation en tout ou rien, filtrages..).

3 Etude de cas : Segmentation et quantification

Segmenter une image I, c'est trouver une partition de cette dernière, c'est à dire un ensemble $P = (P_1, P_2, \cdots P_g)$ de parties non vides de I tel que :

1. $(\forall k \neq l) P_k \cap P_l = \emptyset$
2. $\cup_{i=1}^{g} P_i = \Omega$
3. $(\forall k) P_k \neq \emptyset$

Ces parties peuvent être trouvées directement ou c'est leurs bords qu'il peut être intéressant de déterminer. Ainsi, la segmentation d'images s'aborde soit à l'aide d'approches régions (on trouve les P_i), soit à l'aide d'approches frontières (on trouve les bords des parties). Il n'est pas question ici de détailler l'ensemble de ces approches, mais de proposer un outil simple de segmentation d'image, basé sur l'analyse de l'histogramme.

Le code 9 propose une méthode simple de segmentation : à partir d'une image de synthèse (blobs créés aléatoirement en binaire, puis bruités et transformés en image en niveaux de gris), l'histogramme est calculé et un seuil à 50% du max des niveaux de gris est appliqué. La figure 11 donne le résultat correspondant. Le seuillage s'opérant sur l'histogramme, aucune information spatiale n'est prise en compte (seuls les niveaux de gris au dessus de la moitié du max dans l'image sont comptés comme dans l'objet). On constate alors des blobs piquetés de noir, et un fond piqueté en blanc. Il faut alors post traiter cette image, puisque l'on suppose que les blobs (et le fond) doivent être uniformes. On peut par exemple utiliser des outils de morphologie mathématique, comme il est proposé dans la seconde partie du code 9, ce qui permet d'obtenir l'image finale segmentée 12. A partir de cette image, un comptage de composantes connexes peut être effectué à l'aide de la fonction label (module ndimage) (13)

```
# -*- coding: utf-8 -*-
import scipy as sp
from scipy import misc
from scipy import signal
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt
from matplotlib import *

nl = 256
n  = 10

#image de synthèse (blobs)
im = np.zeros((nl, nl))
np.random.seed(1)
points = nl*np.random.random((2, n**2))
im[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
im = ndimage.gaussian_filter(im, sigma=nl/(4.*n))
mask = (im > im.mean()).astype(np.float)
mask += 0.1 * im
img = mask + 0.2*np.random.randn(*mask.shape)

#histogramme de l'image : retourne l'histogramme et les intervalles des classes
hist, classes = np.histogram(img, bins=60)
# centres des intervalles
centres_classes = 0.5*(classes[:-1] + classes[1:])
#seuillage (au dessus de 0.5 de l'histogramme)
image_seuillee = img > 0.5

plt.figure(figsize=(30,6))
plt.subplot(131)
plt.imshow(img,cmap=plt.cm.gray)
plt.imsave('originale.png',img)
plt.title('Originale')
plt.axis('off')
plt.subplot(132)
plt.plot(hist)
plt.title('histogramme')
plt.xlabel('niveau de gris')
plt.ylabel('occurrences')
plt.subplot(133)
plt.imshow(image_seuillee,cmap=plt.cm.gray)
plt.title('image segmentee')
plt.axis('off')
plt.subplots_adjust(wspace=0.2, hspace=0.5, top=1, bottom=.1, left=0, right=0.9)
plt.show()

# post traitement spatial
#ouverture pour supprimer les petits objets blancs du fond
ouverture = ndimage.binary_opening(image_seuillee)
#fermeture de cette image pour rendre homogènes les objets
fermeture = ndimage.binary_closing(ouverture)

plt.figure(figsize=(25,6))
plt.subplot(121)
plt.imshow(ouverture,cmap=plt.cm.gray)
plt.title('Ouverture')
plt.axis('off')
plt.subplot(122)
plt.imshow(fermeture,cmap=plt.cm.gray)
plt.title('Fermeture')
plt.axis('off')
plt.subplots_adjust(wspace=0.2, hspace=0.5, top=.9, bottom=0, left=0, right=0.9)
plt.show()

#composantes connexes en 4-connexité
s = [[0, 1, 0], [1,1,1], [0,1,0]]
lab,nb_objets = ndimage.label(fermeture, s)
plt.imshow(lab)
plt.title('Composantes connexes')
plt.axis('off')
plt.show()
```

Listing 9 – Seuillage d'histogramme.

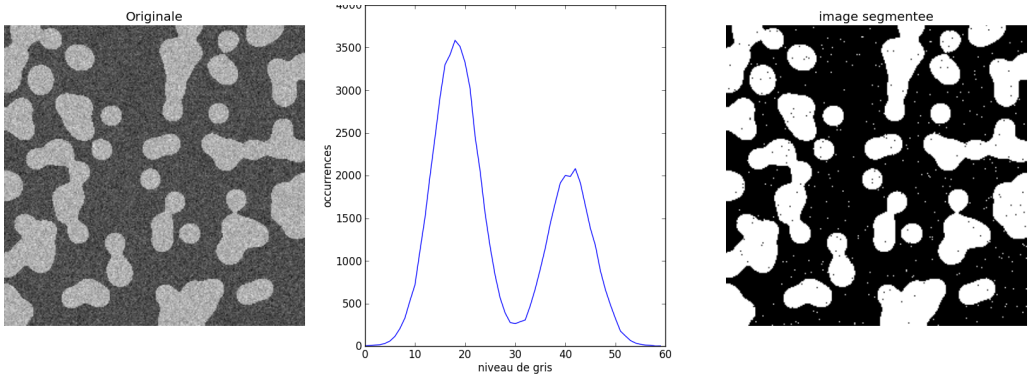


FIGURE 11 – Seuillage d'histogramme

```
# -*- coding: utf-8 -*-
import scipy as sp
from scipy import misc
from scipy import signal
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt


img = misc.imread('originale.png')
fermeture = misc.imread('fermeture.png')
print(fermeture)
s = [[0, 1, 0], [1,1,1], [0,1,0]]
lab, nb_objets = ndimage.label(fermeture, s)

#Quantification : taille et valeurs des intensités des régions
tailles = ndimage.sum(fermeture, lab, range(nb_objets + 1))
print(tailles)
val_moyennes = ndimage.mean(img, lab, range(1, nb_objets + 1))
print(val_moyennes)

# superposition des centres de masse
centroid = ndimage.measurements.center_of_mass(fermeture,lab,xrange(1,nb_objets+1))
plt.imshow(lab)
for i in range(nb_objets):
        plt.plot(centroid[i][1],centroid[i][0],marker='o')
plt.title('Centres de masse')
plt.axis('off')
plt.show()

# suppression des petites composantes connexes
seuil = 1200
masque = tailles < seuil
trop_petites = masque[lab]
lab2=lab.copy()
lab2[trop_petites] = 0
plt.figure(figsize=(25,6))
plt.subplot(121)
plt.imshow(lab)
plt.title('Toutes composantes')
plt.axis('off')
plt.subplot(122)
plt.imshow(lab2)
plt.title('Plus grosses composantes')
plt.axis('off')
plt.show()

# Classement des composantes connexes
labels = np.unique(lab2)
lab2 = np.searchsorted(labels, lab2)
plt.imshow(lab2)
plt.title('Plus grosses composantes')
plt.axis('off')
plt.show()

#région d intérêt d'un object donné
zone_x, zone_y = ndimage.find_objects(lab==4)[0]
region = im[zone_x, zone_y]
plt.imshow(region)
```

Listing 10 – quelques outils de quantification.

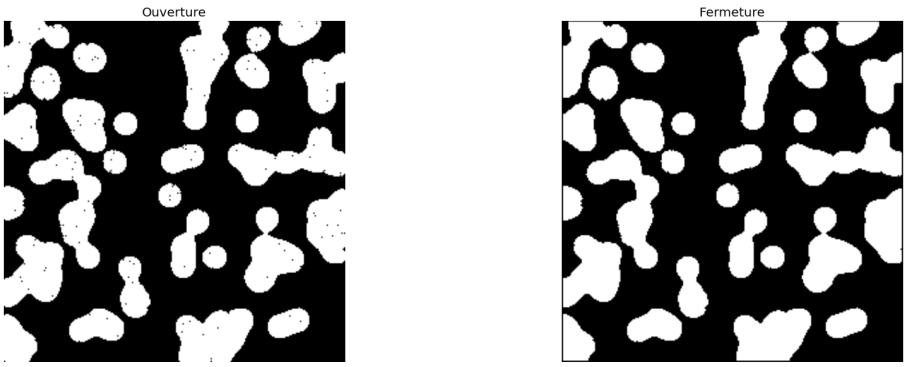


FIGURE 12 – Seuillage d'histogramme et post traitement spatial

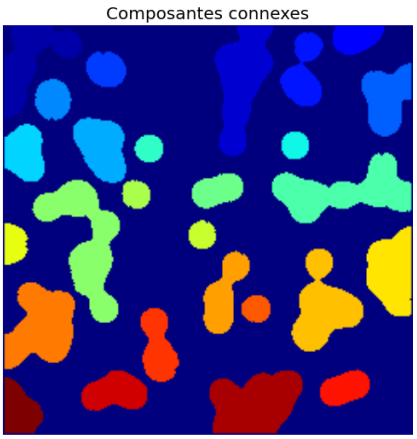


FIGURE 13 – Composantes connexes

A partir de l'image précédente, il est possible de mesurer les paramètres géométriques et radiométriques des composantes extraites (tailles, niveaux moyens), de ne retenir que les plus grosses composantes (figure14 , de les classer par taille, de trouver une composante donnée et sa région, de calculer les centres de masse ... (cf code 10 et figure 15)

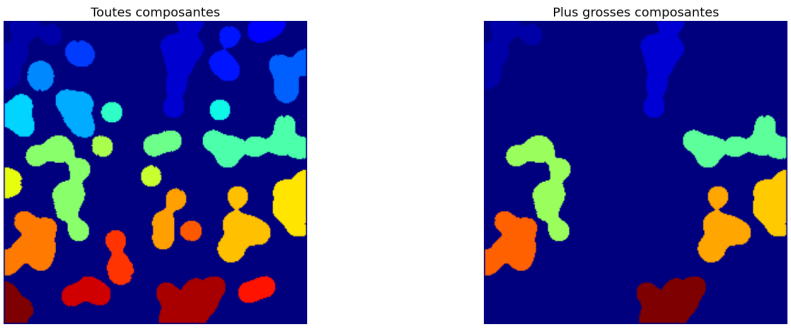


FIGURE 14 – Plus grandes composantes connexes

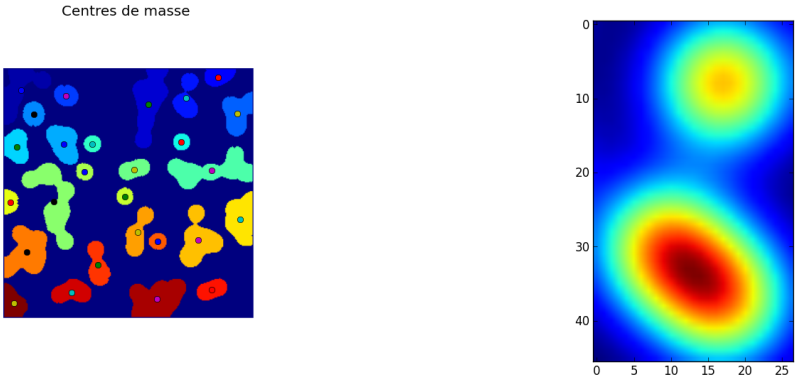


FIGURE 15 – centres de masse, région d'une composante