

# Traitement d'images [3EIB2]

Alain Lebret ([alain.lebret@ensicaen.fr](mailto:alain.lebret@ensicaen.fr))

Crédits : le canevas de ce cours est inspiré par celui de E. Agu (Worcester Polytechnic Institute, Massachusetts, USA)  
Toutes les figures sont sous licence Creative Commons

## À propos du cours

### Prérequis

- Bases en Python (nous utilisons [JupyterLab](#) en TP)
- Bases en algèbre linéaire ; transformée Fourier

3

## À propos du cours

Deux parties

### 1. Techniques de traitement d'images

- Cours : 4 h / TP : 18 h
- Intervenant : Alain Lebret ([alain.lebret@ensicaen.fr](mailto:alain.lebret@ensicaen.fr))

### 2. Introduction à l'apprentissage profond pour le traitement d'images

- Cours : 2h / TP : 4h
- Intervenant : Loïc Simon ([loic.simon@ensicaen.fr](mailto:loic.simon@ensicaen.fr))

2

## À propos du cours

### Plan de la partie 1

1. Introduction
2. Couleurs et espaces de couleurs
3. Histogrammes & opérations point à point
4. Filtres spatiaux
5. Filtres fréquentiels
6. Morphologie mathématique
7. Régions

4

## 1. Introduction

### Qu'est-ce qu'une image ?

Nous introduirons aussi les images 3D (mais pas 4D)

Matrice 2D de valeurs d'intensités (niveaux de gris ou couleurs)

Ensemble de valeurs d'intensités

Coordonnées

$$I(u, v) \in \mathbb{P} \text{ et } u, v \in \mathbb{N}$$



$F(x, y)$

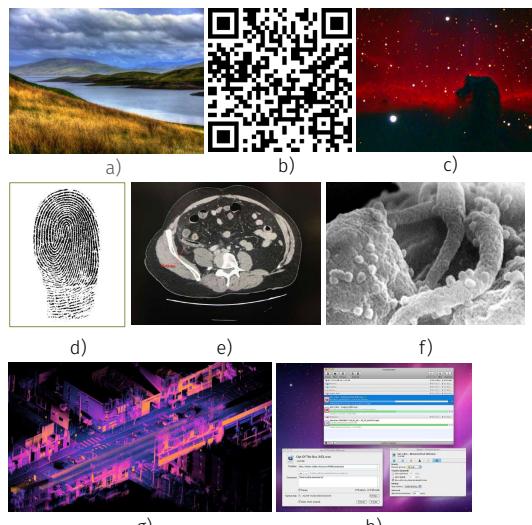
$I(u, v)$

104	104	104	103	103	103	103	102	103	103
105	105	102	104	104	103	103	104	104	103
106	106	106	106	106	106	105	105	105	105
106	107	107	106	106	107	108	107	107	107
108	108	109	109	109	109	110	111	110	111
110	109	107	107	108	109	109	109	108	109
107	107	107	106	109	109	109	107	106	107
106	105	103	103	103	104	103	103	103	104

5

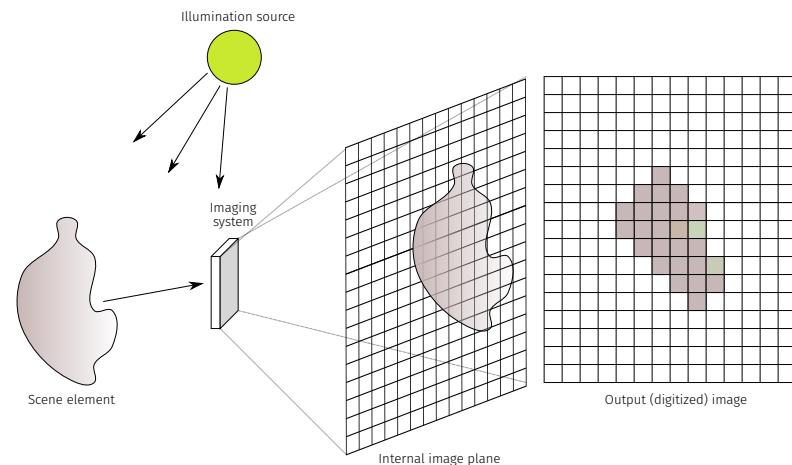
### Exemples d'images numériques

- a) Panorama
- b) QR-code
- c) Objet astronomique
- d) Empreinte
- e) Image scanner
- f) Image de microscope
- g) Image Lidar
- h) Capture d'écran



7

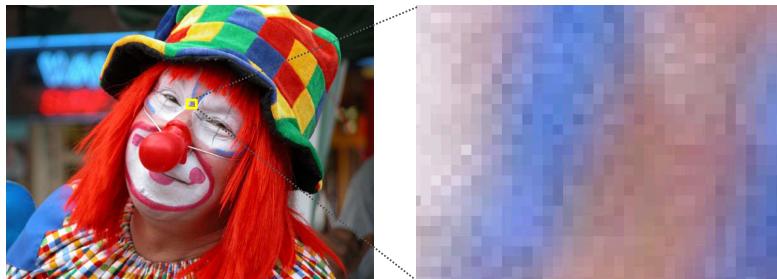
### Système d'imagerie



8

## Image numérique ?

L'image issue d'une numérisation est une approximation d'une scène réelle



9

## Que propose le traitement d'images ?

Algorithmes qui modifient une image et en créent une nouvelle



Les objectifs peuvent être de :

- Éditer une image
- Afficher et imprimer une image
- Améliorer une image
- Compresser une image

11

## Formats d'images numériques

### Formats courants

**1** valeur par point/pixel (N&B ou niveaux de gris)

**3** valeurs par point/pixel (rouge, vert et bleu)

**4** valeurs par point/pixel (rouge, vert, bleu, + "alpha" ou opacité)



Niveaux de gris

RGB

RGBA

10

## Exemples : débruitage

Image bruitée

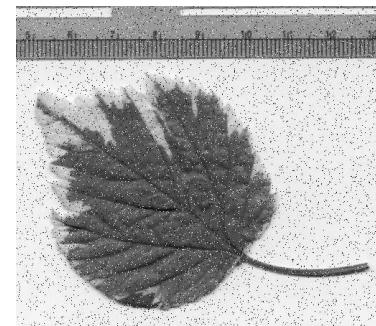


Image débruitée



12

## Exemples : ajustement de contraste

Image faiblement contrastée



Image d'origine



Image fortement contrastée



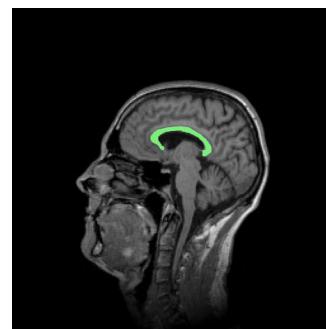
## Exemples : détection de contours



13

14

## Exemples : extraction de régions et segmentation



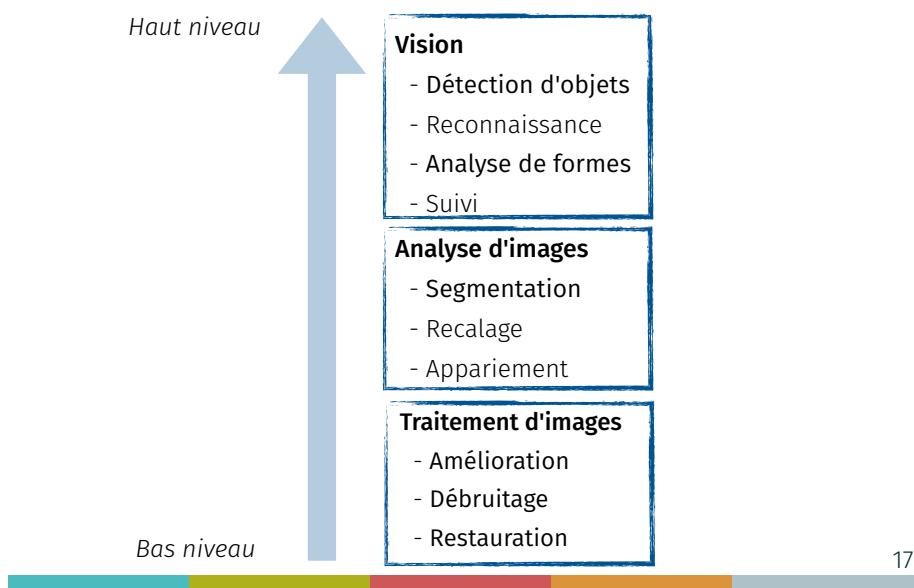
15

## Exemples : restauration (in-painting)

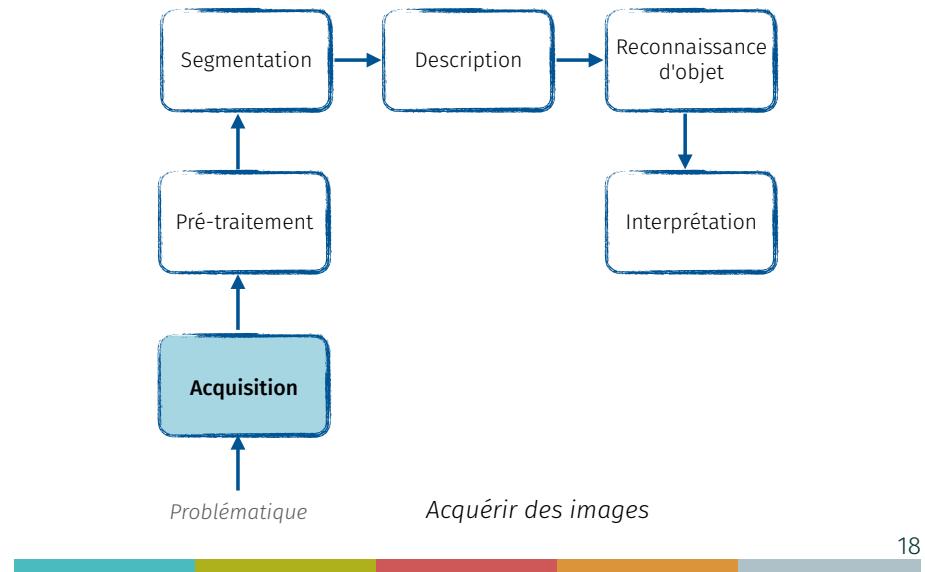


16

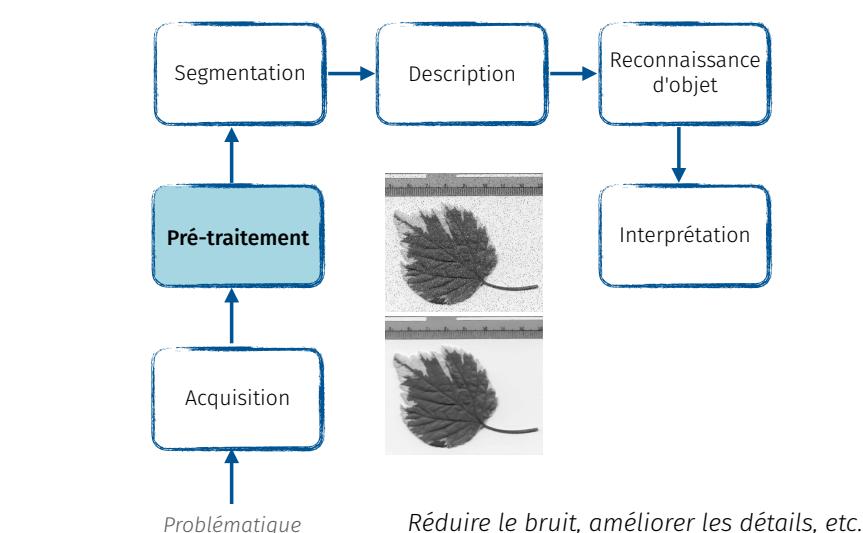
## Relations entre les domaines



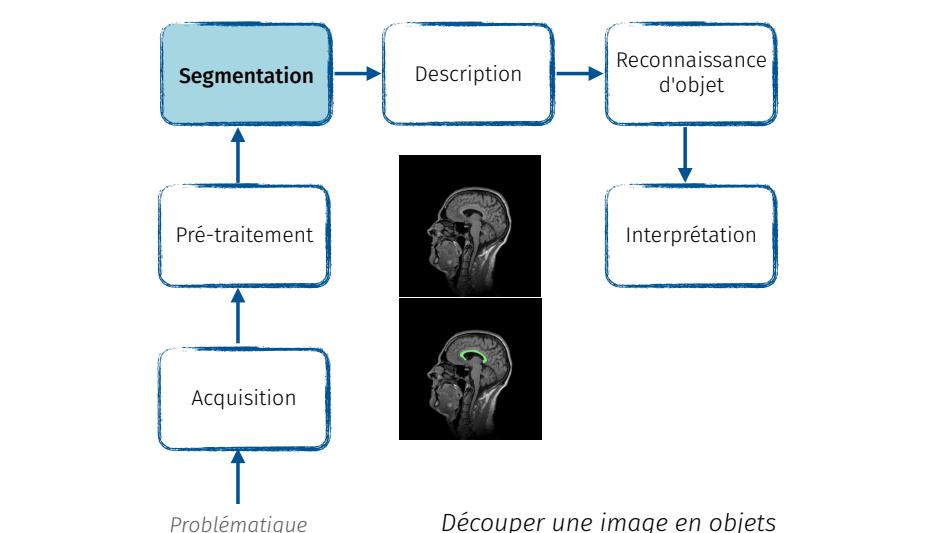
## Étapes clés d'un système de vision



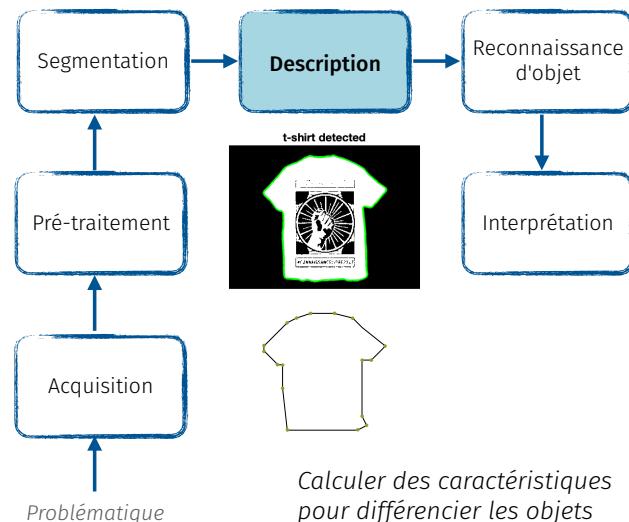
## Étapes clés d'un système de vision



## Étapes clés d'un système de vision

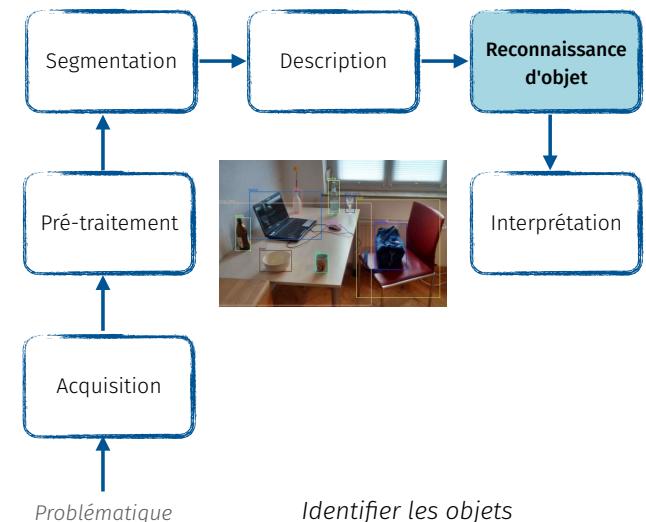


## Étapes clés d'un système de vision



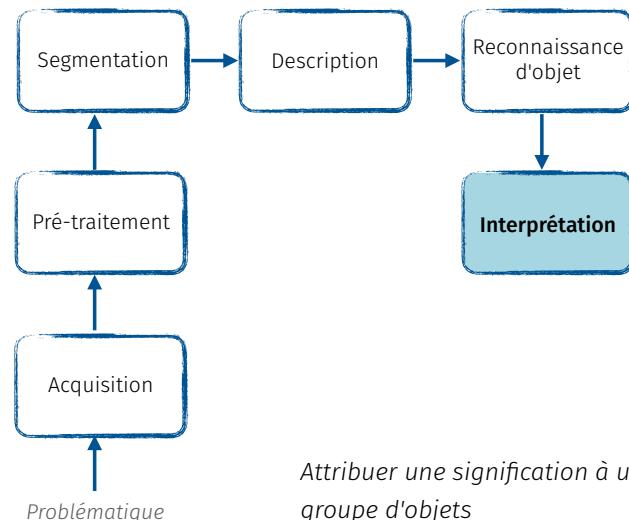
21

## Étapes clés d'un système de vision



22

## Étapes clés d'un système de vision



23

## Formats des fichiers images

### Exemples de formats

- Portable Network Graphics (PNG)
- Graphics Interchange Format (GIF)
- Tagged Image File Format (TIFF)
- Portable Bitmap Format (PBM), JPEG, BMP

La valeur d'un pixel peut être

- **Binaire** : 0 ou 1 (affichée avec 0 ou 255)
- **Niveau de gris** : intervalle 0-255
- **Couleur** : couleurs RGB dans l'intervalle 0-255 pour chaque canal
- **Liée à l'application** (ex. valeurs flottantes en imagerie médicale, astronomie)

24

## Nombre de bits par pixel ?

### Images en niveaux de gris

Composantes	Bits/pixel	Intervalle	Utilisation
1	1	0..1	Binaire : document, fax
1	8	0...255	Universel : photo, numérisation, impression
1	12	0...4095	Haute qualité : photo, numérisation, impression
1	14	0..16383	Professionnel : photo, numérisation, impression
1	16	0..65535	Plus haute qualité : imagerie médicale, astronomie

### Images couleur

Composantes	Bits/pixel	Intervalle	Utilisation
3	24	[0...255] <sup>3</sup>	RGB universel : photo, numérisation, impression
3	36	[0...4095] <sup>3</sup>	RGB haute qualité : photo, numérisation, impression
3	42	[0...16383] <sup>3</sup>	RGB professionnel : photo, numérisation, impression
3	32	[0...255] <sup>4</sup>	CMYK, livres couleurs

25

## Références



26

## Bibliothèques Python pour le traitement d'images

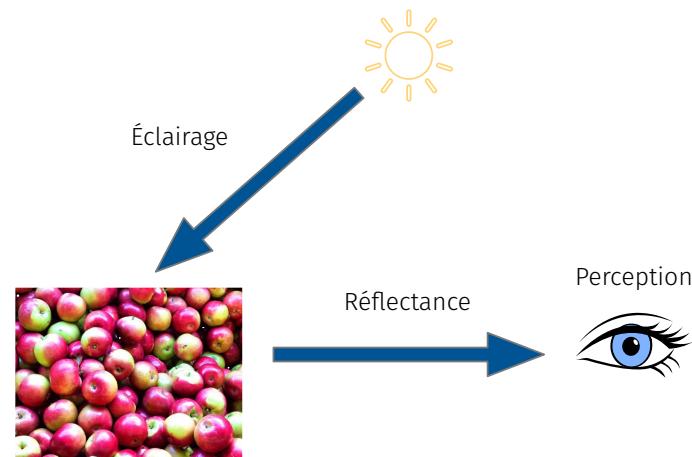
Bibliothèques utilisées sous Jupyter-lab (ou Google Colab)

- [NumPy](#) : algèbre linéaire, transformée de Fourier
- [Pandas](#) : manipulation de données
- [CV2](#) : vision par ordinateur
- [Scikit-image](#) : traitement d'images
- [PIL](#) : traitement d'images
- [Matplotlib](#) : génération de figures et interactions avec l'utilisateur

27

## 2. Couleurs et espaces de couleurs

## Introduction



29

## Qu'est-ce que la couleur ?

La couleur est définie de différentes manières

Pour un physicien

Spectre électromagnétique de l'infrarouge à l'ultraviolet, etc.

Pour un médecin

Impulsion électrique au travers du nerf optique, excitation de certaines molécules photosensibles de l'œil, interprétation par le cerveau, etc.

30

## Qu'est-ce que la couleur ?

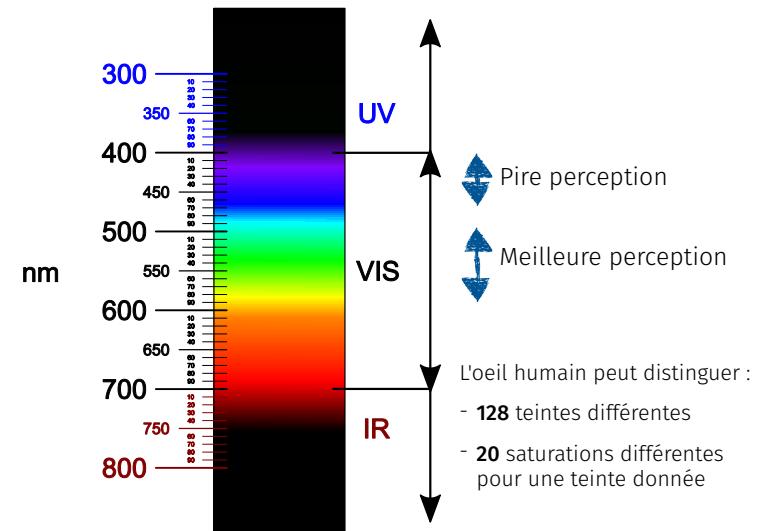
L'informaticien caractérise une couleur par :

- **Teinte (hue)** : longueur d'onde dominante (couleur que nous voyons)
- **Saturation** :
  - Pureté du mélange de longueurs d'onde
  - Distance de la couleur au gris
- **Luminosité (brightness)** : intensité de la lumière
- Teinte, luminance (en  $cd/m^2$ ) et saturation sont utiles pour décrire les couleurs

Le rose est moins saturé que le rouge, et le bleu ciel est moins saturé que le bleu royal

31

## Qu'est-ce que la couleur ?

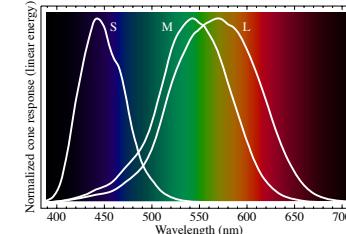


32

## Espaces de couleurs

### Espaces de couleurs

3 types de cônes dans l'oeil humain suggèrent que la couleur est une quantité 3D



Il existe différents espaces de couleurs :

RGB, CMY, HSL, HSV, etc.

33

34

## Espaces de couleurs additifs ou soustractifs



Les espaces de couleurs sont **additifs** ou **soustractifs**

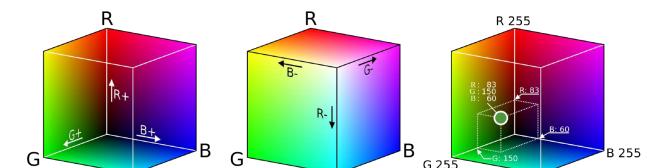
Obtenus en soustrayant les composantes du blanc

35

## Espace de couleurs RGB

Définit les couleurs avec les quantités ( $r, g, b$ ) de rouge, vert, bleu

- Le plus populaire des espaces additifs
- Valeur maximale = **255** / **1.0** si normalisée
- Noir =  $(0, 0, 0)$  / Blanc =  $(255, 255, 255)$
- Quantités égales de  $(r, g, b)$  = gris (diagonale blanc-noir)



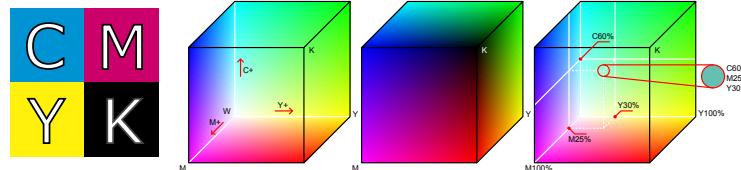
La plupart des opérations adaptées aux images à niveaux de gris le seront sur les images couleur en les appliquant à chaque composante

36

## Espace de couleurs CMYK (quadrichromie)

Définit les couleurs avec les quantités ( $c, m, y$ ) de cyan, magenta, et jaune

- C'est un espace soustractif utilisé dans l'imprimerie
- Parfois, le noir (K) est utilisé pour l'obtention d'un noir plus profond
- ( $c, m, y$ ) signifie soustraire le complément à C (rouge), M (vert), et Y (bleu)

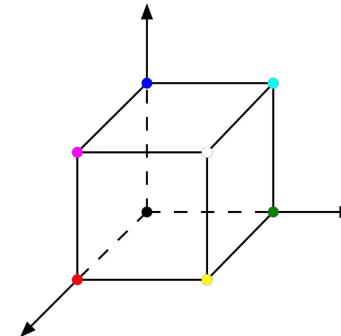


37

## Relation entre RGB et CMYK

On peut placer RGB et CMY dans le même volume cubique

- R, G, B, C, M et Y sont aux sommets



38

## RGB ou CMYK



RGB

CMYK

39

## Espace de couleurs HSL (TSL)

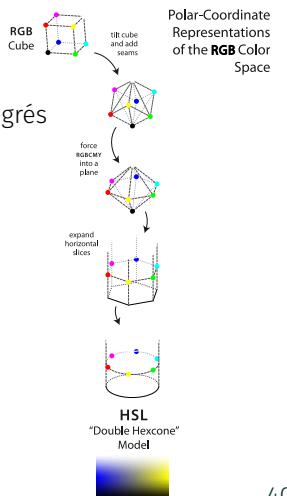
HSL signifie hue (teinte), saturation et lightness (luminosité)

- Basé sur la déformation du cube RVB
  - Les teintes sont placées sur un hexagone
  - La teinte est exprimée comme un angle en degrés
  - 0 degré : rouge



RGB

HSL



40

## Espace de couleurs HSV (TSV)

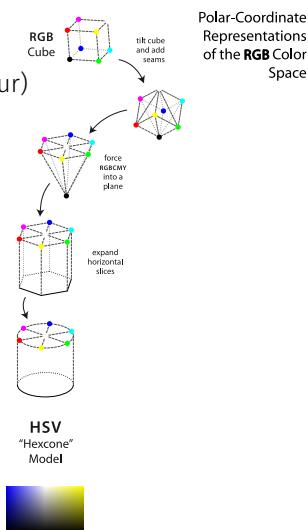
HSV signifie hue (teinte), saturation et value (valeur)

- Espace similaire à HLS au niveau conceptuel



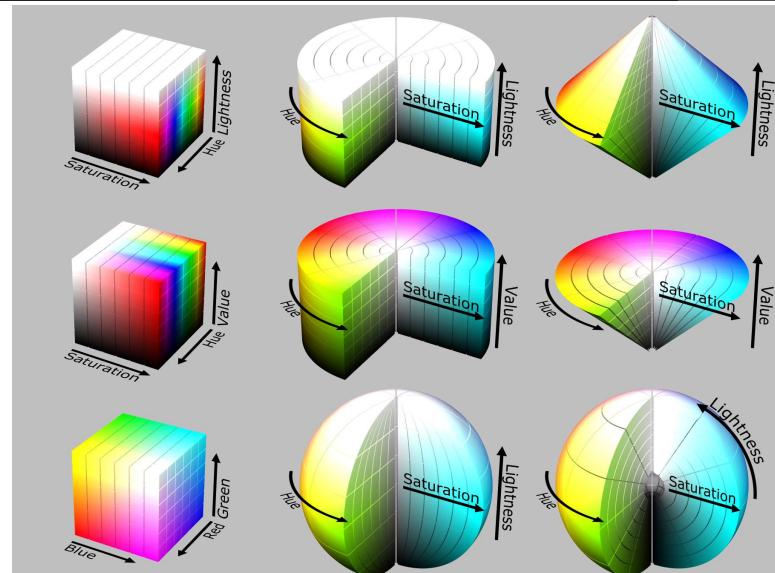
RGB

HSV



41

## Représentations RGB, HSV et HSL



42

## Espaces de couleurs télévisuels : YUV, YIQ, YCbCr

- YUV et YIQ : télévision analogique NTSC ou PAL

- YCbCr : télévision numérique

### Principe

- Séparation de la composante Y (luminance) et des 2 composantes de chrominance (H, S)
- Les différences de couleur entre composantes sont encodées



RGB

YUV

YIQ

YCbCr

43

## Espace de couleurs CIE

La CIE (Commission Internationale d'Éclairage) a proposé en 1931 l'utilisation des valeurs X, Y, et Z associées aux spectres suivants :

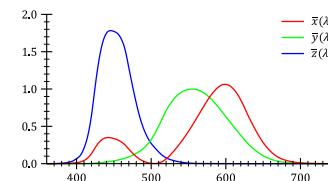
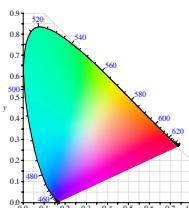


Table des valeurs XYZ associées à toutes les couleurs visibles



Voir [https://fr.wikipedia.org/wiki/CIE\\_XYZ](https://fr.wikipedia.org/wiki/CIE_XYZ)

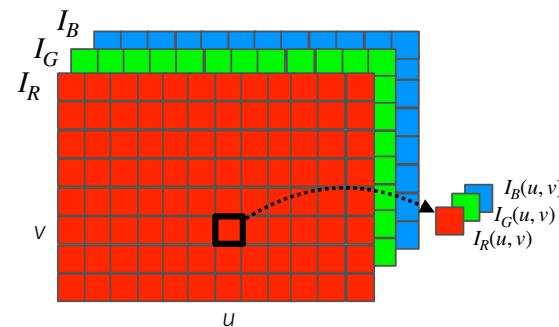
44

## Quelques mots de plus sur les images RGB

45

### Images en vraies couleurs

- Composantes contenues dans 3 tableaux distincts
- Récupère la même position  $(u, v)$  pour chaque tableau R, G et B



47

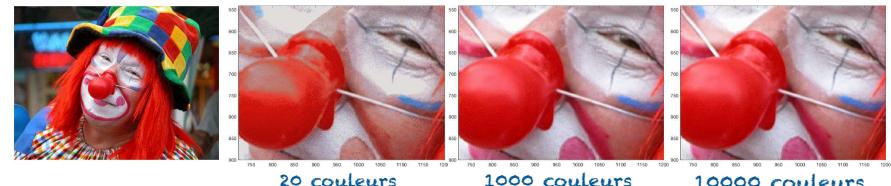
### Organisation des images couleurs

#### Vraies couleurs

- Utilise tout le spectre de couleurs de l'espace
- Utilisées dans les applications nécessitant la mise en évidence de différences subtiles (photographie numérique, rendu réaliste, etc.)

#### Couleurs indexées

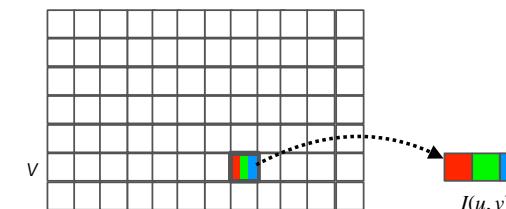
- Utilise un sous-ensemble de couleurs dépendant de l'application



46

### Images en vraies couleurs

Les composantes RGB représentant la couleur d'un pixel sont stockées dans un même élément : un entier

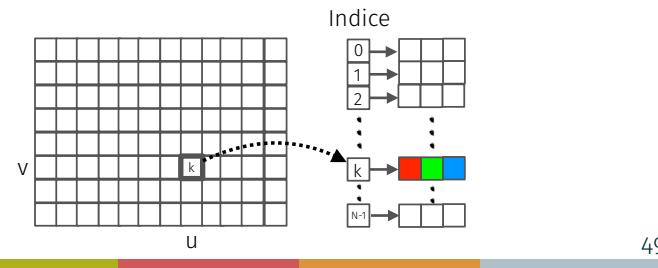


Un entier de 4 octets est utilisé pour le stockage d'une couleur

48

## Image en couleurs indexées

- Nombre limité de couleurs distinctes ( $N = 2$  à 256)
- Utilisée pour les images contenant de larges régions de couleurs identiques
- L'image contient les indices d'une table des couleurs ou d'une palette
- La palette est enregistrée comme une partie de l'image
- La conversion de vraies couleurs à couleurs indexées nécessite une quantification



## Stratégies de traitement des images couleurs

### Stratégie 1

Traiter séparément chaque composante RGB

### Stratégie 2

Calculer la luminance, puis traiter la matrice des intensités

$$Y = w_R R + w_G G + w_B B$$

50

## Conversion entre espaces de couleurs

51

## Convertir entre espaces de couleurs

La conversion entre des espaces de couleurs peut être vue comme une transformation matricielle :

$$[X' \ Y' \ Z'] = [X \ Y \ Z] \begin{bmatrix} c_{11} & c_{21} & c_{31} \\ c_{12} & c_{22} & c_{32} \\ c_{13} & c_{23} & c_{33} \end{bmatrix}$$

52

## Conversion en niveaux de gris

Conversion couleur à niveaux de gris la plus simple

$$\text{Luminance } Y = \frac{R + G + B}{3}$$

L'image résultante peut être trop sombre dans ses composantes rouge et verte

Une meilleure approche consiste à pondérer les composantes :

$$Y = w_R R + w_G G + w_B B$$

Poids utilisés :

$$w_R = 0.2125, \quad w_G = 0.7154, \quad w_B = 0.072$$

3 composantes



1 composante



53

## Désaturation des images couleurs

La désaturation est une réduction uniforme de la quantité de R, G, B

- La couleur désaturée est obtenue par interpolation linéaire de la couleur RGB et du point en niveau de gris correspondant (Y, Y, Y) dans l'espace RGB

$$\begin{pmatrix} R_d \\ G_d \\ B_d \end{pmatrix} = \begin{pmatrix} Y \\ Y \\ Y \end{pmatrix} + s_{\text{couleur}} \cdot \begin{pmatrix} R - Y \\ G - Y \\ B - Y \end{pmatrix}$$

valeur dans l'intervalle [0, 1]



55

## Une image couleur qui apparaît en gris

Une image RGB image est **sans teinte** ou grise lorsque toutes ses composantes sont égales :

$$R = G = B$$

Pour supprimer la couleur :

1. Calculer la luminance Y à partir de la somme pondérée précédente
2. Remplacer R, G et B par la valeur de Y

3 composantes



$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} Y \\ Y \\ Y \end{pmatrix}$$

3 composantes



54

## Conversion RGB vers HSL

Calculer la teinte de la même manière que pour le modèle HSV :

$$H_{HSL} = H_{HSV}$$

Puis calculer :

$$L_{HSL} = \frac{(C_{\text{high}} + C_{\text{low}})/255}{2}$$

$$S_{HSL} = \begin{cases} 0 & \text{pour } L_{HSL} = 0 \\ 0.5 \cdot \frac{C_{\text{intervalle}}/255}{L_{HSL}} & \text{pour } 0 < L_{HSL} \leq 0.5 \\ 0.5 \cdot \frac{C_{\text{intervalle}}/255}{1 - L_{HSL}} & \text{pour } 0.5 < L_{HSL} < 1 \\ 0 & \text{pour } L_{HSL} = 1 \end{cases}$$

56

## Conversion RGB vers HSL



Image RGB



HSL



H



S



L

57

## Conversion HSL vers RGB

En supposant H, L et S dans l'intervalle [0, 1]

$$(R', G', B') = \begin{cases} (0,0,0) & \text{pour } L_{HSL} = 0 \\ (1,1,1) & \text{pour } L_{HSL} = 1 \end{cases}$$

Puis calculer :

$$H' = (6H_{HSL}) \bmod 6$$

Enfin, calculer :

$$c_1 = H' \quad c_2 = H' - c_1$$

$$x = L_{HSL} - d \quad w = L_{HSL} + d$$

$$y = w - (w - x) \cdot c_2$$

$$z = w + (w - x) \cdot c_2$$

$$d = \begin{cases} S_{HSL} \cdot L_{HSL} & \text{pour } L_{HSL} \leq 0.5 \\ S_{HSL} \cdot (1 - L_{HSL}) & \text{pour } L_{HSL} > 0.5 \end{cases}$$

58

## Conversion HSL vers RGB

Alors, les composantes RGB sont données par :

$$(R', G', B') = \begin{cases} (w, z, x) & \text{si } c_1 = 0 \\ (y, w, x) & \text{si } c_1 = 1 \\ (x, w, z) & \text{si } c_1 = 2 \\ (x, y, w) & \text{si } c_1 = 3 \\ (z, x, w) & \text{si } c_1 = 4 \\ (w, x, y) & \text{si } c_1 = 5. \end{cases}$$

59

## Conversion RGB vers HSV

Soit les valeurs suivantes :

$$C_{\text{high}} = \max(R, G, B)$$

$$C_{\text{low}} = \min(R, G, B)$$

$$C_{\text{range}} = C_{\text{high}} - C_{\text{low}}$$

Trouver la saturation pour les composantes RGB ( $C_{\text{max}} = 255$ ):

$$S_{HSV} = \begin{cases} \frac{C_{\text{range}}}{C_{\text{high}}} & \text{pour } C_{\text{high}} > 0 \\ 0 & \text{sinon} \end{cases}$$

Alors la luminance est :

$$V_{HSV} = \frac{C_{\text{high}}}{C_{\text{max}}}$$

60

## Conversion RGB vers HSV

Normaliser chaque composante :

$$R' = \frac{C_{\text{high}} - R}{C_{\text{range}}} \quad G' = \frac{C_{\text{high}} - G}{C_{\text{range}}} \quad B' = \frac{C_{\text{high}} - B}{C_{\text{range}}}$$

Calculer la teinte préliminaire  $H'$  :

$$H' = \begin{cases} B' - G' & \text{si } R = C_{\text{high}} \\ R' - B' + 2 & \text{si } G = C_{\text{high}} \\ G' - R' + 4 & \text{si } B = C_{\text{high}} \end{cases}$$

Finalement, la teinte est obtenue en normalisant sur l'intervalle [0, 1] :

$$H_{HSV} = \frac{1}{6} \cdot \begin{cases} H' + 6 & \text{pour } H' < 0 \\ H' & \text{sinon} \end{cases}$$

61

## Conversion HSV vers RGB

Tout d'abord, calculons :

$$H' = (6H_{HSV}) \bmod 6$$

Puis calculons les valeurs intermédiaires suivantes :

$$\begin{aligned} c_1 &= H' & x &= (1 - S_{HSV}) \cdot V_{HSV} \\ c_2 &= H' - c_1 & y &= (1 - (S_{HSV} \cdot c_2)) \cdot V_{HSV} \\ & & z &= (1 - (S_{HSV} \cdot (1 - c_2))) \cdot V_{HSV} \end{aligned}$$

Normalisons les valeurs RGB dans l'intervalle [0, 1] :

$$(R', G', B') = \begin{cases} (V_{HSV}, z, x) & \text{si } c_1 = 0 \\ (y, V_{HSV}, x) & \text{si } c_1 = 1 \\ (x, V_{HSV}, z) & \text{si } c_1 = 2 \\ (x, y, V_{HSV}) & \text{si } c_1 = 3 \\ (z, x, V_{HSV}) & \text{si } c_1 = 4 \\ (V_{HSV}, x, y) & \text{si } c_1 = 5. \end{cases}$$

63

## Conversion RGB vers HSV



Image RGB



HSV



H



S



V

62

## Conversion HSV vers RGB

Finalement, les composantes RGB sont obtenues ainsi :

$$\begin{aligned} R &= \min(\text{round}(255R'), 255) \\ G &= \min(\text{round}(255.G'), 255) \\ B &= \min(\text{round}(255.B'), 255) \end{aligned}$$

64

## Espace de couleurs YUV

- Encodage des couleurs en TV analogique en Amérique du nord (NTSC) et en Europe (PAL)

- La composante Y est calculée à partir des composantes RGB :

$$Y = 0.299.R + 0.587.G + 0.114.B$$

- Les composantes U et V sont données par :

$$U = 0.492.(B - Y) \quad V = 0.877.(R - Y)$$



## Espace de couleurs YUV



Image RGB



YUV



Y



U



V

66

## Espace de couleurs YUV

Transformation de RGB à YUV

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Transformation de YUV à RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$



## Espace de couleurs YIQ

NTSC utilise une variante de YUV : YIQ

- La composante Y est la même que pour YUV
- Les composantes U et V subissent la transformation suivante :

$$\begin{pmatrix} I \\ Q \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \cos(\beta) & \sin(\beta) \\ -\sin(\beta) & \cos(\beta) \end{pmatrix} \cdot \begin{pmatrix} U \\ V \end{pmatrix}$$

$$\beta = 0.576 \text{ (33 degrés)}$$

68

## Espace de couleurs YIQ



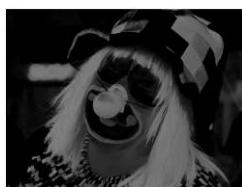
Image RGB



YIQ



Y



I



Q

69



Image RGB



YCbCr



Y



Cb



Cr

71

## Espace de couleurs YCbCr

- Variante normalisée de l'espace YUV

- Utilisation : TV numérique, compression d'image (e.g. JPEG)

Les composantes  $Y$ ,  $C_b$ ,  $C_r$  sont calculées de la manière suivante :

$$Y = w_R \cdot R + (1 - w_B - w_R) \cdot G + w_B \cdot B$$

$$C_b = \frac{0.5}{1 - w_B} \cdot (B - Y)$$

$$C_r = \frac{0.5}{1 - w_R} \cdot (R - Y)$$

ITU préconise :

$$w_R = 0.299, \quad w_B = 0.114, \quad w_G = 1 - w_B - w_R = 0.587$$

70

## Espace de couleurs YCbCr

Transformation YCbCr vers RGB

$$R = Y + \frac{1 - w_R}{0.5}$$

$$G = Y - \frac{w_B \cdot (1 - w_B) \cdot C_b - w_R \cdot (1 - w_R) \cdot C_r}{0.5 \cdot (1 - w_B - w_R)}$$

$$B = Y + \frac{1 - w_B}{0.5} \cdot C_b$$

72

## Espace de couleurs YCbCr

Transformation RGB vers YCbCr

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Transformation YCbCr vers RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.403 \\ 1 & -0.344 & -0.714 \\ 1 & 1.773 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix}$$

73

## Comparaison entre RGB, HSV, HSL, YUV, YIQ et YCbCr



74

## Couleurs, espaces couleurs et Python

75

## Scikit-image

La bibliothèque Python [scikit-image](#) supporte 2 types d'images couleur

- Images RGB en vraies couleurs (24 bits)

- Formats TIFF, BMP, JPEG, PNG et RAW

- Images indexées (8 bits)

- Jusqu'à 256 couleurs max.

- Formats GIF, PNG, BMP et TIFF (non compressé)

Les modules `io` et `color` de [scikit-image](#) fournissent des opérateurs de conversion entre différents formats couleurs et niveaux de gris

Remarque : la bibliothèque `cv2` (OpenCV) propose aussi des opérateurs de conversion

76

### 3. Histogrammes et opérateurs point à point

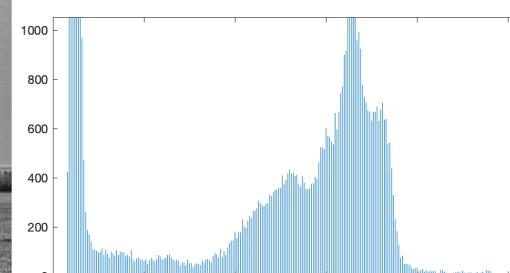


### Histogrammes

78

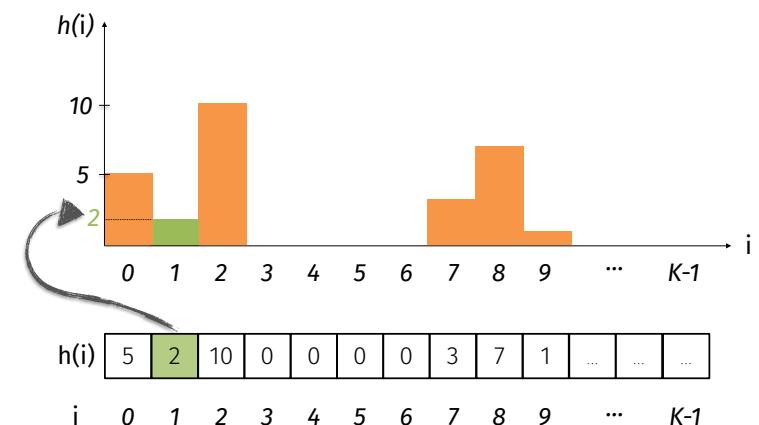
#### Histogramme

Un histogramme affiche la fréquence des valeurs d'intensité



79

#### Histogramme



80

## Histogramme



Des images différentes peuvent avoir le même histogramme



81

## Histogramme

Certains défauts d'une image peuvent être mis en évidence par l'histogramme :

- Sur- et sous-exposition
- Contraste
- Dynamique

82

## Luminosité

La luminosité d'une image en niveaux de gris est la moyenne de l'intensité de tous ses pixels

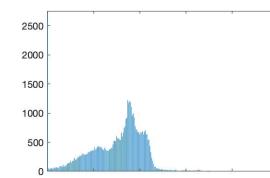
$$B(I) = \frac{1}{LH} \sum_{v=1}^H \sum_{u=1}^L I(u, v)$$

Nombre de pixels

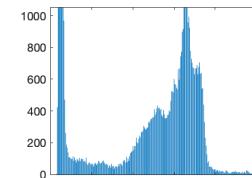
Somme des intensités

83

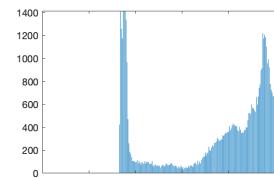
## Histogramme et détection d'une mauvaise exposition



sous-exposée



exposition correcte



surexposée

84

## Contraste

Le contraste d'une image en niveaux de gris renseigne sur la facilité avec laquelle les objets dans l'image peuvent être distingués

Contraste élevé : de nombreuses intensités distinctes

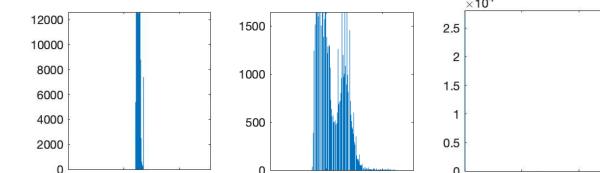
Contraste faible : peu d'intensités distinctes

Un bon contraste correspond à une large gamme d'intensités et une grande différence entre les intensités min et max



85

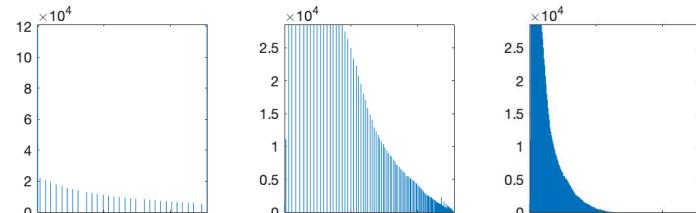
## Histogramme et contraste d'une image



Très faible contraste      Contraste normal      Contraste trop élevé

86

## Histogramme et dynamique d'une image



Très faible dynamique

Faible dynamique

Grande dynamique



87

## Histogramme et dynamique d'une image

Une grande dynamique signifie des parties très claires et très sombres dans l'image (nombreuses intensités distinctes)

La dynamique peut dépasser le nombre de bits disponibles permettant de représenter les pixels d'une image

Solution

- 1) Capturer plusieurs images à différentes expositions
- 2) Combiner celles-ci en utilisant des techniques de traitement d'images

88

## Histogrammes d'images hautes résolution

- Une image haute résolution peut aboutir à un histogramme trop grand et donc difficile à afficher
- Exemple : image 32-bits =  $2^{32} = 4\ 294\ 967\ 296$  colonnes

### Solution

Regrouper des plages d'intensités dans chaque colonne de l'histogramme (on parlera de *bins*)

89

## Comment calculer le nombre de regroupements

$$\text{taille bin} = \frac{\text{nombre de valeurs distinctes dans l'image}}{\text{nombre de bins}}$$

Exemple: créer 256 bins depuis une image 14-bits

$$\text{taille bin} = \frac{2^{14}}{256} = 2^{(14-8)} = 2^6 = 64$$

$$\begin{aligned} h(0) &\leftarrow 0 \leq I(u, v) < 64 \\ h(1) &\leftarrow 64 \leq I(u, v) < 128 \end{aligned}$$

⋮

⋮

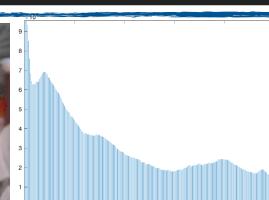
$$h(255) \leftarrow 16320 \leq I(u, v) < 16384$$

90

## Histogramme d'une image couleur

1

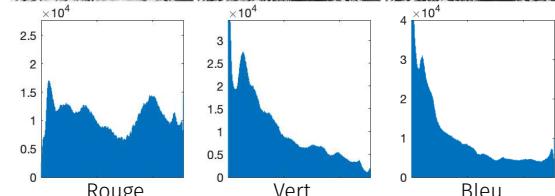
Histogramme d'intensités (\*)



(\*) image préalablement convertie en gris

2

Histogrammes pour chaque composante



91

## Opérateurs point à point

92

## Opérateurs point à point

Un opérateur point à point applique une fonction identique sur chaque pixel

$$I'(u, v) = f(I(u, v))$$

### Exemples

- Addition (change la luminosité)
- Multiplication (étire ou contracte l'intervalle de contraste)
- Fonctions  $\exp(x)$ ,  $\log(x)$ ,  $x^k$ ,  $(1/x)$ , etc.
- Correction gamma
- Seuillage global



Nécessité de circonscrire les valeurs  
dans l'intervalle d'origine

93

## Opérateurs courants

## Transformations d'images en niveaux de gris

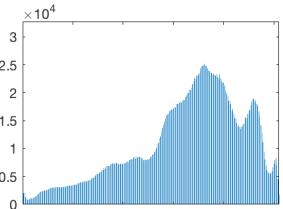
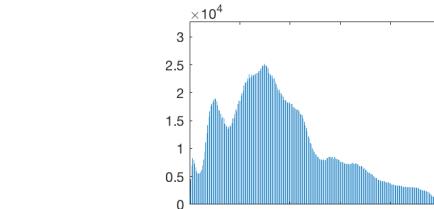
- Linéaire (identité/négatif)
- Logarithmique (Log / Log inverse)
- Puissance (puissance  $n^{ieme}$  / racine  $n^{ieme}$ )

95



## Négatif d'une image

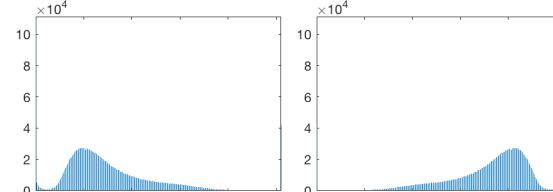
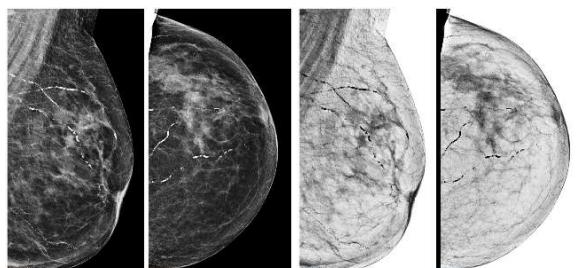
$$f_{invert}(I) = -I + Cste$$



96

## Négatif d'une image

Améliorer les détails clairs dans les zones sombres d'une image



97

## Transformation logarithmique

Étirer les plages de valeurs étroites

$$I'(u, v) = C \times \log(1 + I(u, v))$$



98

## Transformation à l'aide d'une fonction puissance

Étirer ou contracter les plages de valeurs sombres

$$I'(u, v) = Cste \times I^r(u, v)$$



0.5

0.7

0.9



1

2

3

99

## Ajustement de contraste automatique

Modifier les intensités de manière à "couvrir" l'intégralité des valeurs possibles

$$f_{aca}(I) = I_{\min} + (I - I_{\text{low}}) \times \frac{I_{\max} - I_{\min}}{I_{\text{high}} - I_{\text{low}}}$$



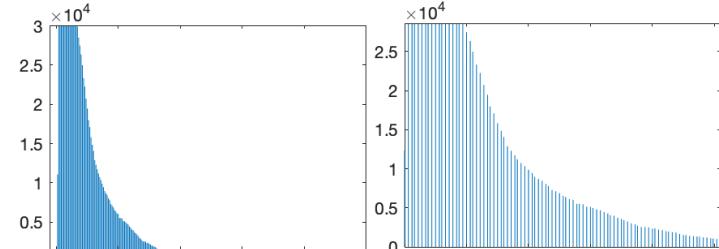
100

## Ajustement de contraste automatique

Original



Automatic histogram adjustment



101

## Seuillage global

## Seuillage

$$f_{seuil}(I) = \begin{cases} I_0 & \text{pour } I < I_{seuil} \\ I_1 & \text{pour } I \geq I_{seuil} \end{cases}$$

### Exemple

Convertir en image binaire une image en niveaux de gris si :

-  $I_0 = 0$

-  $I_1 = 1$

## Exemple de seuillage

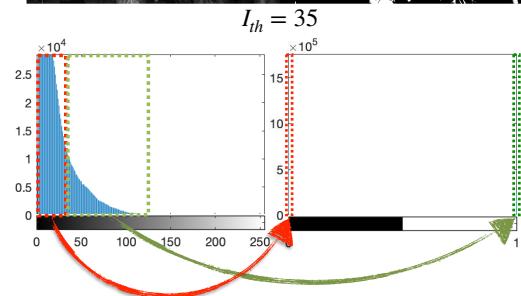


$I_{th} = 35$

103

104

## Histogrammes et seuillage

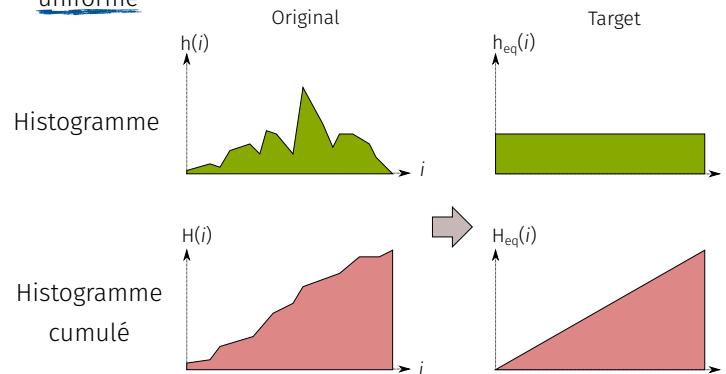


105

## Égalisation d'histogramme

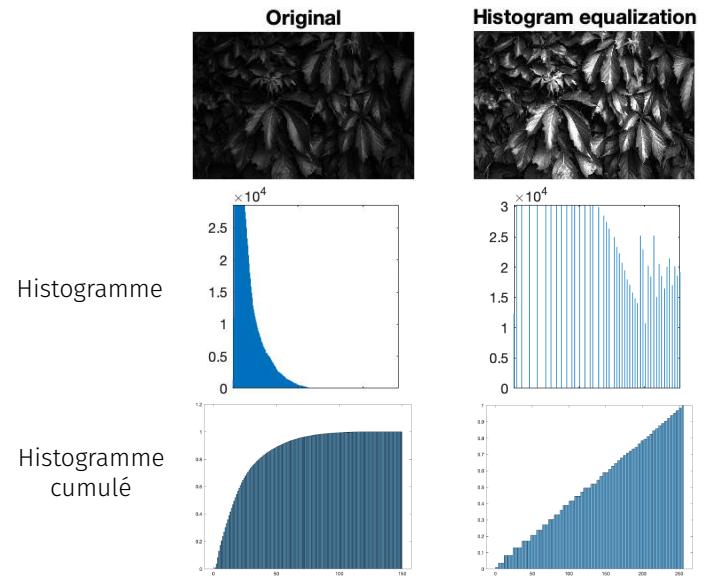
### Égalisation d'histogramme

- Ajuster 2 images differentes afin de rendre leurs histogrammes semblables
- Changer l'histogramme de l'image modifiée en une distribution uniforme



107

### Égalisation d'histogramme



108

## Égalisation d'histogramme

### Fonction histogramme

```
def get_histogram(image, bins):
    # array with size of bins, set to zeros
    histogram = np.zeros(bins)

    # loop through pixels and sum up counts
    # of pixels
    for pixel in image:
        histogram[pixel] += 1

    # return our final result
    return histogram
```

### Fonction somme cumulée

```
def cumsum(a):
    a = iter(a)
    b = [next(a)]
    for i in a:
        b.append(b[-1] + i)
    return np.array(b)
```

```
# execute histogram function
hist = get_histogram(img, 256)

# execute the cumulative sum fn
cs = cumsum(hist)

# display the result
plt.plot(cs)



### Normalisation


# numerator & denominator
nj = (cs - cs.min()) * 255
N = cs.max() - cs.min()

# re-normalize the cumsum
cs = nj / N

# cast it back to uint8 since we can't use
# floating point values in images
cs = cs.astype('uint8')

plt.plot(cs)
```



109

## Égalisation d'histogramme

```
# get the value from cumulative sum for
# every index, and set that as img_new
img_new = cs[img]

# put array back into original shape since
# we flattened it
img_new = np.reshape(img_new, img.shape)

# set up side-by-side image display
fig = plt.figure()
fig.set_figheight(15)
fig.set_figwidth(15)
fig.add_subplot(1,2,1)
plt.imshow(img, cmap='gray')

# display the new image
fig.add_subplot(1,2,2)
plt.imshow(img_new, cmap='gray')
plt.show(block=True)
```



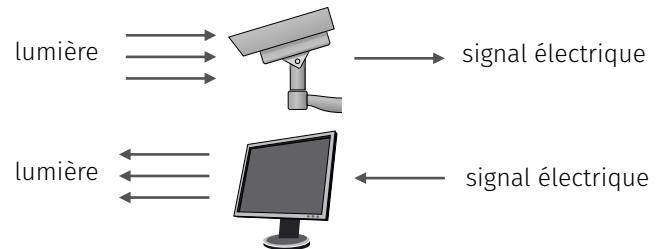
110

## Correction gamma



111

## Correction gamma



### Différents capteurs de caméra

- Ont différentes réponses face à une intensité lumineuse
- Produisent différents signaux électriques pour une même entrée

### Comment s'assurer de la cohérence

- Des images enregistrées par différentes caméras pour une entrée lumineuse donnée ?
- De la lumière émise par différents écrans pour la même image ?

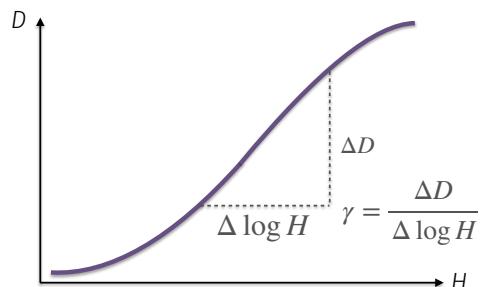


112

## Correction gamma

Fonction d'exposition : relation entre le logarithme de l'intensité lumineuse reçue ( $H$ ) et de la densité du film ( $D$ )

Gamma : pente de la partie linéaire de la courbe



113

## Correction gamma

- La fonction gamma est une bonne approximation de la courbe d'exposition

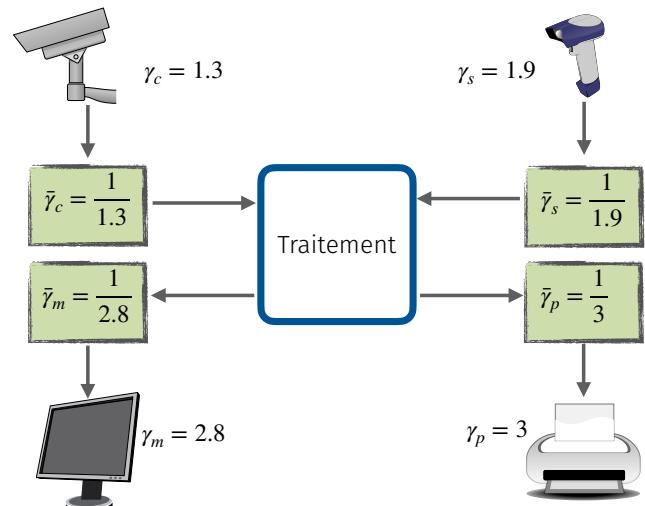
- L'inverse d'une fonction gamma est une fonction gamma telle que :

$$\bar{\gamma} = \frac{1}{\gamma}$$

- Pour les écrans à tubes cathodiques et à leds, les gammes sont dans l'intervalle 1.8 - 2.8 (typiquement 2.4)

114

## Correction gamma



115

## Correction gamma en Python

```
def load_image(filename, width, gamma):  
    img = imageio.imread(filename)  
  
    if img.shape[-1] == 4:  
        # Blend the alpha channel  
        img = color.rgb2rgb(img)  
  
    # Convert to grayscale  
    img = color.rgb2gray(img)  
  
    # Resample and adjust the aspect ratio  
    width_px = (3 * width) * 16  
  
    img_width = 1.0 * width_px  
    img_height = int(img.shape[0] * 3 * (img_width / (4 * img.shape[1])))  
    img = transform.resize(img, (img_height, img_width), anti_aliasing=True,  
    mode='constant')  
  
    # Adjust the exposure  
    img = exposure.adjust_gamma(img, gamma)  
  
    return img
```

116

## *Histogrammes et opérateurs point à point en Python*

### **Histogrammes et opérateurs point à point en Python**

Le module `exposure` de la bibliothèque [scikit-image](#) propose des fonctions de correction gamma et de manipulation des histogrammes

117

118

## **4. Filtres spatiaux**

### **Qu'est-ce qu'un filtre ?**

Les opérations basées pixels sont limitées

Les filtres combinent une opération sur la valeur d'un pixel et la valeur de ses voisins

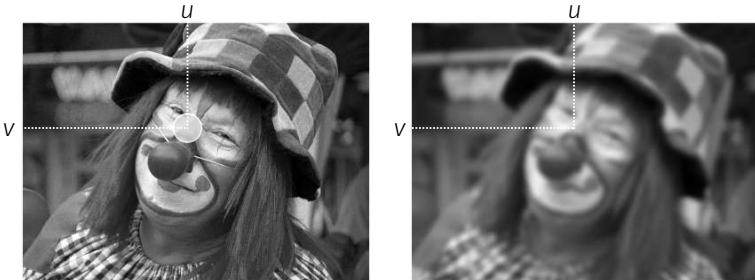
120

## Filtre spatial

### Définition

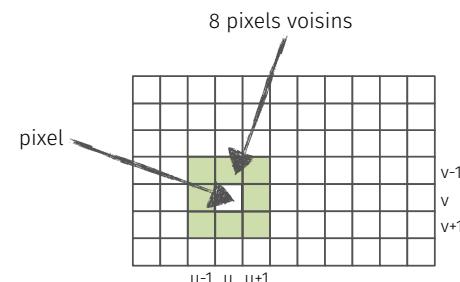
Opération qui combine l'intensité  $I(u, v)$  de chaque pixel avec celles de ses voisins

Ex. : le **lissage** calcule la moyenne ou la moyenne pondérée d'un groupe de pixels



121

## Exemple : lissage par moyenne d'un voisinage 3x3



Le **lissage** remplace l'intensité de chaque pixel par la moyenne de son intensité et de celle de ses voisins

122

## Lissage par moyenne

On peut utiliser différents paramètres pour les filtres (taille, poids, fonction, etc.) :

- Taille du voisinage : 3x3, 5x5, 7x7, ..., 21x21, ...
- Forme : carré, rectangle, disque, etc.
- Poids : permet de pondérer l'importance des différents pixels
- Fonction : peut être linéaire (somme pondérée) ou non linéaire (médian)

Exemple avec différentes tailles de masques, de forme carrée et utilisant une sommation équipondérée



Image d'Origine

3x3

41x41

83x83

123

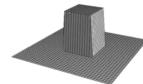
## Filtres linéaires

124

## Filtres linéaires

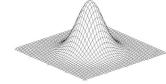
2 classes principales de filtres linéaires :

- Filtres de lissage : poids positifs (ex. : boite, gaussien)
- Filtres différentiels : poids positifs et négatifs (ex. : Laplacien)



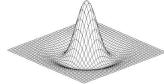
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Boite



0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Gaussien

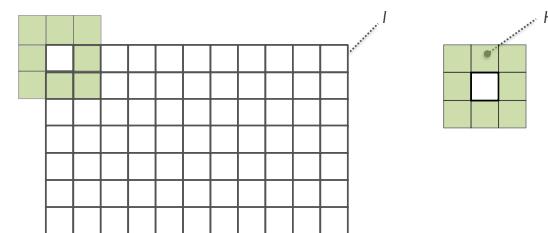


0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Laplacien



## Appliquer un filtre linéaire par convolution



$$I'(u, v) = \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u + i, v + j) \cdot H(i, j)$$

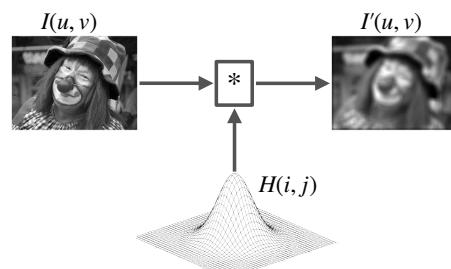


## Propriétés de la convolution

- L'application d'un filtre est appelée convolution linéaire
- Pour un signal 2D, la convolution est définie par :

$$I'(u, v) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} I(u + i, v + j) \cdot H(i, j)$$

$$I' = I * H$$



## Propriétés de la convolution

- Commutativité

$$I * H = H * I$$

Résultat identique si nous convoluons l'image avec le filtre ou vice-versa

- Associativité

$$A * (B * C) = (A * B) * C$$

Résultat identique quel que soit l'ordre d'application des filtres

- Linéarité

$$(s \cdot I) * H = I * (s \cdot H) = s \cdot (I * H)$$

Résultat identique pour la multiplication par un scalaire d'une image ou de sa convoluée

$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$

Résultat identique pour la somme de 2 images convoluée avec un noyau H et 2 images convoluées par H, puis additionnées

- Séparabilité

$$H = H_1 * H_2 * \dots * H_n$$

$$I * H = I * (H_1 * H_2 * \dots * H_n) = (((I * H_1) * H_2) * \dots * H_n)$$

Les petits noyaux sont moins gourmands en calcul



## Séparabilité en x et y

Le noyau d'un filtre peut quelquefois être séparé en ses composantes "verticale" et "horizontale"

Soit  $H_x = [1 \ 1 \ 1 \ 1 \ 1]$ , et  $H_y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Alors  $H = H_x * H_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

Complexité :  $15 \times \text{largeur} \times \text{hauteur}$

Complexité :  $(3 + 5) \times \text{largeur} \times \text{hauteur}$



## Noyau avec fonction impulsionnelle (Dirac)

La fonction impulsionnelle a un pixel blanc en  $(0, 0)$  sur fond noir

$$I * \delta = I$$

Cas inverse

$$H * \delta = \delta * H = H$$



## Noyau gaussien

Un noyau gaussien 2D est le **produit** de noyaux gaussiens 1D

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{y^2}{2\sigma^2}\right) \\ &= g_\sigma(x) \cdot g_\sigma(y) \end{aligned}$$

→ La convolution avec un noyau gaussien est séparable



Bruit



## Types de bruit

Bruit poivre et sel : pixels blancs et noirs dispersés de manière aléatoire

- Appelé aussi bruit impulsionnel, bruit de grenaille ou bruit binaire
- Causé par une perturbation brusque et soudaine



Image d'origine



Bruit poivre et sel

133

## Types de bruit

Bruit gaussien : bruit blanc ajouté à l'image et normalement distribué

$$I = I + \text{bruit}$$



Image d'origine



Bruit gaussien

134

## Types de bruit

*Speckle* : la valeur des pixels est multipliée par un bruit aléatoire

$$I = I \cdot (1 + \text{bruit})$$



Image d'origine



Speckle

135

## Types de bruit

Bruit périodique : causé par des perturbations de nature périodique



136

## Types de bruit

Reconnaitre le type d'un bruit permet de choisir le meilleur filtre pour le supprimer !

Les bruits de type poivre et sel, gaussien et speckel peuvent être supprimés à l'aide de **filtres spatiaux**

Les bruits périodiques peuvent être supprimés à l'aide de **filtres fréquentiels**

137

## Filtres non linéaires

Les filtres linéaires floutent les structures telles que les contours

→ Ils réduisent la qualité de l'image

→ Leur utilisation doit être limitée pour supprimer le bruit



Image avec poivre et sel



Filtre linéaire (noyau 15x15)

139

## Filtres non linéaires

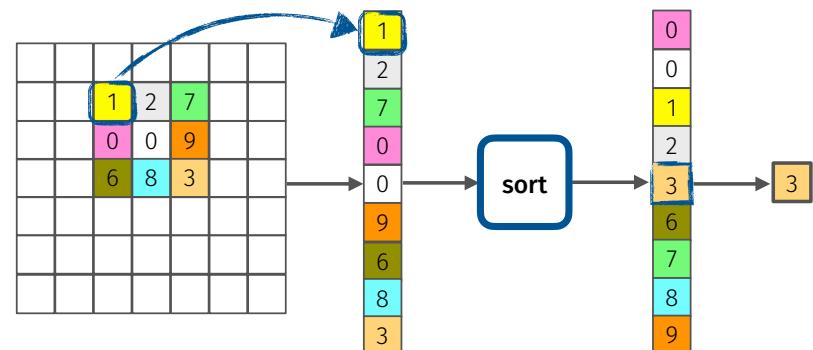


138

## Filtre médian

Un filtre médian supprime le bruit tout en conservant les structures

$$I'(u, v) = \text{median}\{ I(u + i, v + j) \mid (i, j) \in \mathbb{R} \}$$



140



## Effet du filtre médian



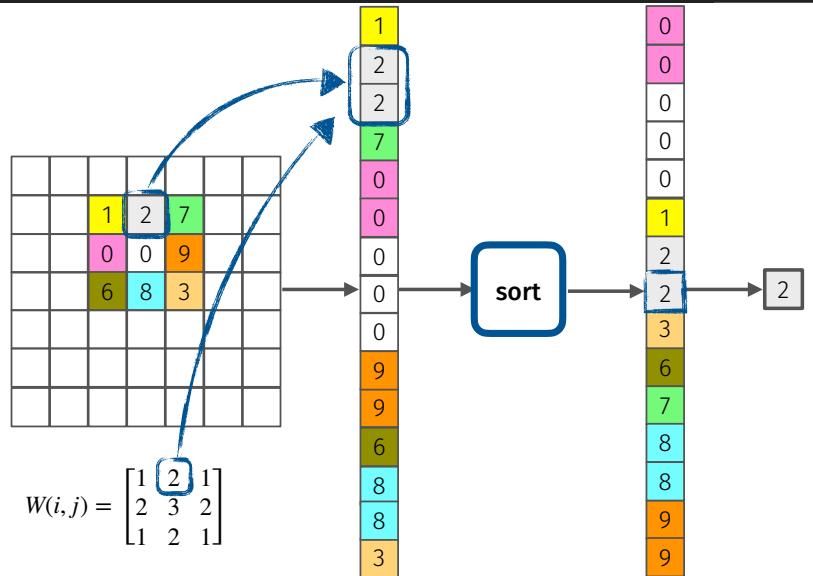
Filtre linéaire (noyau 15x15)



Filtre médian

141

## Filtre médian pondéré



142

## Autres filtres non linéaires

- Filtres min et max
- Filtres de détection de coins
- Filtres morphologiques

143

## Détection de contours

144

## Qu'est-ce qu'un contour ?

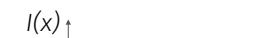
Un contour correspond à un changement brutal d'intensité

Exemples

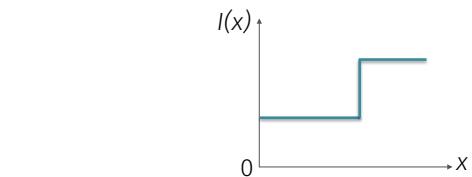
- Séparations entre des objets
- Changements d'orientation d'une surface
- Changements d'éclairage



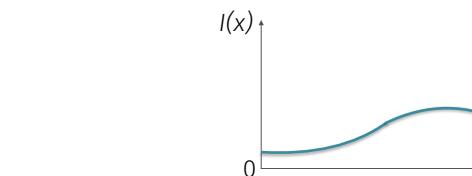
145



- Un contour idéal correspond à une fonction échelon de Heaviside



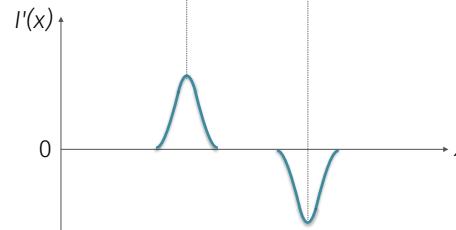
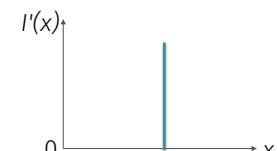
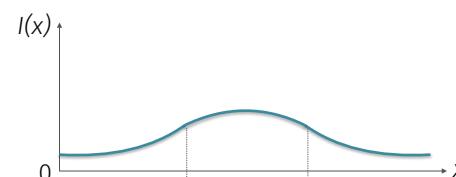
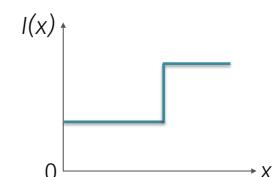
- Un contour réel correspond à une version adoucie de la fonction échelon



146

## Caractéristiques d'un contour

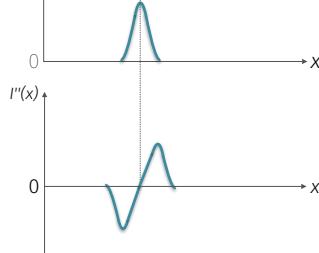
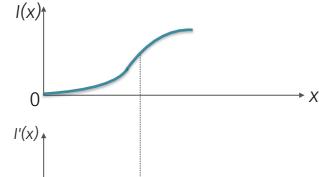
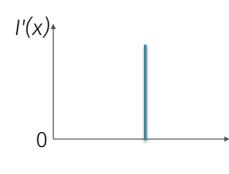
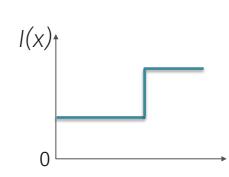
- Un contour peut être caractérisé par un **pic** pour la dérivée première



147

## Caractéristiques d'un contour

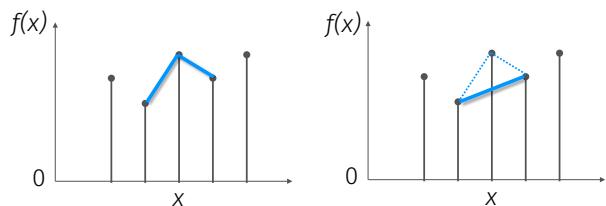
- La dérivée première de  $I(x)$  correspond à un **pic** au niveau du contour
- La dérivée seconde de  $I(x)$  passe par 0 au niveau du contour



148

## Pente d'une fonction discrète

- Les pentes gauche et droite peuvent être différentes
- Solution ? Prendre la moyenne des pentes gauche et droite



149

## Différences finies

Différence avant (pente droite)

$$\Delta_+ f(x) = f(x+1) - f(x)$$

Différence arrière (pente gauche)

$$\Delta_- f(x) = f(x) - f(x-1)$$

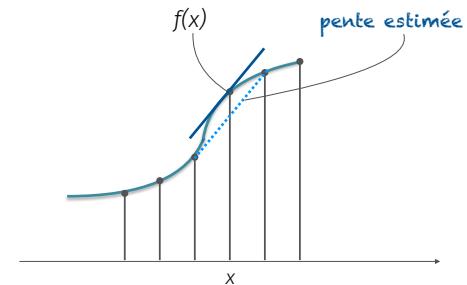
Différence centrée (pente moyenne)

$$\Delta f(x) = \frac{1}{2}(f(x+1) - f(x-1))$$

151

## Calculer la dérivée d'une fonction discrète

$$\frac{df}{dx}(x) \sim \frac{f(x+1) - f(x-1)}{2} = 0.5.(f(x+1) - f(x-1))$$



150

## Gradient

Soit  $f(x, y)$  une fonction 2D

- **Gradient** : vecteur dont la direction est celle du taux de variation maximal de  $f$  et dont l'amplitude est égale au taux de variation maximal de  $f$
- Le gradient est perpendiculaire au contour

$$\nabla f = \left[ \frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \right]^T$$

$$\text{Amplitude : } \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

$$\text{Direction : } \tan^{-1} \left( \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \right)$$

152

## Gradient d'une image

- Les dérivées d'une image s'obtiennent pour les directions horizontale et verticale

$$\frac{\partial I}{\partial u}(u, v) \quad \text{et} \quad \frac{\partial I}{\partial v}(u, v)$$

- Le gradient de l'image en  $(u, v)$  est :

$$\nabla I(u, v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u, v) & \frac{\partial I}{\partial v}(u, v) \end{bmatrix}^T$$

- L'amplitude du gradient est :

$$|\nabla I|(u, v) = \left[ \left( \frac{\partial I}{\partial u}(u, v) \right)^2 + \left( \frac{\partial I}{\partial v}(u, v) \right)^2 \right]^{1/2}$$

L'amplitude est invariante par rotation !

153

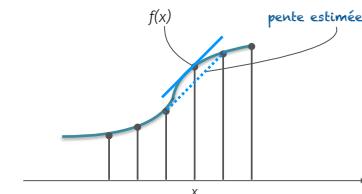
## Filtres dérivatifs

- On rappelle que la dérivée d'une fonction discrète s'obtient par :

$$\frac{df}{dx}(x) \sim \frac{f(x+1) - f(x-1)}{2} = 0.5.(f(x+1) - f(x-1))$$

- Peut-on définir un filtre linéaire pour calculer la différence centrée à l'aide d'un noyau ?

$$H_x^D = [-0.5 \quad 0 \quad 0.5] = 0.5.[-1 \quad 0 \quad 1]$$



154

## Réaliser une différence finie avec une convolution

- Différence avant

$$\Delta_+ f(x) = f(x+1) - f(x)$$

- Noyau de convolution :  $H = [0 \quad -1 \quad 1]$

$$\Delta_+ f = f * H$$

155

## Réaliser une différence finie avec une convolution

- Différence centrée

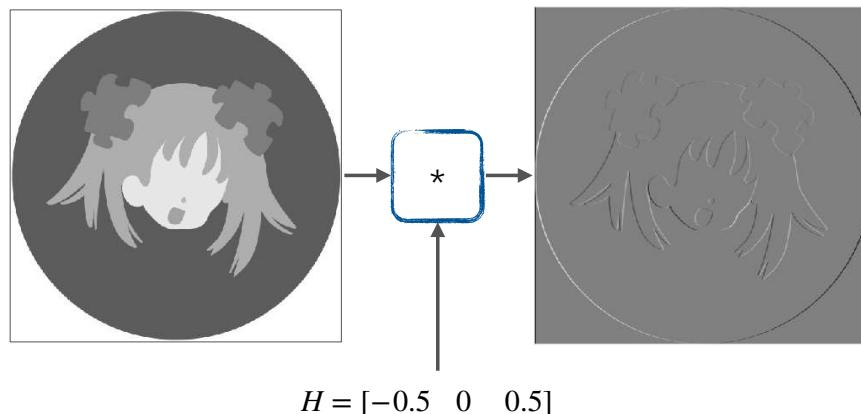
$$\Delta f(x) = \frac{1}{2}(f(x+1) - f(x-1))$$

- Noyau de convolution :  $H = [-0.5 \quad 0 \quad 0.5]$

$$\Delta f = f * H$$

156

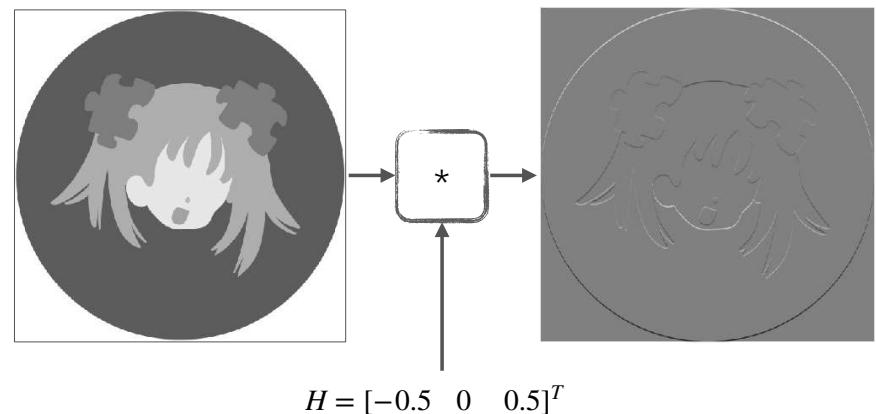
### dérivée en x d'une image à l'aide d'une différence centrée



$$H = [-0.5 \ 0 \ 0.5]$$

157

### dérivée en y d'une image à l'aide d'une différence centrée

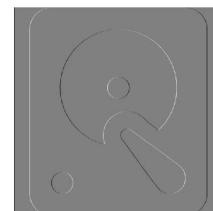


$$H = [-0.5 \ 0 \ 0.5]^T$$

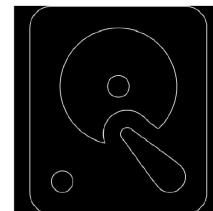
158

### Filtres dérivatifs

$$H_x^D = [-0.5 \ 0 \ 0.5]$$



$$H_y^D = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix}$$



Amplitude du gradient

159

### Dérivées partielles pour une image

- Dérivées partielles remplacées par des différences finies

$$\Delta_x f = f(x, y) - f(x - 1, y)$$

$$\Delta_y f = f(x, y) - f(x, y - 1)$$

Prewitt

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- Alternatives

$$\Delta_{2x} f = f(x + 1, y) - f(x - 1, y)$$

$$\Delta_{2y} f = f(x, y + 1) - f(x, y - 1)$$

Sobel

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Gradient de Robert

$$\Delta_+ f = f(x + 1, y + 1) - f(x, y)$$

$$\Delta_- f = f(x, y + 1) - f(x + 1, y)$$

160

## Combiner moyenne et dérivées : opérateur de Prewitt

- Un opérateur à différence finie est sensible au bruit
- La dérivation est plus robuste si le calcul se fait en moyennant sur un voisinage
- Opérateur de Prewitt : dérivée en x, puis moyenne en y

$$H_x^{\text{Prewitt}} = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * [0.5 \quad 0 \quad -0.5] = \frac{1}{6} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Moyenne dans la direction y

Dérivée dans la direction x

$H_y^{\text{Prewitt}}$  est définie de la même façon



161

- Similaire à Prewitt, mais le noyau de moyennage est plus élevé au centre

$$H_x^{\text{Sobel}} = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [0.5 \quad 0 \quad -0.5] = \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_y^{\text{Sobel}} = \frac{1}{4} [1 \quad 2 \quad 1] * \begin{bmatrix} 0.5 \\ 0 \\ -0.5 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Moyenne dans la direction x

Dérivée dans la direction y



162

## Opérateurs de Prewitt et Sobel

### Opérateur de Prewitt

$$H_x^{\text{Prewitt}} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{et} \quad H_y^{\text{Prewitt}} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

### Opérateur de Sobel

$$H_x^{\text{Sobel}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{et} \quad H_y^{\text{Sobel}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



163

## Opérateur de Sobel amélioré

- Jähne a proposé une version améliorée de l'opérateur de Sobel

$$H_x^{\text{meilleurSobel}} = \frac{1}{32} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$

$$H_y^{\text{meilleurSobel}} = \frac{1}{32} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

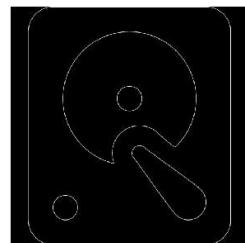


164

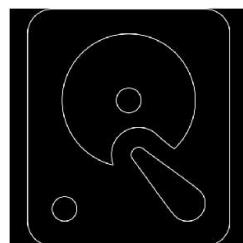
## Opérateurs de Roberts, Prewitt et Sobel



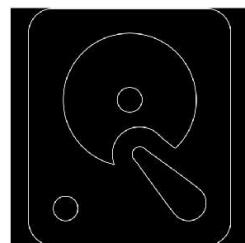
Image



Roberts



Prewitt



Sobel

165

## Opérateurs de Roberts, Prewitt et Sobelors



Image



Roberts



Prewitt



Sobel

166

## Opérateurs de Roberts, Prewitt et Sobel



Image



Roberts



Prewitt



Sobel

167

## Autres opérateurs de détection de contour

Pour les opérateurs basés sur la dérivée première, le contour est proportionnel à la variation d'intensité sous-jacente et peut être difficile à localiser avec précision

### Solution

Utiliser la dérivée seconde avec un préfiltrage pour lisser l'image afin de modérer l'amplification du bruit

168

## Filtre de Canny

L'opérateur de Canny minimise le nombre de pixels n'appartenant pas au contour et en permet une meilleure localisation

### Étapes

1. Lisser l'image à l'aide d'un filtre gaussien
2. Trouver les gradients d'intensité
3. Appliquer un seuillage en fonction de l'amplitude du gradient
4. Appliquer un double seuil pour déterminer les contours potentiels
5. Supprimer les contours faibles qui ne sont pas connectés aux contours forts



169

## Netteté des images

## Netteté d'une image

Un flou peut apparaître lors de la numérisation ou de la mise à l'échelle d'une image

- Rehausser la netteté va réduire les effets de flou
- Comment ? En amplifiant les composantes haute fréquence
- Les contours sont justement des zones hautes fréquences

Deux approches pour rehausser les contours :

- Opérateur laplacien
- Masquage flou

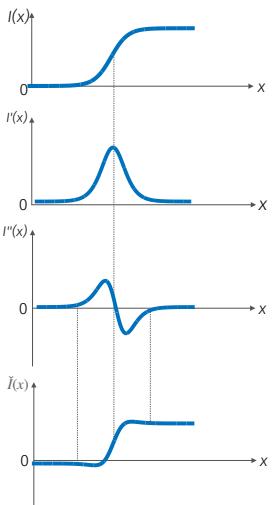
171

## Rehausser un contour à l'aide d'un opérateur laplacien

$$\tilde{I} = I(x) - w \cdot I''(x)$$

Diagramme montrant les étapes de calcul de l'opérateur laplacien :

- Intensité de l'image :  $I(x)$
- Poids :  $w$
- Dérivée seconde de l'intensité :  $I''(x)$



172

## Opérateur laplacien

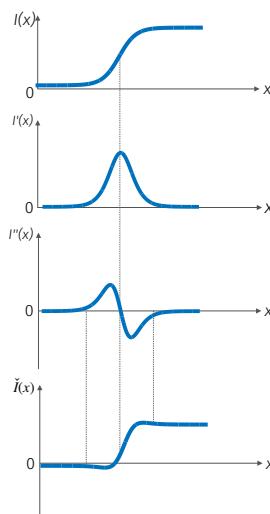
- L'opérateur laplacien 2D combine des dérivées secondes dans les directions verticale et horizontale

- L'opérateur laplacien est défini par :

$$(\nabla^2 I)(x, y) = \frac{\partial^2 I}{\partial x^2}(x, y) + \frac{\partial^2 I}{\partial y^2}(x, y)$$

Dérivée 2nde dans  
la direction x

Dérivée 2nde dans  
la direction y



173

## Opérateur laplacien

- Approximation du laplacien :

$$\nabla^2 I(x, y) = [I(x+1, y) - I(x, y)] - [I(x, y) - I(x-1, y)] +$$

$$[I(x, y+1) - I(x, y)] - [I(x, y) - I(x, y-1)]$$

$$= [I(x+1, y) + I(x-1, y) + I(x, y+1) - I(x, y-1)] - 4I(x, y)$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

174

## Opérateur laplacien

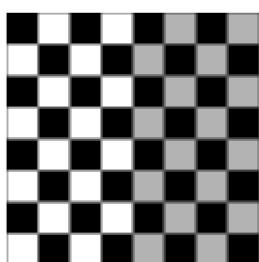
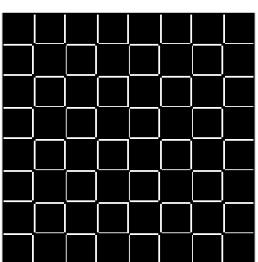


Image floue



Laplacien

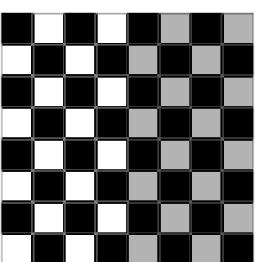


Image plus nette

L'image rehaussée est obtenue en soustrayant  
le laplacien de l'image originale

175

## Masquage flou

- 1) Soustraire l'image lissée (lissage gaussien  $\sigma$ ) de l'image d'origine afin d'obtenir un masque de contours

$$M = I - (I * \tilde{H})$$

- 2) Ajouter à l'image d'origine une version pondérée du masque

$$\check{I} = I + pM$$

Avantages par rapport à l'opérateur laplacien

- Sensibilité au bruit réduite grâce au lissage
- Opération améliorée grâce aux paramètres  $\sigma$  et  $p$

176

## Masquage flou

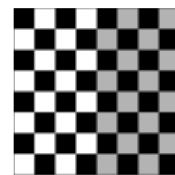
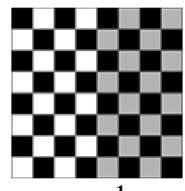
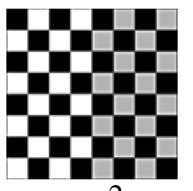


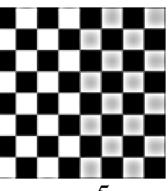
Image d'origine floue



$\sigma = 1$



$\sigma = 2$



$\sigma = 5$

Rehaussements

177

## Filtres spatiaux et Python

## Filtres spatiaux en Python

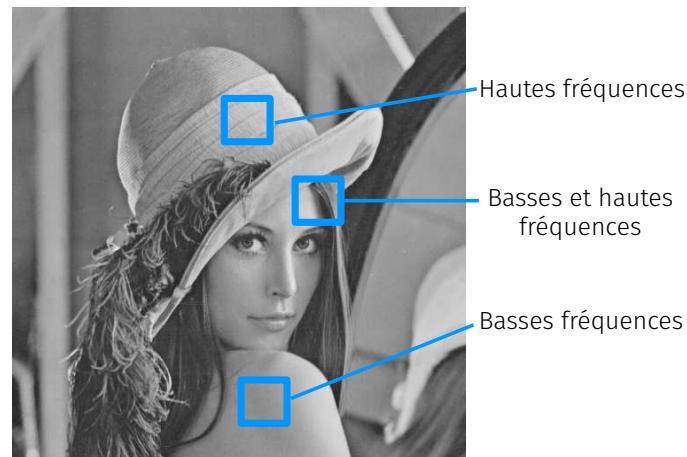
Le module `filters` de la bibliothèque [scikit-image](#) propose la plupart des filtres spatiaux vus ici

179

## 5. Filtres fréquentiels

178

## Fréquences dans une image



181

Transformation de Fourier discrète (TFD)

183

## Filtres dans le domaine fréquentiel

L'une des raisons de l'utilisation de la transformée de Fourier dans le traitement d'images est due à la convolution spatiale, très gourmande en ressources.

La convolution spatiale est remplacée par une **multiplication élément par élément** de la transformée de Fourier par la matrice d'un filtre

182

## Transformation de Fourier discrète 1D (TFD 1D)

Soit  $f = [f_0, f_1, f_2, \dots, f_{N-1}]$

Alors  $F = [F_0, F_1, F_2, \dots, F_{N-1}]$

où  $F_u = \frac{1}{N} \sum_{x=0}^{N-1} f_x \exp \left[ -2\pi i \frac{xu}{N} \right]$

obtenu par fenêtrage  
+ échantillonnage

Transformation inverse :  $x_u = \sum_{x=0}^{N-1} F_u \exp \left[ 2\pi i \frac{xu}{N} \right]$

184

## Transformation de Fourier discrète 2D (TFD 2D)

Soit  $F$  la matrice de la **transformée de Fourier** de  $f$ :

$$F = \mathcal{F}(f)$$

Alors la matrice d'origine  $f$  est la transformée de Fourier inverse de  $F$ :

$$f = \mathcal{F}^{-1}(F)$$

185

## Transformation de Fourier discrète 2D (TFD 2D)

Pour une matrice  $M \times N$ , la transformée de Fourier s'écrit :

$$\text{TFD complexe} \quad \text{image}$$
$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp \left[ -2\pi i \left( \frac{xu}{M} + \frac{yv}{N} \right) \right],$$

fréquences  $u, v$  pour les directions  $x, y$

et son inverse s'écrit :

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp \left[ 2\pi i \left( \frac{xu}{M} + \frac{yv}{N} \right) \right].$$

186

## Transformation de Fourier rapide (FFT)

- Plusieurs algorithmes pour calculer la TFD rapidement
- La **transformée de Fourier rapide (FFT)** en est un
  - L'algorithme divise le vecteur d'origine en 2
  - Calcule récursivement la FFT de chaque moitié
  - Puis fusionne les résultats
- Le calcul direct prend  $2^{2n}$  multiplications en temps
- Avec la FFT, le calcul prend  $n2^n$  multiplications en temps
- Temps gagné :  $2^n/n$

187

## Propriétés de la TFD 2D : séparabilité

- Remarquons que :

$$\exp \left[ \pm 2\pi i \left( \frac{xu}{M} + \frac{yv}{N} \right) \right] = \exp \left[ \pm 2\pi i \frac{xu}{M} \right] \exp \left[ \pm 2\pi i \frac{yv}{N} \right]$$

$\longleftrightarrow$  TFD 2D       $\longleftrightarrow$  TFD 1D (ligne)       $\longleftrightarrow$  TFD 1D (colonne)

- La propriété de séparabilité nous permet d'utiliser les TFD 1D pour calculer les lignes, puis les colonnes de la transformée de Fourier 2D

188

## Propriétés de la TFD 2D : linéarité

- La TFD d'une somme est la somme des TFD individuelles

$$\mathcal{F}(f+g) = \mathcal{F}(f) + \mathcal{F}(g)$$

- De même :

$$\mathcal{F}(kf) = k\mathcal{F}(f) \quad \text{où } k \text{ est un scalaire}$$

- Ces propriétés sont intéressantes pour traiter des dégradations représentées par des sommes (ex. : bruit)

$$f' = f + n$$

$$\mathcal{F}(f') = \mathcal{F}(f) + \mathcal{F}(n)$$

⇒ Le bruit peut être supprimé ou réduit en modifiant la transformée de  $n$



## Convolution et TFD

La TFD propose une méthode alternative pour réaliser une convolution entre une image  $I$  et un filtre spatial  $S$

$$I * S = \mathcal{F}^{-1}(\mathcal{F}(I) \cdot \mathcal{F}(S'))$$

où  $S'$  est tel que  $S$  a subi un "rembourrage" afin d'avoir la même taille que  $I$

## Convolution et TFD

Exemple :  $I = 512 \times 512$ ,  $S = 32 \times 32$

### Calcul de convolution classique

- $32^2 = 1024$  multiplications pour chaque pixel
- Total des multiplications =  $512 \times 512 \times 1024 = \mathbf{268 \ 435 \ 456}$

### Calcul en utilisant des TFD

- Chaque ligne requiert 4608 multiplications
- Multiplications pour les lignes =  $4608 \times 512 = 2 \ 359 \ 296$
- Itération sur les colonnes, TFD :  $4 \ 718 \ 592$  multiplications
- Même chose pour la TFD du filtre et pour la TFD inverse
- En plus :  $512 \times 512$  multiplications pour le produit des 2 TFD
- Total des multiplications =  $4 \ 718 \ 592 \times 3 + 262144 = \mathbf{14 \ 417 \ 920}$



## Composante continue de la TFD

La valeur  $F(0, 0)$  de la TFD est appelée **composante continue**

$$F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp(0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

Elle correspond à la somme de tous les termes dans la matrice d'origine

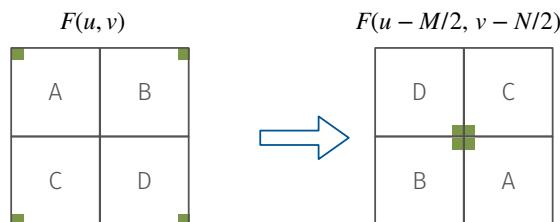


190

192

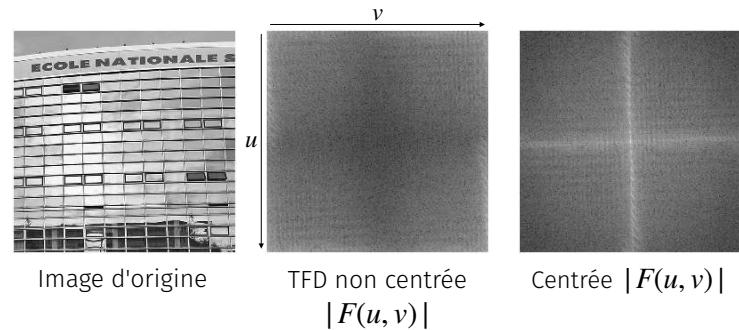
## Centrer le spectre de la TFD

- $F(0, 0)$  est au coin en haut à gauche
- Pour afficher la TFD, il est plus pratique de placer la composante continue au centre
- Pour ce faire, nous échangeons les cadrons



193

## Centrer le spectre de la TFD

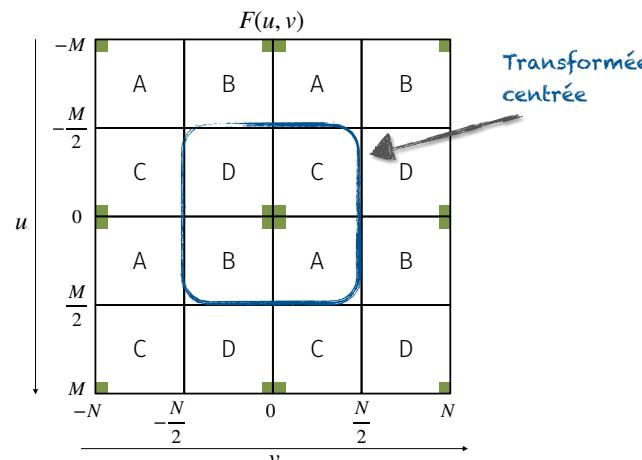


La composante continue est beaucoup plus grande que les autres valeurs, nous étirons donc les valeurs de la transformation en affichant :  $\log(1 + |F(u, v)|)$

194

## Symétrie et conjugué

- L'information dans une TFD est redondante sur une moitié



195

## Exemple de calcul d'une TFD 2D en Python

```
import numpy as np
import matplotlib.pyplot as plt
import cmath

def DFT2D(image):
    data = np.asarray(image)
    M, N = image.size # (img x, img y)
    dft2d = np.zeros((M, N), dtype=complex)
    for u in range(M):
        for v in range(N):
            sum_matrix = 0.0
            for m in range(M):
                for n in range(N):
                    e = cmath.exp(- 2j * np.pi * ((u * m) / M + (v * n) / N))
                    sum_matrix += data[m, n, 1] * e
            dft2d[u, v] = sum_matrix
    return dft2d

img = ...
img2 = img.resize((50,50))
plt.imshow(img2)
dft = DFT2D(img2)
plt.imshow(dft.real)
```

196

## Quelques exemples de TFD



197

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## TFD d'une image uniforme avec un contour



Image d'origine



$\log(1+|F|)$



199

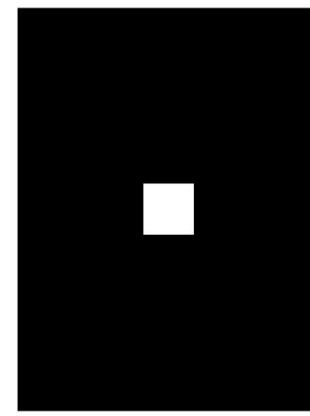
## TFD d'une image uniforme

- Supposons une image uniforme 13x8 formée de pixels à 1 :  $I(x, y) = 1$
- La TFD à une composante continue et le reste est à 0
- La composante continue est la somme de tous les éléments = 104

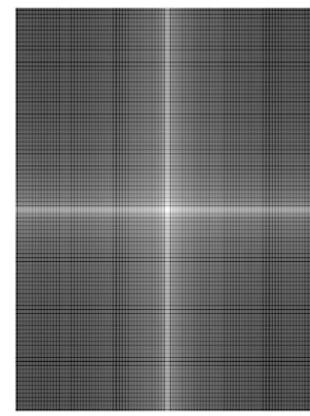


198

## TFD d'un carré synthétique



Image

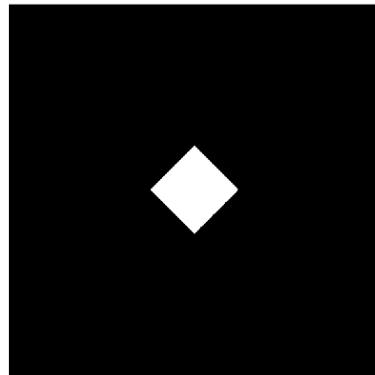


$\log(1+|F|)$

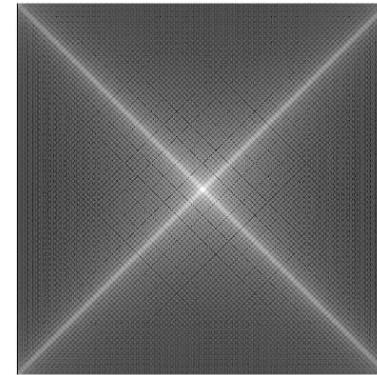


200

### TFD d'un carré synthétique ayant subi une rotation



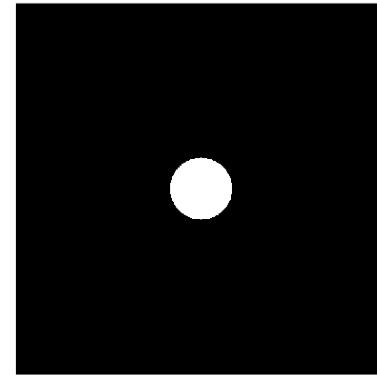
Image



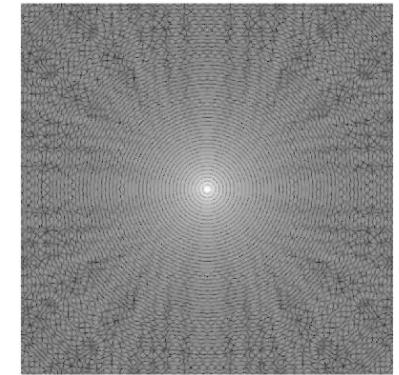
$\log(1+|F|)$

201

### TFD d'un disque synthétique



Image



$\log(1+|F|)$

202

## Filtrage dans le domaine fréquentiel

203

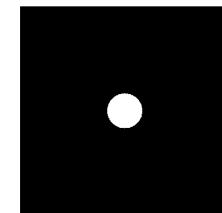
### Filtre passe-bas idéal

Un filtre passe-bas conserve les fréquences situées en dessous d'une fréquence de coupure

- Après une TFD, la composante continue et les composantes basse fréquence sont situées autour du centre

- Un filtre spécifie sa fréquence de coupure à l'aide d'un disque

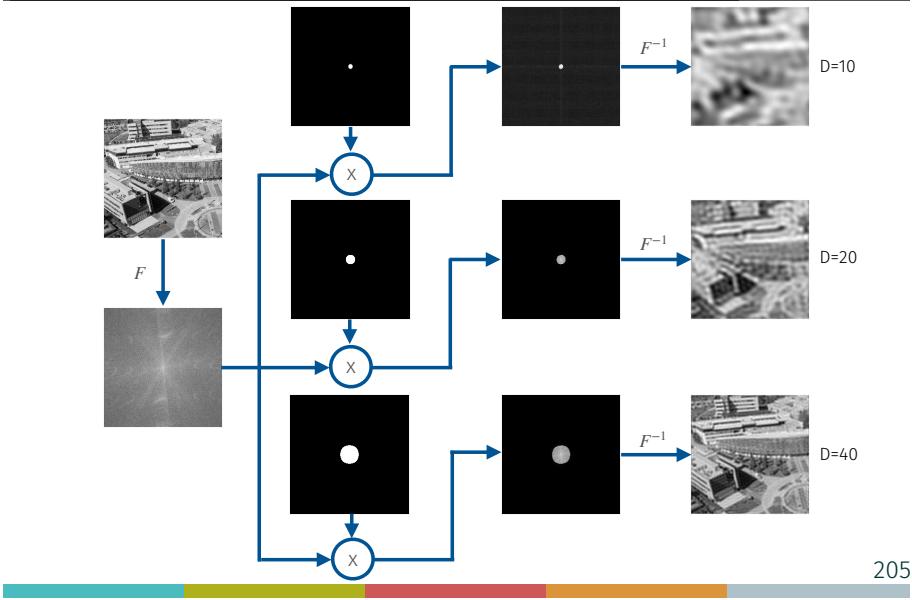
$$m(x, y) = \begin{cases} 1 & \text{si } (x, y) \text{ est plus proche du centre qu'une valeur donnée } R \\ 0 & \text{si } (x, y) \text{ est plus éloigné du centre que } R \end{cases}$$



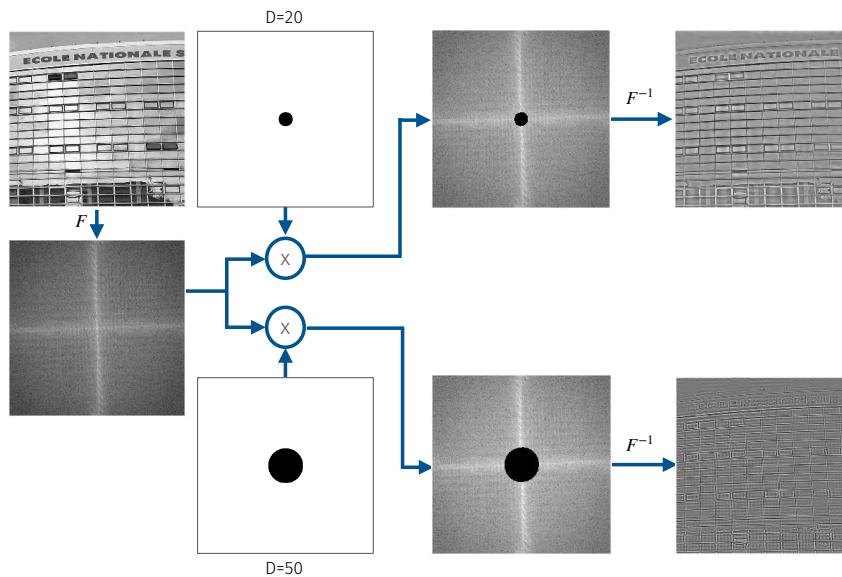
Un filtrage passe-bas amène un flou

204

## Filtre passe-bas idéal



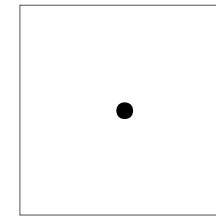
## Filtre passe-haut idéal



## Filtre passe-haut idéal

C'est l'opposé du filtre passe-bas : il faut éliminer les fréquences au centre et conserver les autres

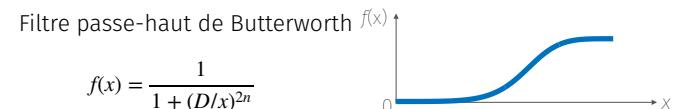
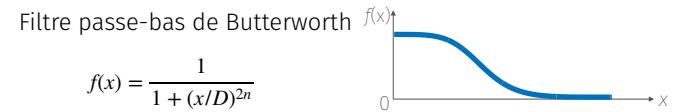
- Un filtrage passe-haut **accentue les contours**
- Nous utilisons à présent un disque binaire noir sur fond blanc



206

## Filtre de Butterworth

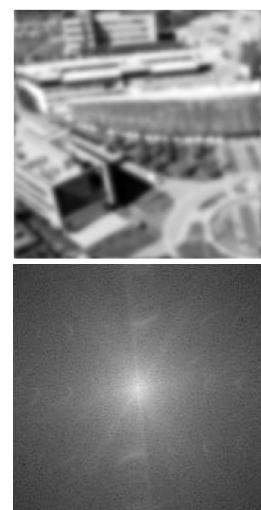
- Une fréquence de coupure nette entraîne des artefacts oscillatoires sur la TFD
- Les filtres de Butterworth ont une bordure plus douce (disque avec une bordure adoucie)



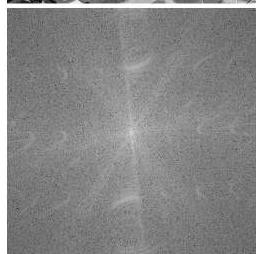
$n$  est l'ordre du filtre et permet de contrôler l'adoucissement

208

## Filtre passe-bas de Butterworth



D=20, n = 2

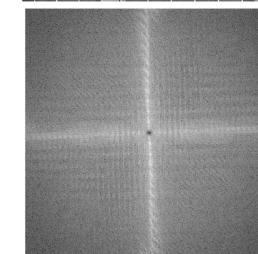
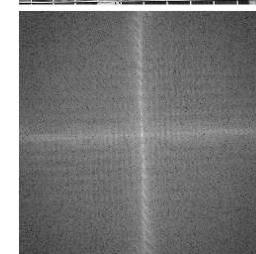


209

## Filtre passe-haut de Butterworth



D=10, n = 2



210

## Filtrage gaussien

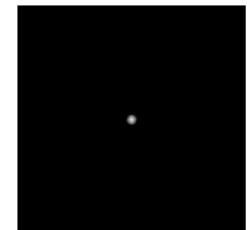
Les filtres gaussiens peuvent être appliqués dans le domaine fréquentiel

- Créer un filtre gaussien
- Multiplier la TFD de l'image par le filtre gaussien
- Inverser le résultat

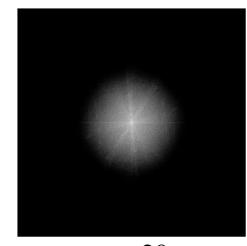
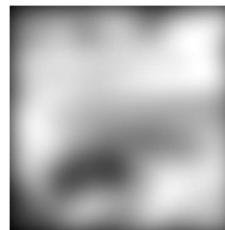
La transformée de Fourier d'une gaussienne étant aussi une gaussienne, il suffit de multiplier directement la TFD de l'image par la gaussienne (pas besoin de trouver sa transformée de Fourier)

211

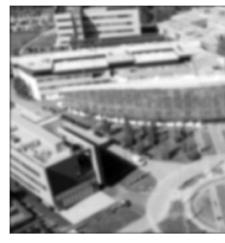
## Filtre passe-bas gaussien



$\sigma = 2$

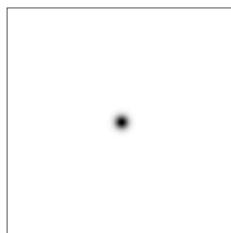
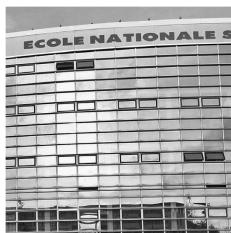


$\sigma = 20$

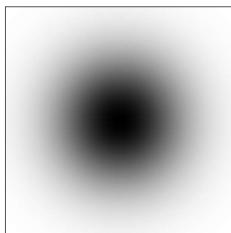


212

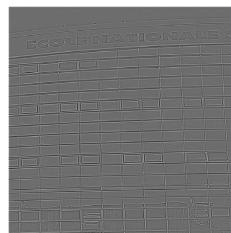
## Filtre passe-haut gaussien



$\sigma = 2$



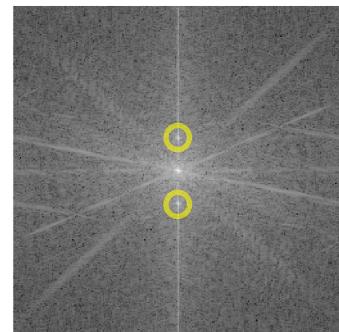
$\sigma = 20$



213

## Suppression de bruit périodique

- Un bruit périodique ne peut être supprimé dans le domaine spatial
- Par contre, il peut l'être dans le domaine fréquentiel
- Un bruit périodique apparaît sous la forme de "pics" éloignés du centre



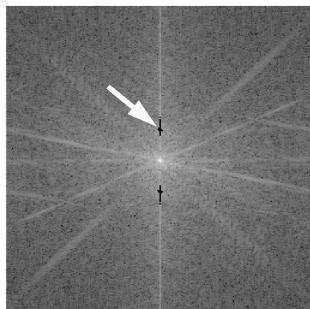
214

## Suppression de bruit périodique

2 façons de supprimer le bruit périodique dans le domaine fréquentiel :

- Filtre coupe-bande
- Filtre réjecteur de bande

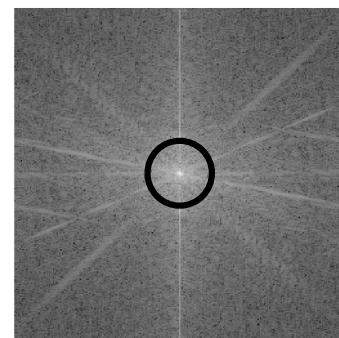
Filtre coupe-bande



215

## Suppression de bruit périodique

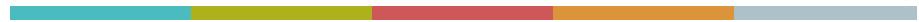
Filtre réjecteur de bande



216

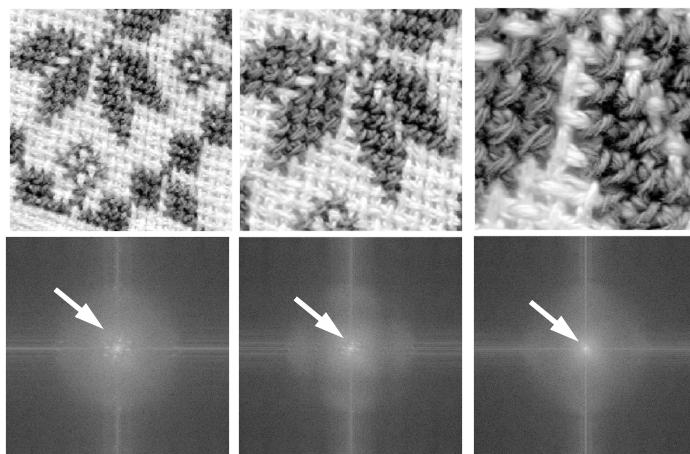
### Autres exemples de TFD

217



### Exemples de TFD 2D : motifs périodiques

Les motifs périodiques répétitifs apparaissent sous forme de pics distincts à des positions correspondantes dans le spectre

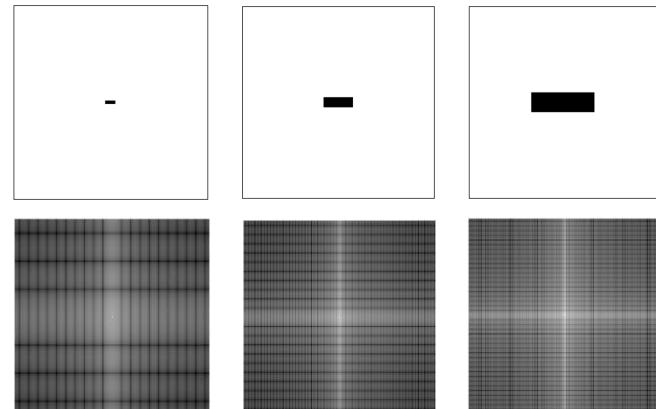


219



### Exemples de TFD 2D : mise à l'échelle

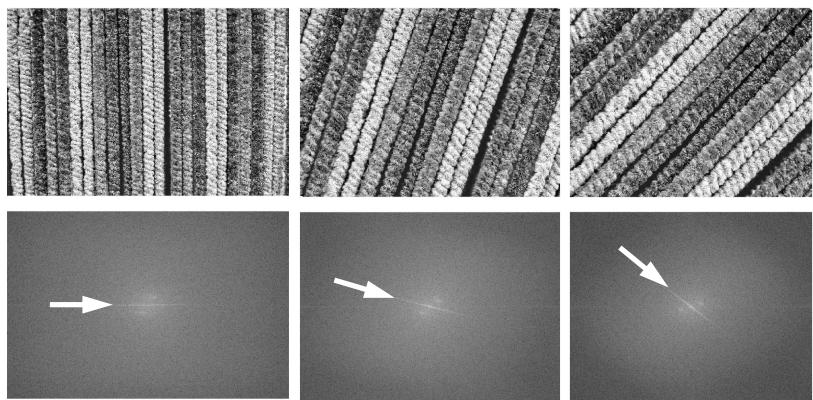
Une élongation de l'image correspond à une contraction du spectre et vice versa



218

### Exemples de TFD 2D : rotation

Une rotation correspond à une rotation du spectre par le même angle/quantité

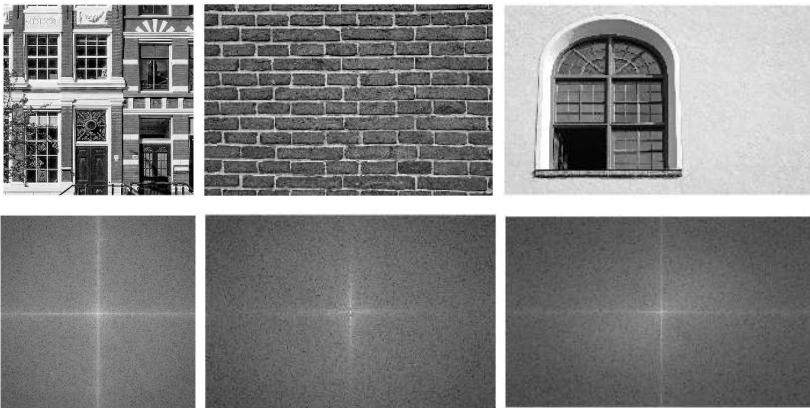


220



## Exemples de TFD 2D : structures allongées et orientées

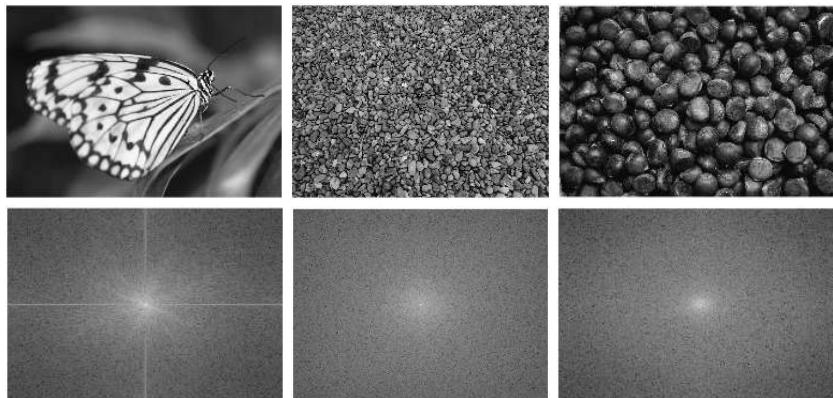
Les motifs réguliers, allongés et artificiels sont dominants dans le spectre



221

## Exemples de TFD 2D : scènes naturelles

Les motifs répétitifs des images naturelles n'ont pas d'orientation dominante



223

## Exemples de TFD 2D : scènes naturelles

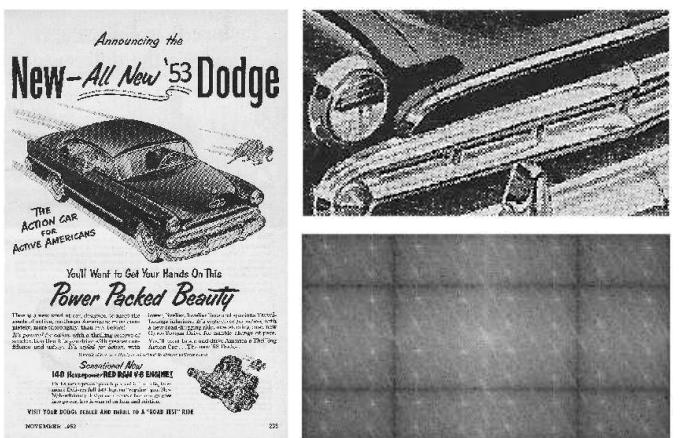
Les répétitions dans les scènes naturelles sont moins dominantes que celles créées par l'homme



222

## Exemples de TFD 2D : motifs imprimés

Les motifs diagonaux et réguliers causés par une impression sont clairement visibles dans le spectre



224

## *Filtres fréquentiels et Python*

### **Filtres fréquentiels et Python**

Les bibliothèques [NumPy](#) et [SciPy](#) proposent toutes les deux un calcul de la FFT. Nous utiliserons la version de NumPy.

225

226

## **6. Morphologie mathématique**

### **Introduction**

- Introduite en 1964 par Georges Matheron et Jean Serra
- Initialement utilisée sur des images binaires dans le cadre d'applications liées à la prospection minière

Quels types d'images binaires ?

- Fax, images imprimées
- Images en niveaux de gris seuillées

*Attention : les filtres morphologiques peuvent altérer les structures locales d'une image*

228

## Dilatation et érosion

### Dilatation et érosion

2 opérations de base

- Dilatation
- Érosion

Des opérations combinant les précédentes

- Ouverture
- Fermeture
- etc.



229

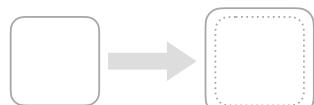


230

### Dilatation

Utilisé pour :

Agrandir



Boucher les trous

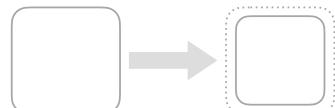


231

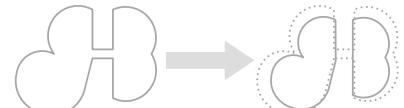
### Érosion

Utilisée pour :

Réduire



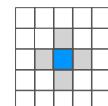
Supprimer les branches et les ponts



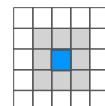
232

## Voisinage

4-voisins ( $\mathcal{N}_4$ )  
8-voisins ( $\mathcal{N}_8$ )



$\mathcal{N}_4$



$\mathcal{N}_8$



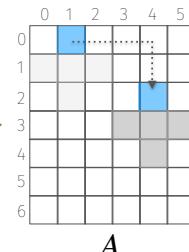
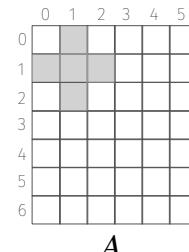
## Translation

Soit  $A$  un ensemble de pixels dans une image binaire

- Si  $w = (x, y)$  est un point de coordonnée spécifique
- $A_w$  est le translaté de  $A$  dans la direction  $(x, y)$ . c.-à-d.

$$A_w = \{(a, b) + (x, y); (a, b) \in A\}$$

Exemple : si  $A$  est l'ensemble formant une croix, et  $w = (3, 2)$



235

## Une image vue comme un ensemble de points

Les opérations morphologiques considèrent les images comme des ensembles de points en 2D

Par exemple, soit  $I$  une image telle que  $I(u, v) \in \{0, 1\}$ , alors :

$$\mathcal{Q}_I = \{ p \mid I(p) = 1 \}, \text{ et}$$

L'opération **OU** est l'**union** des ensembles :

$$\mathcal{Q}_{I_1 \vee I_2} = \mathcal{Q}_{I_1} \cup \mathcal{Q}_{I_2}$$

L'opération **ET** est l'**intersection** des ensembles :

$$\mathcal{Q}_{I_1 \wedge I_2} = \mathcal{Q}_{I_1} \cap \mathcal{Q}_{I_2}$$

234

## Dilatation

Supposons  $A$  et  $B$  des ensembles de pixels, la dilatation de  $A$  par  $B$  est :

$$A \oplus B = \bigcup_{x \in B} A_x$$

- Cette opération est appelée **addition de Minkowski**

- Pour chaque pixel  $x$  de  $B$  :

- 1) On translate  $A$  de  $x$
- 2) On prend l'union de toutes ces translations

$$A \oplus B = \{ (x, y) + (u, v) \mid (x, y) \in A, (u, v) \in B \}$$

236



## Propriétés de la dilatation

La dilatation est commutative

$$I \oplus H = H \oplus I$$

La dilatation est associative

$$(I_1 \oplus I_2) \oplus I_3 = I_1 \oplus (I_2 \oplus I_3)$$

## Exemple de dilatation

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

A

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A_{(-1,-1)}$

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A_{(-1,1)}$

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A_{(1,-1)}$

-1	0	1
-1	0	1
0		

B

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A_{(1,1)}$

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

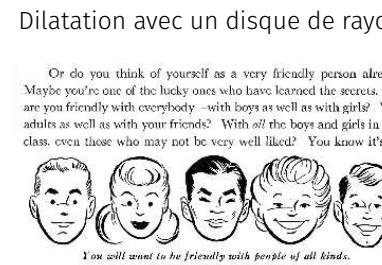
$A \oplus B$

## Élément structurant

- Nous considérons que  $B$  est un ensemble de pixels plus petit de  $A$
- $B$  est appelé **élément structurant**
- Un élément structurant correspond à un "masque" utilisé dans les opérations morphologiques
- Il est formé uniquement de 0 et de 1
- Il peut avoir toute forme et toute taille, et il possède une origine qui peut être en dehors de l'élément

237

## Exemple de dilation sur une image binaire



thing to be friendly with the people you like—with those in your own crowd, or with those you'd like to get in with. It's another thing to be friendly with people whom you don't know very well, or whom you think you don't like very well.



Or do you think of yourself as a very friendly person already? Maybe you're one of the lucky ones who have learned the secrets. But are you friendly with everybody—with boys as well as with girls? With adults as well as with your friends? With all the boys and girls in your class, even those who may not be very well liked? You know it's one

239

240

## Érosion

Soit les ensembles  $A$  et  $B$ , alors l'**érosion** de  $A$  par  $B$  est :

$$A \ominus B = \cap_{b \in B} A_b$$

- Cette opération est appelée **soustraction de Minkowski**
- Elle cherche toutes les occurrences de  $B$  dans  $A$

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

241

## Exemple d'une érosion

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												

$A$

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												

...

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												

$A \ominus B$

-1	0	1
-1		
0		

$B$

243

## Propriétés de l'érosion

L'érosion n'est pas commutative

$$I \ominus H \neq H \ominus I$$

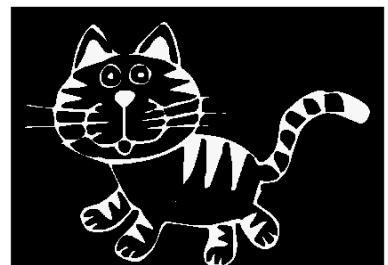
La dilatation d'un objet au premier plan correspond à l'érosion du fond

$$I \oplus H = \overline{I \ominus H^*}$$

242

## Exemple d'érosion sur une image binaire

Érosion par un disque de rayon 5 comme élément structurant



244

## Détection de contours

### Application à la détection de contours

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

A

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A \oplus B$

-1	0	1
1	0	-1
0	1	0

B

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$(A \oplus B) - A$

Contour externe

246



### Application à la détection de contours

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

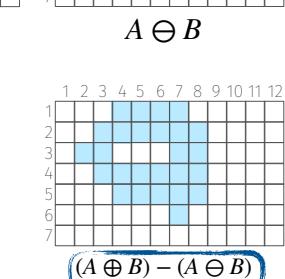
A

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A \ominus B$

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A \oplus B$



Gradient morphologique



247

### Application à la détection de contours

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

A

1	2	3	4	5	6	7	8	9	10	11	12
1											
2											
3											
4											
5											
6											
7											

$A \ominus B$

B

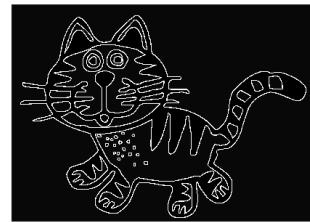
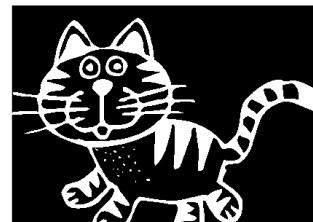
$A - (A \ominus B)$

Contour interne

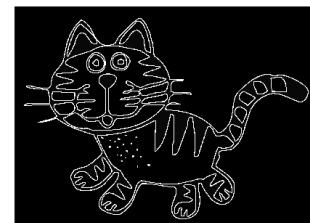
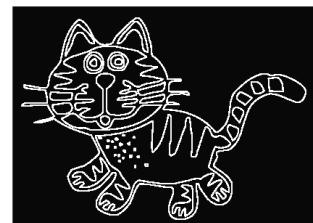
248



## Application à la détection de contours



Contour externe



Gradient morphologique

Contour interne

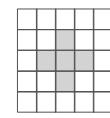
249

## Création de filtres morphologiques

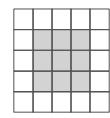
### Création de filtres morphologiques

Un filtre morphologique est caractérisé par :

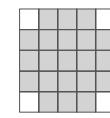
- Le type d'opération (ex. : dilatation, érosion)
- Le type de l'élément structurant



4-voisins



8-voisins



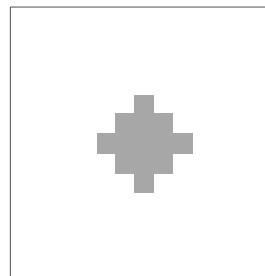
disque

- Les éléments structurants circulaires sont souvent utilisés dans la pratique
- La dilatation par un disque de rayon R ajoute une épaisseur R
- L'érosion par un disque de rayon R supprime une épaisseur R

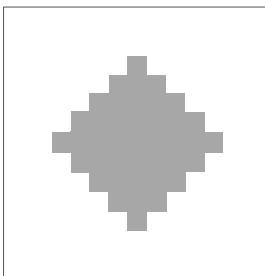
251

### Création de filtres morphologiques

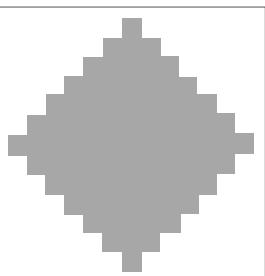
On crée de grands filtres en appliquant plusieurs fois de suite des filtres plus petits (pas forcément les mêmes à chaque itération d'ailleurs)



$H_A$



$H_A \oplus H_A$



$H_A \oplus H_A \oplus H_A$

252

## Ouverture et fermeture

253

## Ouverture

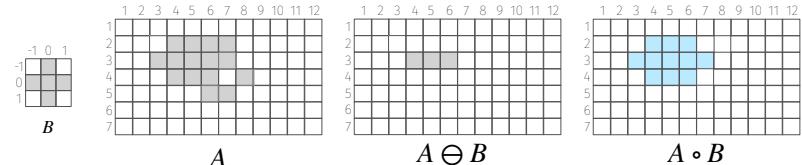
- L'**ouverture** est construite à partir de la dilatation et de l'érosion
- L'ouverture de  $A$  par l'élément structurant  $B$  est :

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

- On peut aussi écrire :

$$A \circ B = \{ B(x, y) \mid B(x, y) \subseteq A \}$$

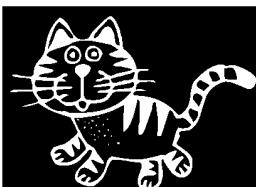
(l'ouverture est l'union de toutes les translations de  $B$  qui sont contenues dans  $A$ )



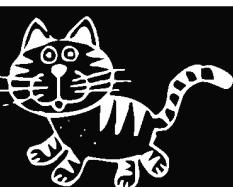
254

## Ouverture

- Les structures de premier plan plus petites que l'élément structurant sont éliminées en premier lieu (érosion)
- Les structures restantes sont lissées à l'étape suivante (dilatation) puis ramenées à leur taille d'origine



disque (R=1)



disque (R=5)



disque (R=11)

255

## Propriétés de l'ouverture

L'ouverture crée un sous-ensemble de  $A$  :

$$A \circ B \subseteq A$$

Idempotence (une ouverture ne peut être appliquée qu'une seule fois)

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

Sous-ensembles :

$$\text{Si } A \subseteq C, \text{ alors } (A \circ B) \subseteq (C \circ B)$$

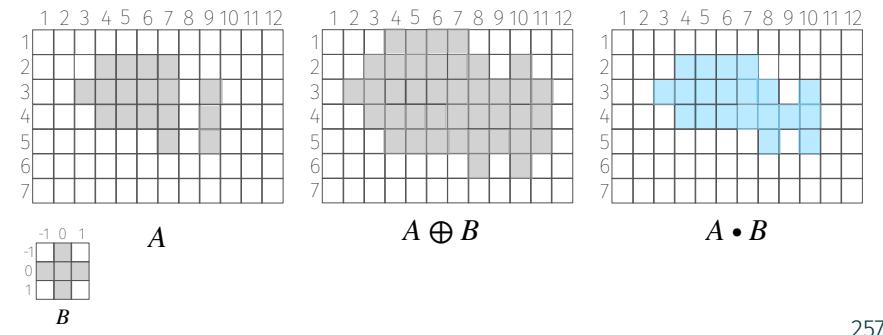
256

## Fermeture

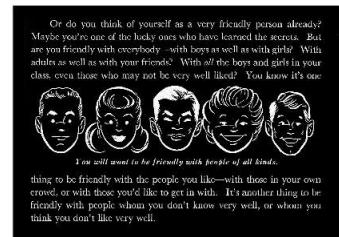
- La **fermeture** est construite à partir de la dilatation et de l'érosion
- La fermeture de  $A$  par l'élément structurant  $B$  est :

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

- La fermeture est une dilatation suivie d'une érosion



## Exemple d'une fermeture



259

## Propriétés de la fermeture

$A$  est un sous-ensemble de la fermeture :

$$A \subseteq (A \bullet B)$$

**Idempotence** (la fermeture ne peut être appliquée qu'une seule fois)

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

Sous-ensembles :

$$\text{Si } A \subseteq C, \text{ alors } (A \bullet B) \subseteq (C \bullet B)$$

258

## Relations entre ouverture et fermeture

- Ouverture et fermeture sont des opérations **duales** (l'ouverture du premier-plan correspond à la fermeture du fond et vice versa)

- Le complémentaire d'une fermeture = l'ouverture du complémentaire

$$A \bullet \bar{B} = \bar{A} \circ \hat{B}$$

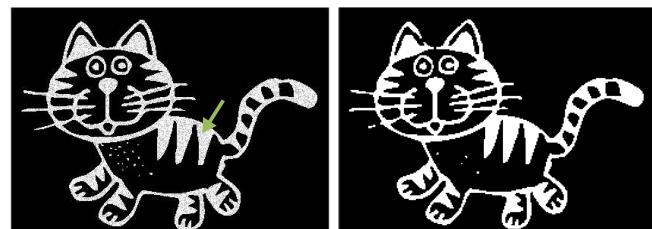
- Le complémentaire d'une ouverture = la fermeture du complémentaire

$$\bar{A} \circ \bar{B} = \bar{A} \bullet \hat{B}$$

260

## Filtrage morphologique

261



$B = \text{disque}(R=3)$

263

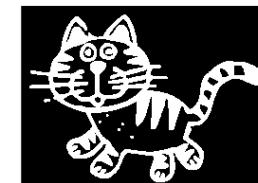
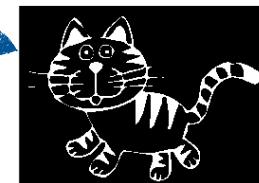
## Débruitage

Supposons  $A$ , une image bruitée par du speckle



- $A \ominus B$  supprime les pixels noirs, mais élargit les trous
- Nous pouvons reboucher les trous en dilatant 2 fois

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



$B = \text{disque}(R=3)$

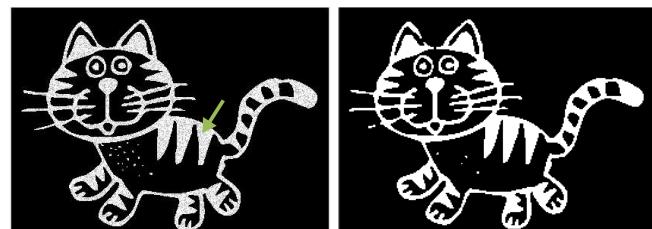
262

## Débruitage

- La 1ère dilatation ramène les trous à leur taille d'origine
- La 2nde dilatation supprime les trous, mais élargit les objets dans l'image
- Pour les ramener à leur taille d'origine, nous appliquons une érosion finale

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

- Ce débruitage correspond à une ouverture suivie d'une fermeture  
 $(A \circ B) \bullet B$



## Morphologie sur les images en niveaux de gris

264

## Morphologie et images en niveaux de gris

Les opérations de morphologie peuvent également être appliquées aux images en niveaux de gris

- Il suffit de remplacer **OU** et **ET** par **max** et **min**
- Les opérateurs pour le niveaux de gris peuvent également s'appliquer sur des images binaires
- Pour les images en couleur, effectuez des opérations de morphologie en niveaux de gris sur chaque canal de couleur (RVB)
- Pour les images en niveaux de gris, l'élément structurant peut contenir des valeurs réelles (les valeurs peuvent également être négatives)



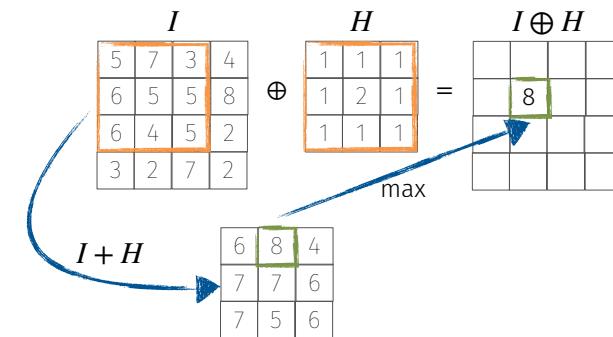
265

## Dilatation en niveaux de gris

Soit l'image en niveaux de gris  $I$  et l'élément structurant  $H$

La dilatation de  $I$  par  $H$  est définie par :

$$(I \oplus H)(u, v) = \max_{(i,j) \in H} \{I(u + i, v + j) + H(i, j)\}$$



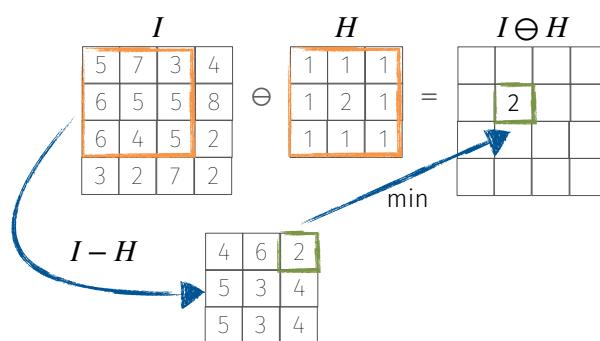
266

## Érosion en niveaux de gris

Soit l'image en niveaux de gris  $I$  et l'élément structurant  $H$

L'érosion de  $I$  par  $H$  est définie par :

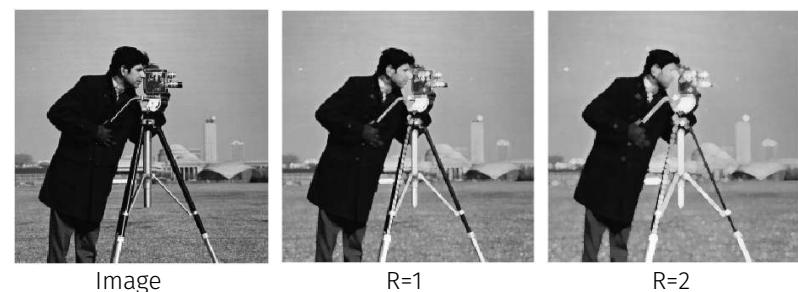
$$(I \ominus H)(u, v) = \min_{(i,j) \in H} \{I(u + i, v + j) - H(i, j)\}$$



267

## Exemple d'une dilatation en niveaux de gris

Dilatation par un disque de rayon  $R = 1$  et  $2$



268

## Exemple d'une érosion en niveaux de gris

Érosion par un disque de rayon  $R = 1$  et  $2$



Image



R=1



R=2



## Exemple d'une ouverture en niveaux de gris

Ouverture par un disque de rayon  $R = 1$  et  $2$



Image



R=1



R=2



## Ouverture et fermeture en niveaux de gris

Mêmes définitions que pour les images binaires



## Exemple de fermeture en niveaux de gris

Fermeture par un disque de rayon  $R = 1$  et  $2$



Image



R=1



R=2



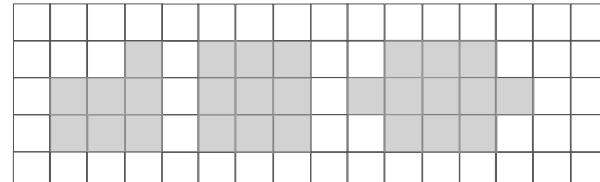
## Autres opérateurs utiles

273

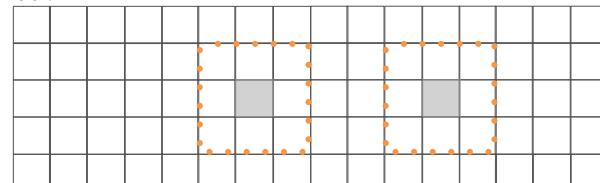
## Opérateur Hit-or-Miss

Permet de localiser des formes spécifiques dans les images

- Supposons que nous voulions localiser les formes carrées de taille 3x3 dans l'image ci-dessous



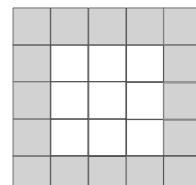
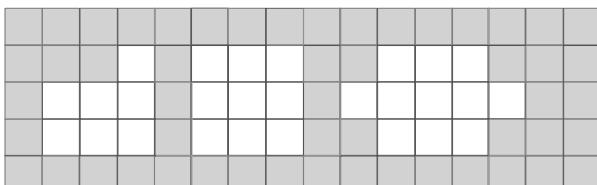
En réalisant une érosion  $A \ominus B$  avec  $B$  qui est la forme carrée recherchée :



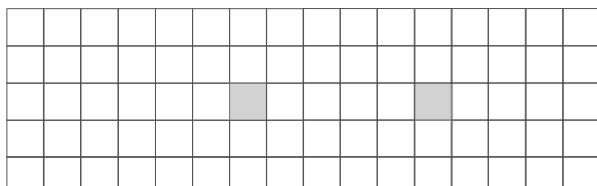
274

## Opérateur Hit-or-Miss

Si nous érodons le complémentaire de  $A$  avec un élément structurant  $C$  qui entoure la forme carrée recherchée



Le résultat de  $\bar{A} \ominus C$  est :



L'intersection de 2 érosions aboutie à 1 pixel au centre de la forme recherchée

275

## Opérateur Hit-or-Miss généralisé

Si nous recherchons une forme spécifique dans une image, nous créons 2 élément structurants :

- $B_1$  qui a la même forme que l'élément recherché
- $B_2$  dont la forme entoure celle que nous recherchons
- Notons  $B = (B_1, B_2)$

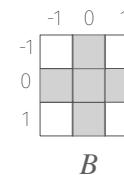
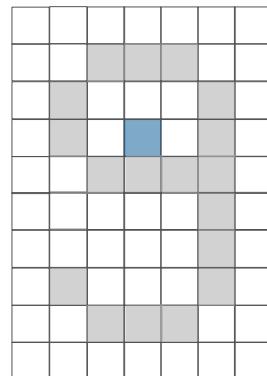
L'opérateur hit-or-miss s'écrit alors :

$$A \otimes B = (A \ominus B_1) \cap (\bar{A} \ominus B_2)$$

276

## Algorithme de remplissage de région

Supposons une image dont le contour est en 8-connectivité



*B*

Prenons un pixel *p* dans la région que nous souhaitons remplir

- Partons de *p* et dilatons autant que nécessaire à l'aide de l'élément structurant *B*

277

## Composante connexe

Un algorithme similaire est utilisé pour trouver les composantes connexes dans les images

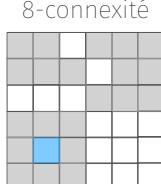
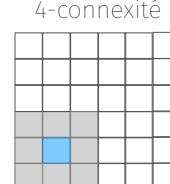
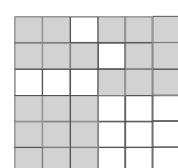
- Élément structurant en forme de croix (composantes 4-connectivité)
- Élément structurant de forme carrée (composantes 8-connectivité)
- Une séquence d'ensembles

$$X_0 = \{p\}, X_1, X_2, \dots$$

tel que

$$X_n = (X_{n-1} \oplus B) \cap A$$

jusqu'à  $X_k = X_{k-1}$



279

## Algorithme de remplissage de région

2. Après chaque dilatation, réalisons l'intersection avec  $\bar{A}$

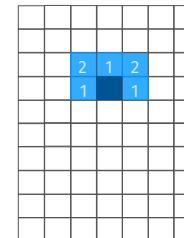
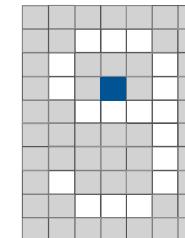
3. Puis créons la séquence :

$$\{p\} = X_0, X_1, \dots, X_k = X_{k+1}$$

avec :

$$X_n = (X_{n-1} \oplus B) \cap \bar{A}$$

On a alors  $X_k \cup A$  qui est la région une fois remplie



278

## Squelettisation

Table des opérations utilisées pour construire un squelette

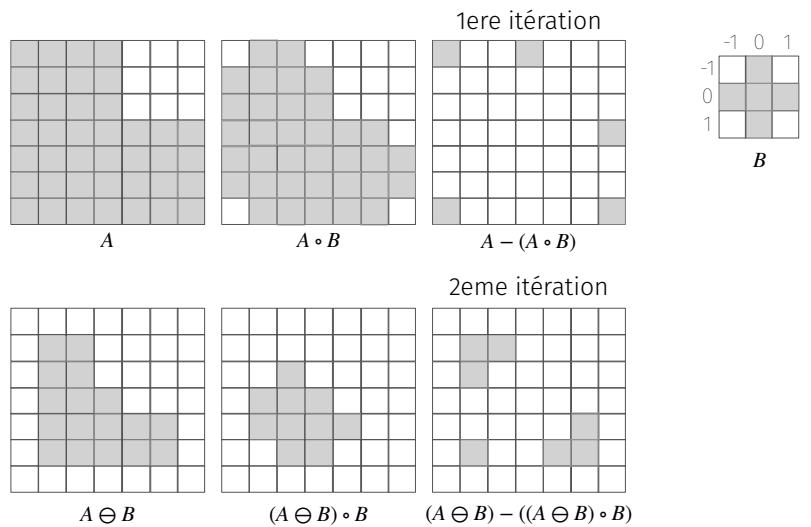
Erosions	Ouvertures	Dérences
$A$	$A \circ B$	$A - (A \circ B)$
$A \ominus B$	$(A \ominus B) \circ B$	$(A \ominus B) - ((A \ominus B) \circ B)$
$A \ominus 2B$	$(A \ominus 2B) \circ B$	$(A \ominus 2B) - ((A \ominus 2B) \circ B)$
$A \ominus kB$	$(A \ominus kB) \circ B$	$(A \ominus kB) - ((A \ominus kB) \circ B)$

Séquence de  $k$  érosions avec le même élément structurant :  $A \ominus kB$

- On continue la table jusqu'à ce que  $(A \ominus kB) \circ B$  soit vide
- Le squelette est l'union de toutes les différences entre ensembles

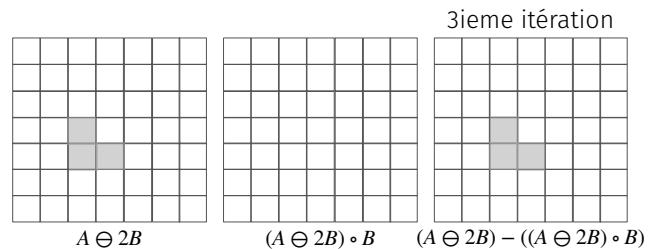
280

## Exemple de squelettisation

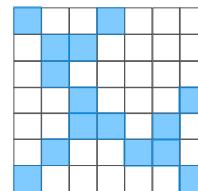


281

## Exemple de squelettisation

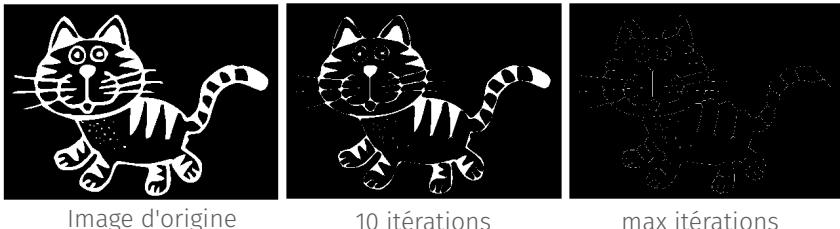


Le squelette est l'union des 3 squelettisations



282

## Exemples de squelettisation



The term watershed  
refers to a ridge that ...

... divides areas  
drained by different  
river systems.

The term watershed  
refers to a ridge that ...

... divides areas  
drained by different  
river systems.

The term watershed  
refers to a ridge that ...

... divides areas  
drained by different  
river systems.

Image d'origine

1 itération

max itérations

283

## Morphologie mathématique et Python

Le module **morphology** de la bibliothèque [scikit-image](#) fournit des opérateurs de morphologie mathématique pour les images binaires et niveaux de gris

284

## 7. Régions



*Trouver les régions*

### Introduction

- De nombreuses tâches nécessitent de reconnaître des objets sur des images en noir et blanc :
  - Du texte sur une page
  - Des objets sur une figure
  - Des structures dans des images issues d'un microscope
- Les images peuvent être en niveaux de gris, auquel cas elles nécessiteront une conversion en noir et blanc
- Les objets sont détectés en rassemblant les pixels en **groupes connexes**
- Chaque objet constitue alors une région binaire appelée **composante connexe**
  - Le type des objets peut ensuite être déterminé en le comparant à des modèles

### Trouver les régions d'une image binaire

#### Trouver

1. Quels pixels appartiennent à quelles régions ?
2. Combien de régions dans l'image ?
3. Où sont situées les régions ?

Ces tâches sont généralement effectuées lors de l'**étiquetage** des régions (attribution d'une étiquette pour les identifier)

#### 3 méthodes

- Remplissage par inondation
- Étiquetage séquentiel des régions
- Combinaison d'étiquetage de région + recherche de contour



## Exemple

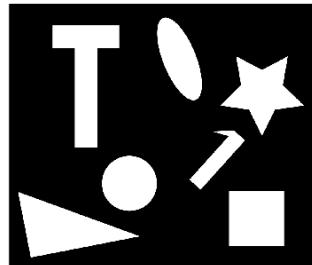


Image d'origine

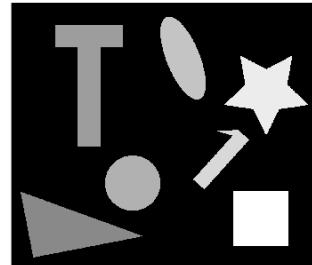


Image étiquetée

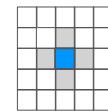
289

Trouver les régions : remplissage par inondation

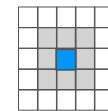
291

## Trouver les régions d'une image binaire

1. Fixons un voisinage 4-voisins ( $\mathcal{N}_4$ ) ou 8-voisins ( $\mathcal{N}_8$ )



$\mathcal{N}_4$



$\mathcal{N}_8$

2. Adoptons la convention suivante :

$$I(u, v) = \begin{cases} 0 & \text{pixel du fond} \\ 1 & \text{pixel du premier plan} \\ 2, 3, \dots & \text{étiquettes des régions} \end{cases}$$

290

## Étiquetage de région par la technique de l'inondation

- Chercher les pixels non étiquetés du premier plan et les étiqueter
- 3 versions :
  - Récursive (adaptée aux petites images)
  - Par profondeur
  - Par largeur
- La méthode est appelée par l'algorithme d'étiquetage des régions

```
EtiqueterRegion(I: image binaire)
étiquette = 2
Itérer sur toutes les coordonnées (u, v) de l'image
Si I(u, v) = 1 Alors
    RemplirParInondation(I, u, v, étiquette)
    étiquette = étiquette + 1
Retourner I
```

292

## Remplissage par inondation en profondeur

- Les pixels non visités sont stockés sur une pile
- Le parcours de l'arbre des pixels se fait en profondeur

```
RemplirParInondation(I, u, v, etiquette)
Créer un pile vide S
déposer(S, (u,v))
TantQue S n'est pas vide Faire
    (x, y) = relever(S)
    Si (x, y) est sur le bord de l'image et I(x, y) = 1 Alors
        I(x, y) = etiquette
        déposer(S, (x+1, y))
        déposer(S, (x, y+1))
        déposer(S, (x, y-1))
        déposer(S, (x-1, y))
    Retourner
```

293

## Remplissage par inondation en largeur

- Similaire à la version en profondeur
- Les pixels non encore visités sont stockés dans une file d'attente

```
RemplirParInondation(I, u, v, etiquette)
Créer une file vide Q
enfiler(Q, (u,v))
TantQue Q n'est pas vide Faire
    (x, y) = defiler(Q)
    Si (x, y) est sur le contour de l'image et I(x, y) = 1 Alors
        I(x, y) = etiquette
        enfiler(Q, (x+1, y))
        enfiler(Q, (x, y+1))
        enfiler(Q, (x, y-1))
        enfiler(Q, (x-1, y))
    Retourner
```

294

## Trouver les régions : étiquetage séquentiel

295

## Étiquetage séquentiel de région

### 2 étapes

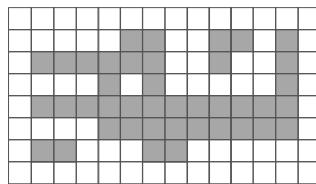
1. Étiquetage préliminaire des régions de l'image
2. Résoudre les cas où un pixel a plus d'une étiquette

- Le test des pixels dépend du voisinage considéré (4-voisinage ou 8-voisinage)



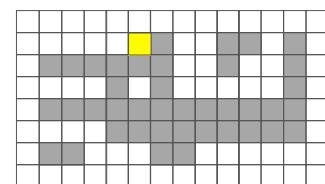
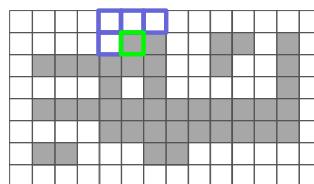
296

## Étiquetage séquentiel de région : propagation des étiquettes

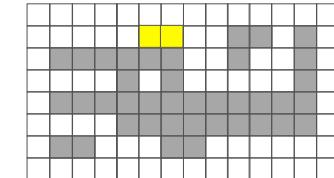


	0
■	1
■	2
■	3
■	4
■	5
■	6
■	7

- Le 1er pixel de premier plan est trouvé ( $\mathcal{N}_8(u, v)$ )
- Tous ses voisins ( $\mathcal{N}_8(u, v)$ ) sont considérés comme appartenant au fond
- Alors on assigne au pixel la première étiquette (2)



297

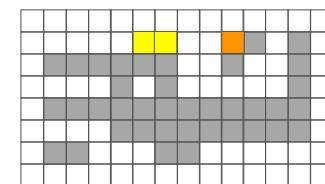
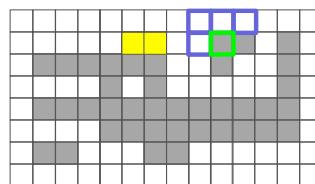


Pour une seule étiquette différente dans le voisinage, on conserve la même étiquette

298

## Étiquetage séquentiel de région : propagation des étiquettes

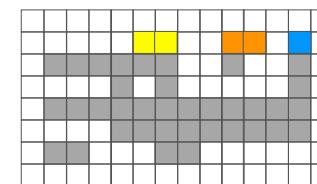
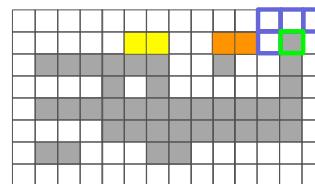
- On continue à tester les pixels comme précédemment
- À l'étape ci-dessous, il n'y a aucune étiquette préalablement définie dans le voisinage du pixel, dont nous incrémentons la valeur de l'étiquette (valeur 3 ici)



299

## Étiquetage séquentiel de région : propagation des étiquettes

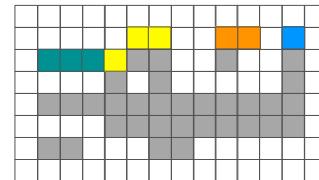
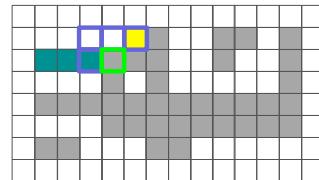
- On continue à tester les pixels comme précédemment
- À l'étape ci-dessous, il n'y a aucune étiquette préalablement définie dans le voisinage du pixel, dont nous incrémentons la valeur de l'étiquette (valeur 4 ici)



300

## Étiquetage séquentiel de région : propagation des étiquettes

- On continue à tester les pixels comme précédemment
- À l'étape ci-dessous, il existe 2 pixels voisins avec des étiquettes différentes (2 et 5)
- Une des deux valeurs est propagée, et un indicateur de collision est enregistré

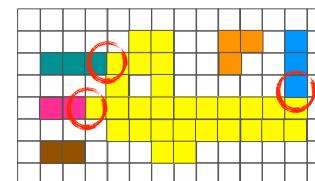


301

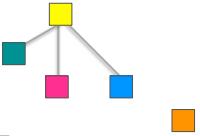
## Étiquetage séquentiel de région : gestion des collisions

À la fin de l'étape d'étiquetage :

- Tous les pixels d'avant-plan sont étiquetés
- Toutes les collisions entre étiquettes ont été enregistrées
- Les étiquettes et les collisions correspondent au noeuds d'un graphe non orienté



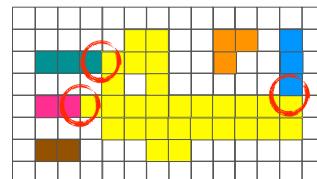
Graphe non-orienté des collisions



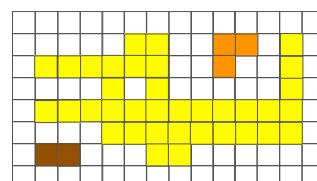
302

## Étiquetage séquentiel de région : résoudre les collisions

- C'est l'étiquette de plus petite valeur (ici 2) qui va être choisie lorsque plusieurs étiquettes sont en confrontation



Graphe non-orienté des collisions



303

## Propriétés des régions binaires

304

## Propriétés des régions binaires

Un homme décrira des régions en fonction de leurs propriétés. Par exemple : "Un T-shirt khaki de taille XL qui a une décoration jaune et est posé sur une table en chêne"



- Pour un ordinateur, une telle description n'est pas encore possible
- Toutefois, il pourra calculer un certain nombre de propriétés pour les différentes régions dans une image, ce afin de permettre une classification

305

## Types de caractéristiques

- Caractéristiques sur la forme
- Caractéristiques géométriques

306

## Caractéristiques d'une forme

- **Caractéristique** : valeur numérique ou qualitative obtenue à partir des valeurs et des coordonnées des pixels (ex. : la taille est le nombre total de pixels de la région)

### - Vecteur de caractéristiques :

- Combinaison de différentes caractéristiques
- Utilisé comme une "signature" de la région dans le but de classer ou comparer
- Les caractéristiques doivent être :
  - simples à calculer
  - non affectées par une translation, une rotation ou une mise à l'échelle

307

## Caractéristiques géométriques

Soit une région  $R$  dans une image binaire

La région correspond à la distribution de points de l'avant-plan dans un plan discret

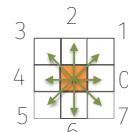
308

## Caractéristiques géométriques : périmètre

- **Périmètre** : longueur du contour externe de la région ( $R$  doit être connectée !)
- Pour un 4-voisinage, la mesure de la longueur est supérieure à la longueur réelle
- Approximation pour un 8-voisinage :  $C'_R = [C'_0, C'_1, \dots, C'_{M-1}]$

$$\text{Périmètre}(R) = \sum_{i=0}^{M-1} \text{longueur}(C'_i)$$

avec  $\text{longueur}(C) = \begin{cases} 1 & \text{pour } c = 0, 2, 4, 6 \\ \sqrt{2} & \text{pour } c = 1, 3, 5, 7 \end{cases}$



La formule précédente conduit à un surestimation (multiplier le résultat par 0,95)



## Caractéristiques géométriques : surface

- **Surface** : nombre des pixels de l'image qui sont dans la région

$$\text{Surface}(R) = |R| = N$$

- **Surface d'une région connexe** (sans trous) : définie par les  $M$  coordonnées des points est estimée à l'aide de la formule :

$$\text{Surface}(R) \simeq \frac{1}{2} \left| \sum_{i=0}^{M-1} (u_i \cdot v_{(i+1)\text{mod}M} - u_{(i+1)\text{mod}M} \cdot v_i) \right|$$

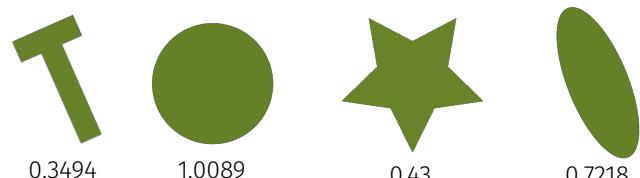
## Caractéristiques géométriques : compacité et circularité

- **Compacité** : rapport entre la surface d'une région et son périmètre

$$\text{Compacité}(R) = \frac{\text{Surface}(R)}{\text{Périmètre}^2(R)}$$

- Elle est invariante en translation, en rotation et en mise à l'échelle
- La compacité est de  $1/4\pi$  pour un disque

$$\text{Circularité}(R) = 4\pi \cdot \frac{\text{Surface}(R)}{\text{Périmètre}^2(R)}$$



311

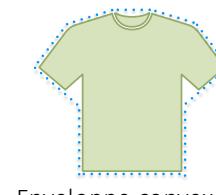
## Caractéristiques géométriques : boîte englobante et enveloppe convexe

- **Boîte englobante** : rectangle minimal qui contient tous les points de la région  $R$

- **Enveloppe convexe** : plus petit polygone qui entoure tous les points de la région  $R$

- Convexité : relation entre la longueur de l'enveloppe convexe et le périmètre de  $R$

- Densité : rapport entre la surface de la région et celle de l'enveloppe convexe



312

## Propriétés statistiques de la forme

- Les points d'une région peuvent aussi être vus comme statistiquement distribués sur un plan
- Ces propriétés ne nécessitent pas que les régions soient connectées
- Exemple : le **centre de masse** d'une région binaire est la moyenne arithmétique de toutes les coordonnées  $(x, y)$  des points de la région

$$\bar{x} = \frac{1}{|R|} \sum_{(u,v) \in R} u \quad \text{et} \quad \bar{y} = \frac{1}{|R|} \sum_{(u,v) \in R} v$$

313

## Propriétés statistiques de la forme

- De la même façon, le centre de masse peut être exprimé par :

$$\bar{x} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} u^1 v^0 = \frac{m_{10}(R)}{m_{00}(R)}$$
$$\bar{y} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} u^0 v^1 = \frac{m_{01}(R)}{m_{00}(R)}$$

315

## Propriétés statistiques de la forme

- Le centre de masse est un cas particulier du concept général de **moment**
- Le **moment ordinaire** d'ordre  $p, q$  pour une fonction discrète  $I(u, v)$  est défini par :

$$m_{pq} = \sum_{(u,v) \in R} I(u, v) \cdot u^p v^q$$

- La surface d'une région binaire correspond au **moment d'ordre 0**

$$\text{Surface}(R) = m_{00}(R) = \sum_{(u,v) \in R} 1 = \sum_{(u,v) \in R} u^0 v^0 = |R|$$

314

## Propriétés statistiques de la forme

### Moments centrés

- Le centre de masse est utilisé comme référence (nouvelle origine) pour calculer des caractéristiques invariantes en translation
- Les moments centrés d'ordre  $p, q$  sont calculés par :

$$\mu_{pq} = \sum_{(u,v) \in R} I(u, v) \cdot (u - \bar{x})^p (v - \bar{y})^q$$

- Pour une image binaire ( $I(u, v) = 1$ ) :

$$\mu_{pq} = \sum_{(u,v) \in R} (u - \bar{x})^p (v - \bar{y})^q$$

316

## Propriétés statistiques de la forme

- La valeur des moments centrés dépend :
  - Des distances de tous les points de la région par rapport au centre de masse
  - De la taille absolue de la région
  - L'invariance par rapport à la taille sera obtenu en normalisant les moments centrés à l'aide d'un facteur  $k$

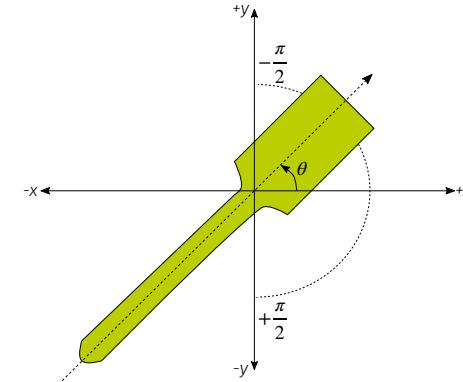
- Les **moments centrés normalisés** sont donnés par :

$$\bar{\mu}_{pq}(R) = \mu_{pq} \cdot \left(\frac{1}{\mu_{00}(R)}\right)^{(p+q+2)/2} \quad \text{pour } (p + q) \geq 2$$

317

## Propriétés géométriques basées sur les moments

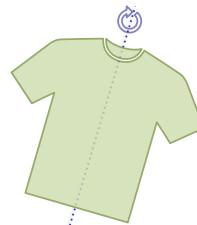
- De nombreuses caractéristiques sont associées aux moments
- **Orientation** : décrit la direction de l'axe principal (passe par le centre de masse et le long de la partie la plus large de la région)



318

## Propriétés géométriques basées sur les moments

- L'**orientation** est aussi appelée l'**axe majeur de rotation**, car c'est suivant cet axe que la rotation pourra se faire avec le minimum d'effort



- La direction de l'axe majeur est calculée ainsi :

$$\theta_R = \frac{1}{2} \tan^{-1} \left( \frac{2\mu_{11}(R)}{\mu_{20}(R) - \mu_{02}(R)} \right) \quad \text{intervalle } [-90, 90]$$

319

## Propriétés géométriques basées sur les moments

- L'affichage de l'orientation peut se faire en utilisant l'équation paramétrique d'une droite

$$\mathbf{x} = \bar{\mathbf{x}} + \lambda \cdot \mathbf{x_d} = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} + \lambda \cdot \begin{pmatrix} \cos(\theta_R) \\ \sin(\theta_R) \end{pmatrix}$$

point de départ      vecteur direction

- $\mathbf{x_d}$  peut être calculé :

$$\mathbf{x_d} = \cos(\theta_R) = \begin{cases} 0 & \text{pour } a = b = 0 \\ \frac{1}{2}(1 + \frac{b}{\sqrt{a^2 + b^2}})^{\frac{1}{2}} & \text{sinon.} \end{cases}$$

$$\mathbf{y_d} = \sin(\theta_R) = \begin{cases} 0 & \text{pour } a = b = 0 \\ \frac{1}{2}(1 - \frac{b}{\sqrt{a^2 + b^2}})^{\frac{1}{2}} & \text{pour } a \geq 0 \\ -[\frac{1}{2}(1 - \frac{b}{\sqrt{a^2 + b^2}})^{\frac{1}{2}}] & \text{pour } a < 0 \end{cases}$$

où  $a = 2\mu_{11}(R)$  et  $b = \mu_{20}(R) - \mu_{02}(R)$

320

## Propriétés géométriques basées sur les moments

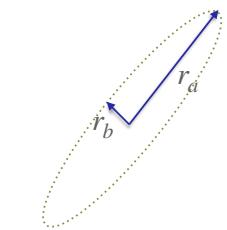
- **Excentricité** : rapport des longueurs de l'axe majeur et de l'axe mineur
- Exprime l'élongation d'une région

$$\text{Excentricité}(R) = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} + \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}} = \frac{a_1}{a_2}$$

- Longueurs des axes majeur et mineur :

$$r_a = 2 \cdot \left( \frac{2a_1}{|R|} \right)^{\frac{1}{2}}$$

$$r_b = 2 \cdot \left( \frac{2a_2}{|R|} \right)^{\frac{1}{2}}$$



321

## Propriétés géométriques basées sur les moments

- Les moments centrés normalisés sont invariants par translation et mise à l'échelle, mais ne le sont pas par rotation
- Les **moments de Hu** (7 combinaisons) sont **invariants** par translation, mise à l'échelle et rotation

$$H_1 = \bar{\mu}_{20} + \bar{\mu}_{02}$$

$$H_2 = (\bar{\mu}_{20} - \bar{\mu}_{02})^2 + 4\bar{\mu}_{11}^2$$

$$H_3 = (\bar{\mu}_{30} - 3\bar{\mu}_{12})^2 + (3\bar{\mu}_{21} - \bar{\mu}_{03})^2$$

$$H_4 = (\bar{\mu}_{30} + \bar{\mu}_{12})^2 + (\bar{\mu}_{21} + \bar{\mu}_{03})^2$$

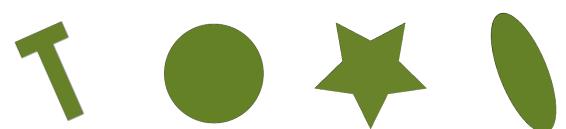
$$H_5 = (\bar{\mu}_{30} - 3\bar{\mu}_{12}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2] \\ + (3\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot [3(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2]$$

$$H_6 = (\bar{\mu}_{20} - \bar{\mu}_{02}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2] + 4\bar{\mu}_{11} \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03})$$

$$H_7 = (3\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2] \\ + (3\bar{\mu}_{12} - \bar{\mu}_{30}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot [3(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2]$$

323

## Exemples



Surface	54903	34822	41743	33433
Orientation	90°	14.84°	-60.92°	-68°
Circularité	0.3494	1.0089	0.43	0.7218
Longueur axe mineur	201	210	250	126
Longueur axe majeur	571	210	250	336
Excentricité	0.9356	0.0265	0.0354	0.9269

322

## Projections

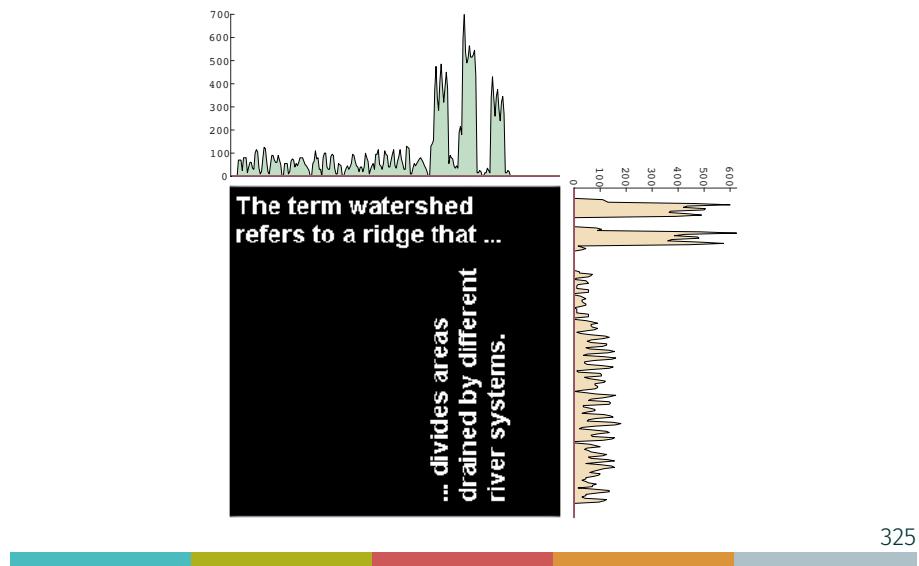
- Les projections sont des représentations 1D du contenu des images
- Les projections verticale et horizontale de  $I(u, v)$  sont définies comme :

$$P_{\text{horizontal}}(v_0) = \sum_{u=0}^{M-1} I(u, v_0) \quad \text{pour } 0 < v_0 < N$$

$$P_{\text{vertical}}(u_0) = \sum_{v=0}^{N-1} I(u_0, v) \quad \text{pour } 0 < u_0 < M$$

324

## Exemple de projections



325

## Propriétés topologiques

- Définissent la structure d'une région
- Invariantes par rapport aux transformations fortes sur l'image
- Le **nombre de trous** est une caractéristique simple et robuste
- Le **nombre d'Euler** : nombre de composantes connexes – nombre de trous

$$N_E(R) = N_{CC}(R) - N_H(R)$$

- Les caractéristiques topologiques sont souvent combinées avec des caractéristiques géométriques

326

## Les régions et Python

- Le module **measure** de la bibliothèque [scikit-image](#) supporte la plupart des méthodes utilisées ici
- Les modules **filters**, **segmentation** et **morphology** proposent aussi quelques fonctions intéressantes

327