

# MGM657 Outils Numériques pour l'Ingénieur

## Traitement d'Images

[ludovic.charleux@univ-savoie.fr](mailto:ludovic.charleux@univ-savoie.fr)

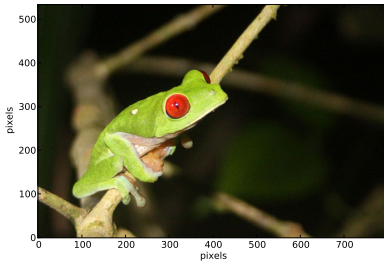
[www.polytech.univ-savoie.fr](http://www.polytech.univ-savoie.fr)

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours

# Lire et afficher une image

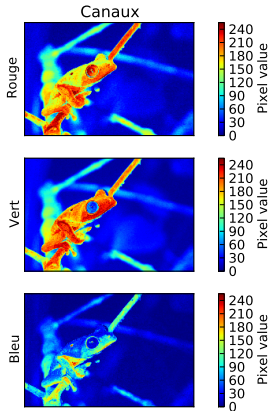


```
1 from PIL import Image
2 import numpy as np
3 import matplotlib.pyplot as plt
4 im = Image.open(
5     '../data/grenouille.jpg')
6 fig = plt.figure(0)
7 plt.clf()
8 plt.imshow(im, origin = "lower")
9 plt.xlabel("pixels")
10 plt.ylabel("pixels")
11 plt.show()
```

## Points clés

- PIL : utile pour lire et écrire divers formats d'images.
- Matplotlib : permet d'afficher les images

# Canaux et couleurs



```

1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 im = Image.open('../data/grenouille.jpg')
5 rouge, vert, bleu = im.split()
6 rouge = np.array(rouge)
7 vert = np.array(vert)
8 bleu = np.array(bleu)

```

## Types d'images

- Canal : 1 information (entier 8 bits)
- Image couleur : 3(+1) canaux
- Imagerie monochrome : 1 canal

## Remarque

- Contexte scientifique : généralement un seul canal.
- On peut afficher une image monochrome avec une échelle de couleurs.

## Image = np.array

```

1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from matplotlib import cm
5 im = Image.open('../data/grenouille.jpg')
6 rouge, vert, bleu = im.split()
7 z = np.array(rouge)

```

```

1 >>> z
2 array([[16, 17, 19, ..., 10, 9, 8],
3        [14, 15, 17, ..., 10, 9, 8],
4        [15, 16, 18, ..., 11, 9, 8],
5        ...,
6        [25, 24, 24, ..., 19, 20, 20],
7        [24, 23, 23, ..., 17, 19, 19],
8        [23, 23, 22, ..., 18, 18, 18]], dtype=uint8)
9 >>> z.shape
10 (534, 800)
11 >>> nx, ny = z.shape
12 >>> z.dtype
13 dtype('uint8')

```

# Sauvegarde

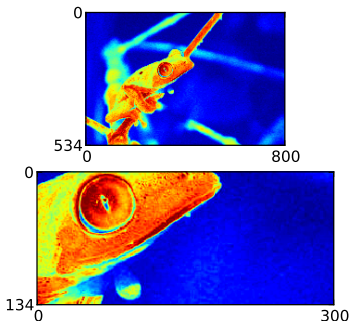


```
1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from matplotlib import cm
5 im = Image.open('../data/grenouille.jpg')
6 rouge, vert, bleu = im.split()
7 z = np.array(rouge)
8 z = np.uint8(cm.copper(z)*255)
9 im2 = Image.fromarray(z)
10 im2.save("grenouille_saved.jpg")
```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques**
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours

# Rognage (Crop)



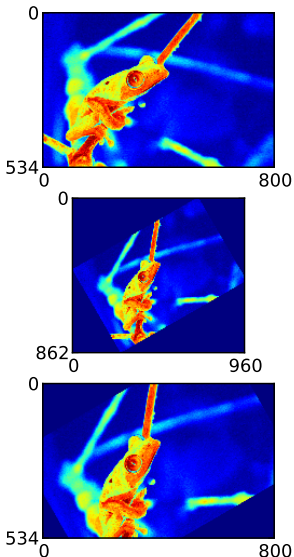
```

1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from matplotlib import cm
5  im = Image.open('../data/grenouille.jpg')
6  rouge, vert, bleu = im.split()
7  z = np.array(rouge)
8  ny, nx = z.shape
9  cx, cy = 200, 250
10 zc = z[cx:-cx, cy:-cy]
11 nyc, nxc = zc.shape
12
13 fig = plt.figure(0) # On cree une figure
14 plt.clf()
15 ax1 = fig.add_subplot(3,1,1)
16 plt.imshow(z, origin = "upper")
17 plt.xticks([0, nx])
18 plt.yticks([0, ny])
19 ax2 = fig.add_subplot(3,1,2)
20 plt.imshow(zc, origin = "upper",
21            interpolation = "nearest")
22 plt.xticks([0, nxc])
23 plt.yticks([0, nyc])
24 plt.show()

```



# Rotation



```

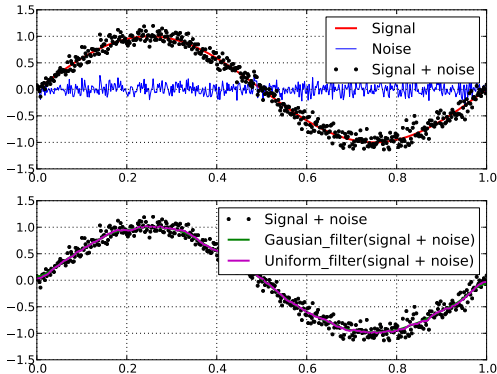
1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from matplotlib import cm
5 from scipy import ndimage
6 im = Image.open('../data/grenouille.jpg')
7 rouge, vert, bleu = im.split()
8 z = np.array(rouge)
9 zrr = ndimage.rotate(z, 30.)
10 zrn = ndimage.rotate(z, 30.,
11     reshape = False)
12
13 ny, nx = z.shape
14 nyrr, nxrr = zrr.shape
15 nyrn, nxrn = zrn.shape
16
17 fig = plt.figure(0) # On cree une figure
18 plt.clf()
19 ax1 = fig.add_subplot(3,1,1)
20 plt.imshow(z, origin = "upper")
21 plt.xticks([0, nx])
22 plt.yticks([0, ny])
23 ax2 = fig.add_subplot(3,1,2)

```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage**
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours

## Lissage : exemple sur un signal 1D



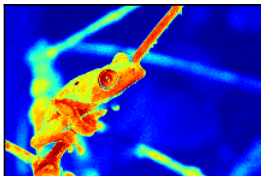
```

1 import numpy as np
2 from matplotlib import pyplot as plt
3 from scipy import ndimage
4 x = np.linspace(0., 1., 500)
5 y_perf = np.sin(2. * np.pi * x)
6 noise = np.random.normal(loc = 0., scale = .1, size = len(x))
7 y = y_perf + noise
8 yg = ndimage.gaussian_filter(y, 10.)
9 ym = ndimage.uniform_filter(y, 20)

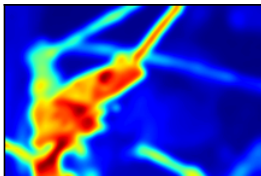
```

## Lissage image

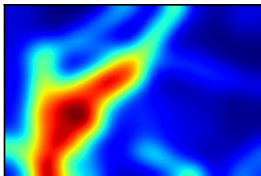
Raw



Blurred



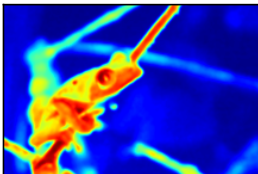
Blurred more !



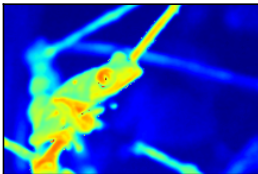
```
1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from scipy import ndimage
5 im = Image.open('../data/grenouille.jpg')
6 rouge, vert, bleu = im.split()
7 z = np.array(rouge)
8 # Blur
9 zg10 = ndimage.gaussian_filter(z, 10.)
10 # Blur more !
11 zg30 = ndimage.gaussian_filter(z, 30.)
```

# Histogramme

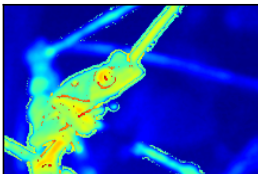
Raw (blurred)



Sharpened



Sharpened

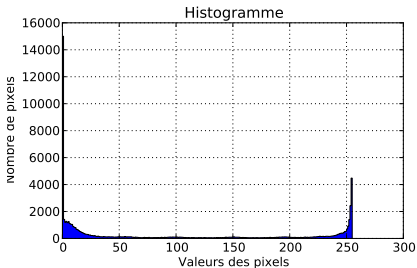
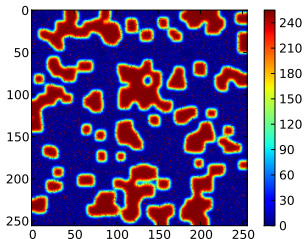


```
1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from scipy import ndimage
5 im = Image.open('../data/grenouille.jpg')
6 rouge, vert, bleu = im.split()
7 z = ndimage.gaussian_filter(np.array(rouge),
8                             4)
9 # Sharpen
10 k = .5
11 zs1 = z + k * (z - ndimage.gaussian_filter(z,
12                                             1.))
13 zs2 = z + k * (z - ndimage.gaussian_filter(z,
14                                             2.))
```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme**
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours

# Histogramme



```

1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  im = Image.open('../Slides/figures/
      image.jpg')
5  channels = im.split()
6  z = np.array(channels[0])
7  N = z.size
8  n_classes = int(N**.5)
9  fig = plt.figure()
10 plt.clf()
11 fig.add_subplot(2, 1, 1)
12 plt.imshow(z, origin = "upper")
13 plt.colorbar()
14 fig.add_subplot(2, 1, 2)
15 plt.title('Histogramme')
16 plt.ylabel('Nombre de pixels')
17 plt.xlabel('Valeurs des pixels')
18 plt.hist(z.flatten(), bins=n_classes
      , histtype = "stepfilled")
19 plt.grid()
20 plt.show()

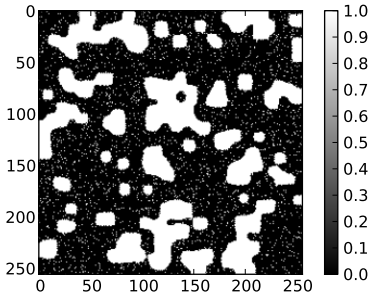
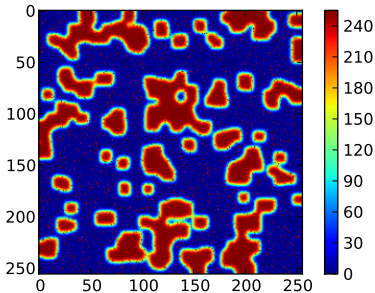
```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage**
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours



## Seuillage



```

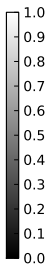
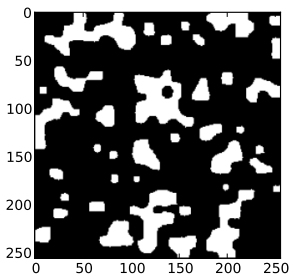
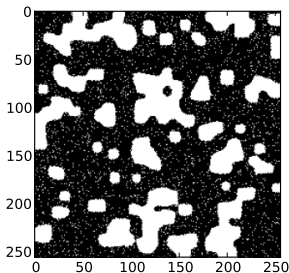
1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from matplotlib import cm
5  im = Image.open('../Slides/figures/
        image.jpg')
6  channels = im.split()
7  z = np.array(channels[0])
8  seuil = 150.
9  zs = z > seuil
10 fig = plt.figure()
11 plt.clf()
12 fig.add_subplot(2, 1, 1)
13 plt.imshow(z, origin = "upper")
14 plt.colorbar()
15 fig.add_subplot(2, 1, 2)
16 plt.imshow(zs, origin = "upper", cmap =
        cm.gray)
17 plt.colorbar()
18 plt.show()

```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation**
- 7 Comptage
- 8 Contours

## Érosion

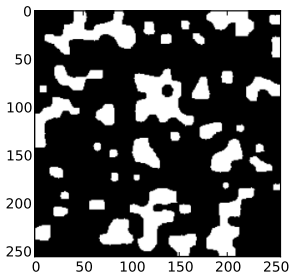
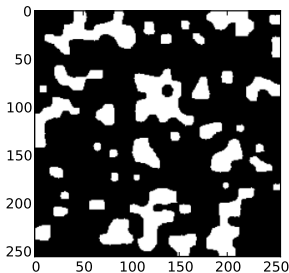


```

1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from matplotlib import cm
5  im = Image.open('../Slides/figures/
      image.jpg')
6  channels = im.split()
7  z = np.array(channels[0])
8  seuil = 150.
9  zs = z > seuil
10 zss = ndimage.morphology.binary_erosion
      (zs, structure=np.ones((3,3)))
11 fig = plt.figure()
12 plt.clf()
13 fig.add_subplot(2, 1, 1)
14 plt.imshow(zs, origin = "upper", cmap =
      cm.gray)
15 plt.colorbar()
16 fig.add_subplot(2, 1, 2)
17 plt.imshow(zss, origin = "upper", cmap
      = cm.gray)
18 plt.colorbar()
19 plt.show()

```

# Dilatation



```

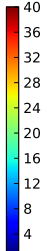
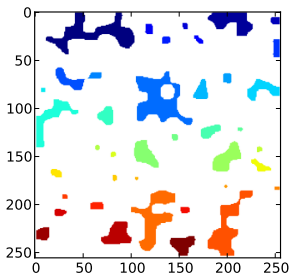
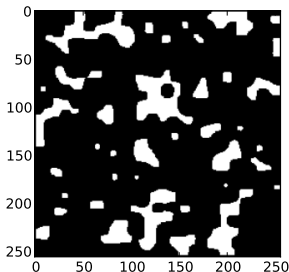
1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from matplotlib import cm
5  im = Image.open('../Slides/figures/
    image.jpg')
6  channels = im.split()
7  z = np.array(channels[0])
8  seuil = 150.
9  zs = z > seuil
10 zse = ndimage.morphology.binary_erosion
    (zs, structure=np.ones((3,3)))
11 zsd = ndimage.morphology.binary_erosion
    (zse, structure=np.ones((3,3)))
12 fig = plt.figure()
13 plt.clf()
14 fig.add_subplot(2, 1, 1)
15 plt.imshow(zss, origin = "upper", cmap
    = cm.gray)
16 plt.colorbar()
17 fig.add_subplot(2, 1, 2)
18 plt.imshow(zse, origin = "upper", cmap
    = cm.gray)
19 plt.colorbar()
20 plt.show()

```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage**
- 8 Contours

## Comptage



```

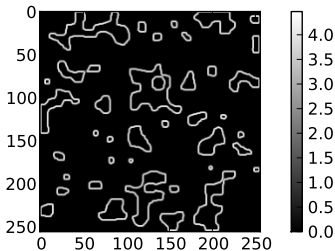
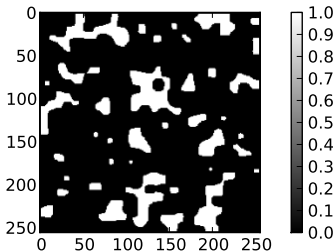
1  from PIL import Image
2  import numpy as np
3  from matplotlib import pyplot as plt
4  from matplotlib import cm
5  im = Image.open('../Slides/figures/
    image.jpg')
6  channels = im.split()
7  z = np.array(channels[0])
8  seuil = 150.
9  zs = z > seuil
10 zse = ndimage.morphology.binary_erosion
    (zs, structure=np.ones((3,3)))
11 zsd = ndimage.morphology.binary_erosion
    (zse, structure=np.ones((3,3)))
12 zl, nombre = ndimage.measurements.label
    (zsd) # On compte les zones
13 zl = np.where(zl == 0, np.nan, zl)
14 fig = plt.figure()
15 plt.clf()
16 fig.add_subplot(2, 1, 1)
17 plt.imshow(zsd, origin = "upper", cmap
    = cm.gray)
18 plt.colorbar()
19 fig.add_subplot(2, 1, 2)
20 plt.imshow(zl, origin = "upper", cmap =
    cm.jet)
21 plt.colorbar()
22 plt.show()

```

# Plan

- 1 Ouverture/Fermeture
- 2 Opérations basiques
- 3 Filtrage
- 4 Histogramme
- 5 Seuillage
- 6 Érosion / Dilatation
- 7 Comptage
- 8 Contours**

## Contours



```

1 from PIL import Image
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from matplotlib import cm
5 from scipy import ndimage
6 im = Image.open('../Slides/figures/image.
      jpg')
7 channels = im.split()
8 z = np.array(channels[0])
9 seuil = 150.
10 zs = z > seuil
11 zse = ndimage.morphology.binary_erosion(zs,
      structure=np.ones((3,3)))
12 zsd = np.float64(ndimage.morphology.
      binary_erosion(zse, structure=np.ones
      ((3,3))))
13 zgx = ndimage.sobel(zsd, axis=0, mode='
      constant')
14 zgy = ndimage.sobel(zsd, axis=1, mode='
      constant')
15 zsob = np.hypot(zgx, zgy)

```