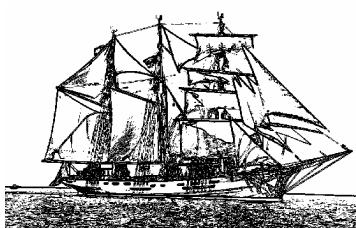
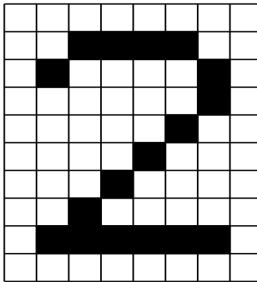


# MAP201 : Découverte des Mathématiques Appliquées

Images

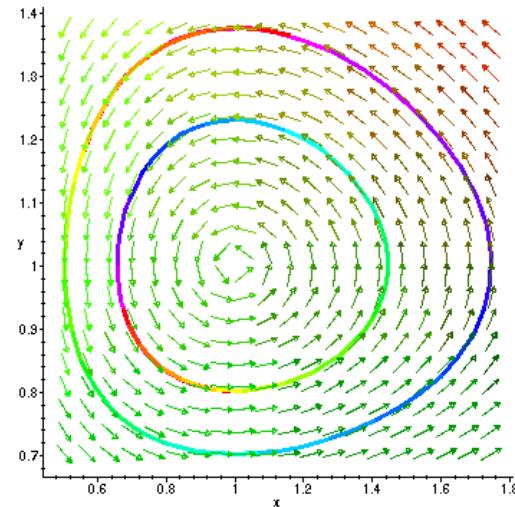
1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1		
1	0	1	1	1	1	0	1		
1	1	1	1	1	1	0	1		
1	1	1	1	1	1	0	1		
1	1	1	1	1	1	0	1		
1	1	1	1	1	0	1	1		
1	1	1	1	0	1	1	1		
1	1	0	1	1	1	1	1		
1	0	0	0	0	1	1			
1	1	1	1	1	1	1	1		



Jérôme Lesaint

Equa Diff

Modèle de Lotka–Volterra



Mickael Nahon

# Plan de la séance d'aujourd'hui

- 1. Organisation de la partie Image
- 2. Image et Mathématiques Appliquées
- 3. Début du cours sur l'image

# Déroulement des TP d'image

- 3h / semaine, 6 séances
- 5 TPs progressifs
- 1 TP noté (TP4)
- 1 possibilité de bonus (TP2)
- TP à rendre impérativement avant la séance de TP suivante.

Semaine TP	TP	Thème
3 (15 janvier)	TP-0	Intro Python
4 (22 janvier)	TP-1	Les bases
5 (29 janvier)	TP-2	Histogrammes
6 (5 février)	TP-3	Filtrage
7 (12 février) 8 (19 février)	TP-4	Détection de contours

# Plan de la séance d'aujourd'hui

- 1. Organisation du module
- 2. Image et Mathématiques Appliquées
- 3. Début du cours sur l'image

# Le “démon” de Laplace (1814)

« Une intelligence qui, à un instant donné, connaît toutes les forces dont la nature est animée et la situation respective des êtres qui la composent, si d'ailleurs elle était suffisamment vaste pour soumettre ces données à l'analyse, embrasserait dans la même formule les mouvements des plus grands corps de l'univers et ceux du plus léger atome ; rien ne serait incertain pour elle, et l'avenir, comme le passé, serait présent à ses yeux. »

Pierre-Simon Laplace, Essai philosophique sur les probabilités (1814)

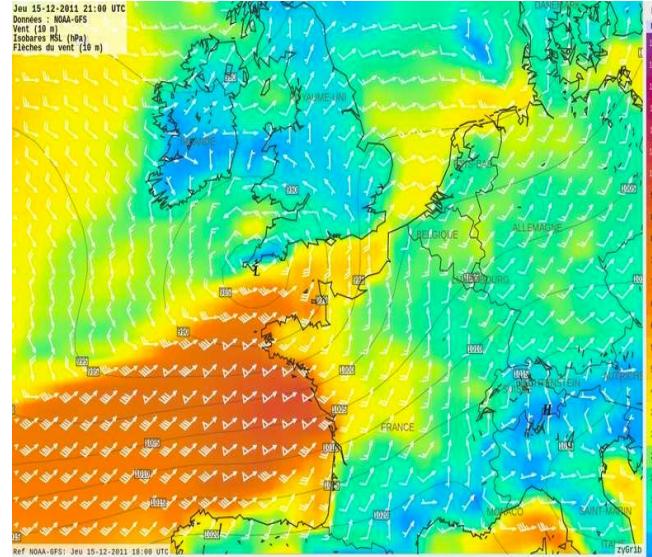
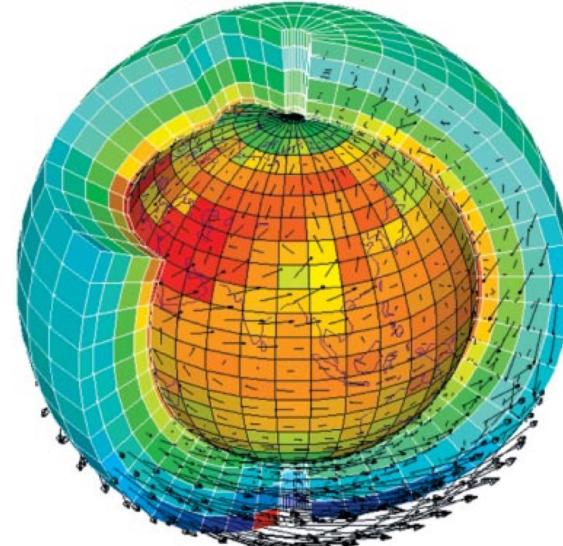
Irréaliste mais proche des problématiques actuelles des maths appli :

- Décrire le réel avec des objets mathématiques : **Modélisation**
- **Résolution des calculs** : complexité à la mesure du réel ⇒ numérique (sur ordinateur)
- Outil pour **comprendre, prédire**

# Un exemple aujourd'hui : les prévisions météo

- Résolution numérique par discréétisation

- ▶ Maillage 3d → Calcul numérique
- ▶ Le problème reste complexe : plusieurs millions de degrés de variables
- ▶ Prédiction avec une marge d'erreur et sur une durée limitée...



# Dans le cadre de ce cours

- Traitement et analyse d'images:
  - ▶ Décrire les images en tant qu'objet mathématique
  - ▶ Elaboration d'algorithmes sur les images numériques : pour permettre un traitement informatique
  - ▶ Mise en oeuvre dans des codes informatiques

# **Plan de la séance d'aujourd'hui**

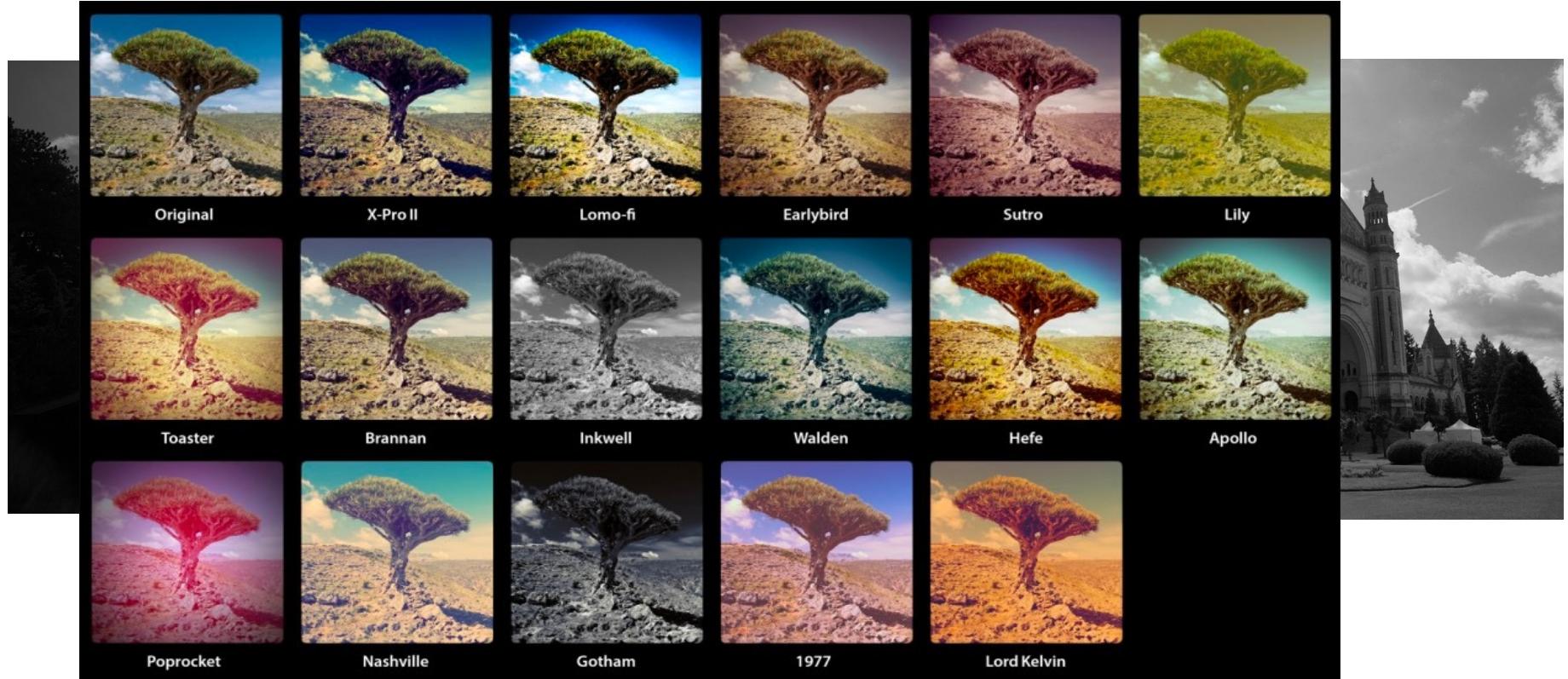
- 1. Organisation du module

2. Introduction sur les Mathématiques Appliquées

3. Début du cours sur l'image

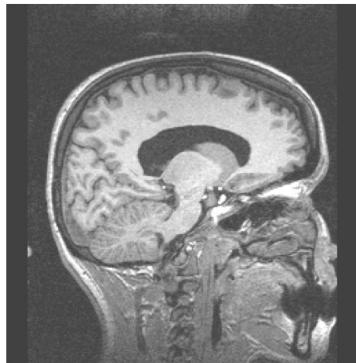
# Traitement d'images : motivations

- Retouche photo



# Traitement d'images : motivations

- Extraction d'information (mesure, métrologie)



Courtesy of D. Vandermeulen



Fluide  
Céphalo-rachidien



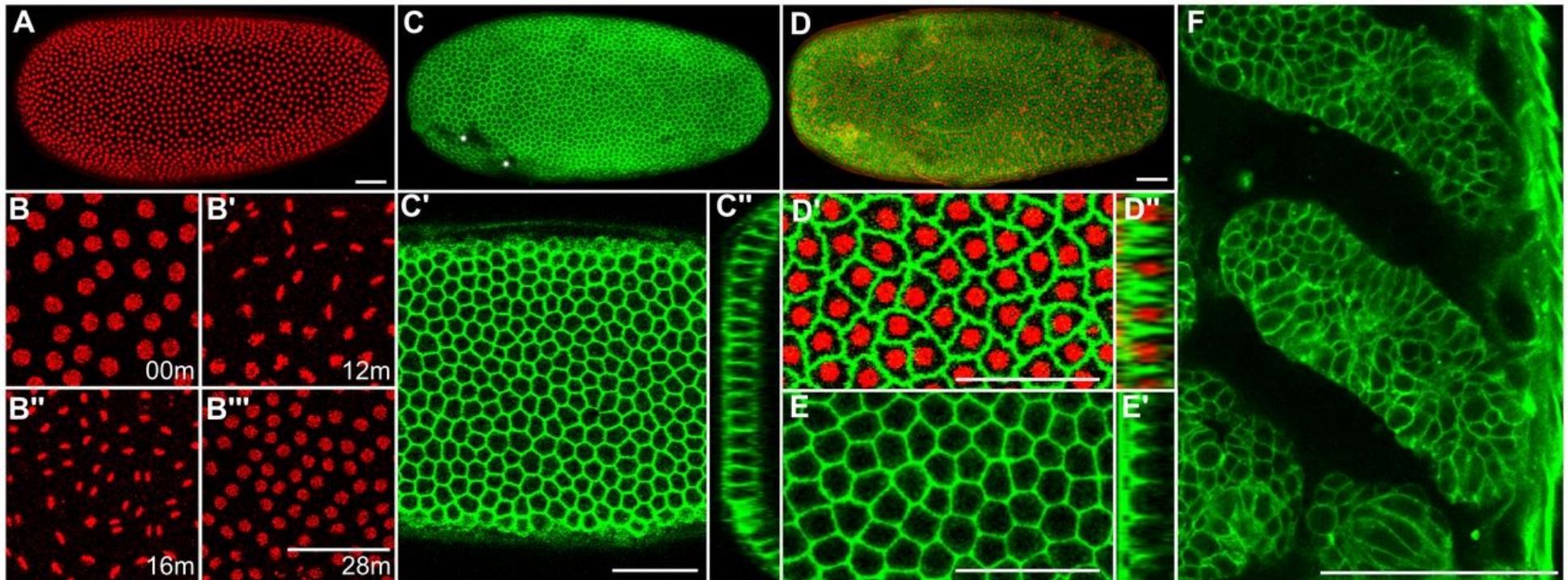
Matière  
Blanche



Matière  
Grise

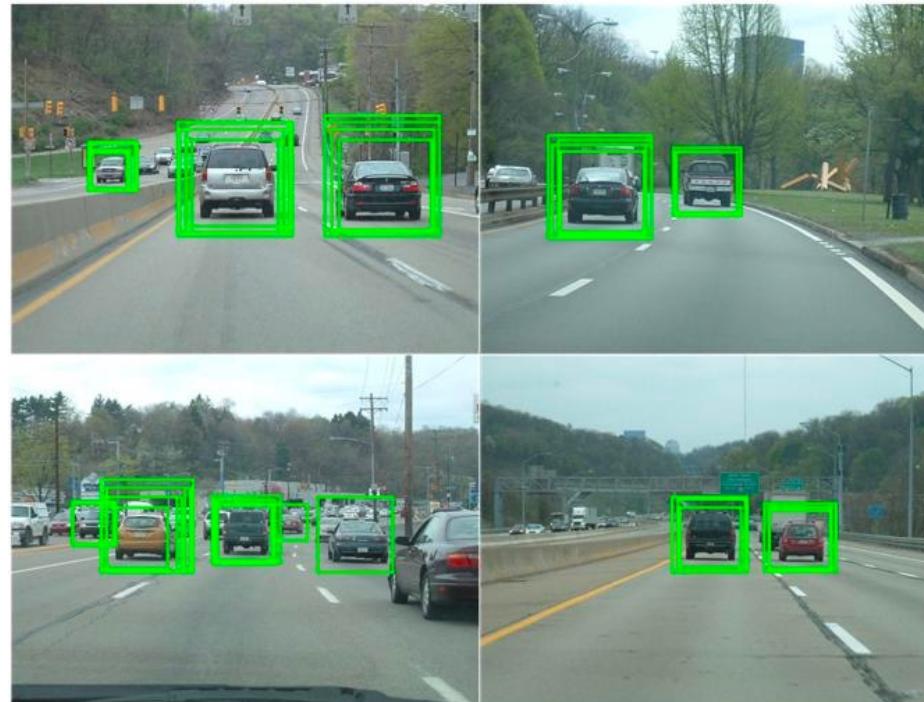
# Traitement d'images : motivations

- Extraction d'information (mesure, métrologie)



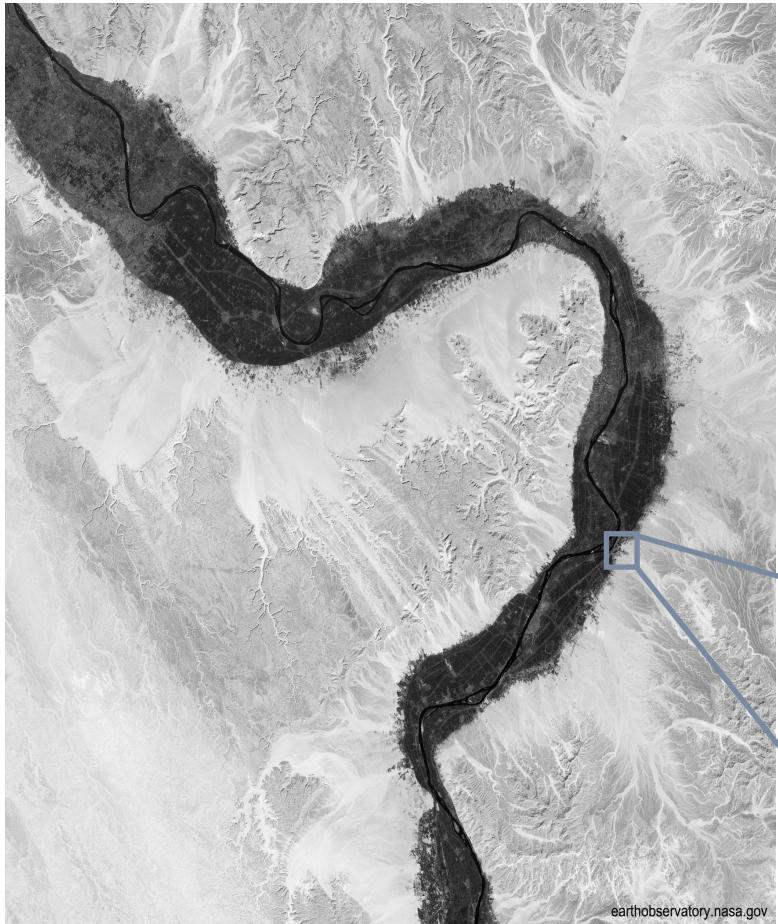
# Traitement d'images : motivations

- Extraction d'information (vision)
  - ▶ Voitures autonomes, robotiques : traitement d'image + IA



# 1er Cours: LES IMAGES NUMÉRIQUES

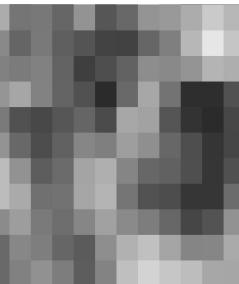
# Qu'est-ce qu'une image numérique?



Images en provenance de capteurs

- ▶ Photo, scanner, microscope, ...

Fonction mathématique discrétisée



# Images “continues”



$$I : \begin{cases} \mathbb{R}^2 & \rightarrow [0, 1] \\ (x, y) & \mapsto I(x, y) \end{cases}$$

$I(x, y)$  intensité



Image en “niveaux de gris”

Définie par une fonction d'intensité

# Exemple

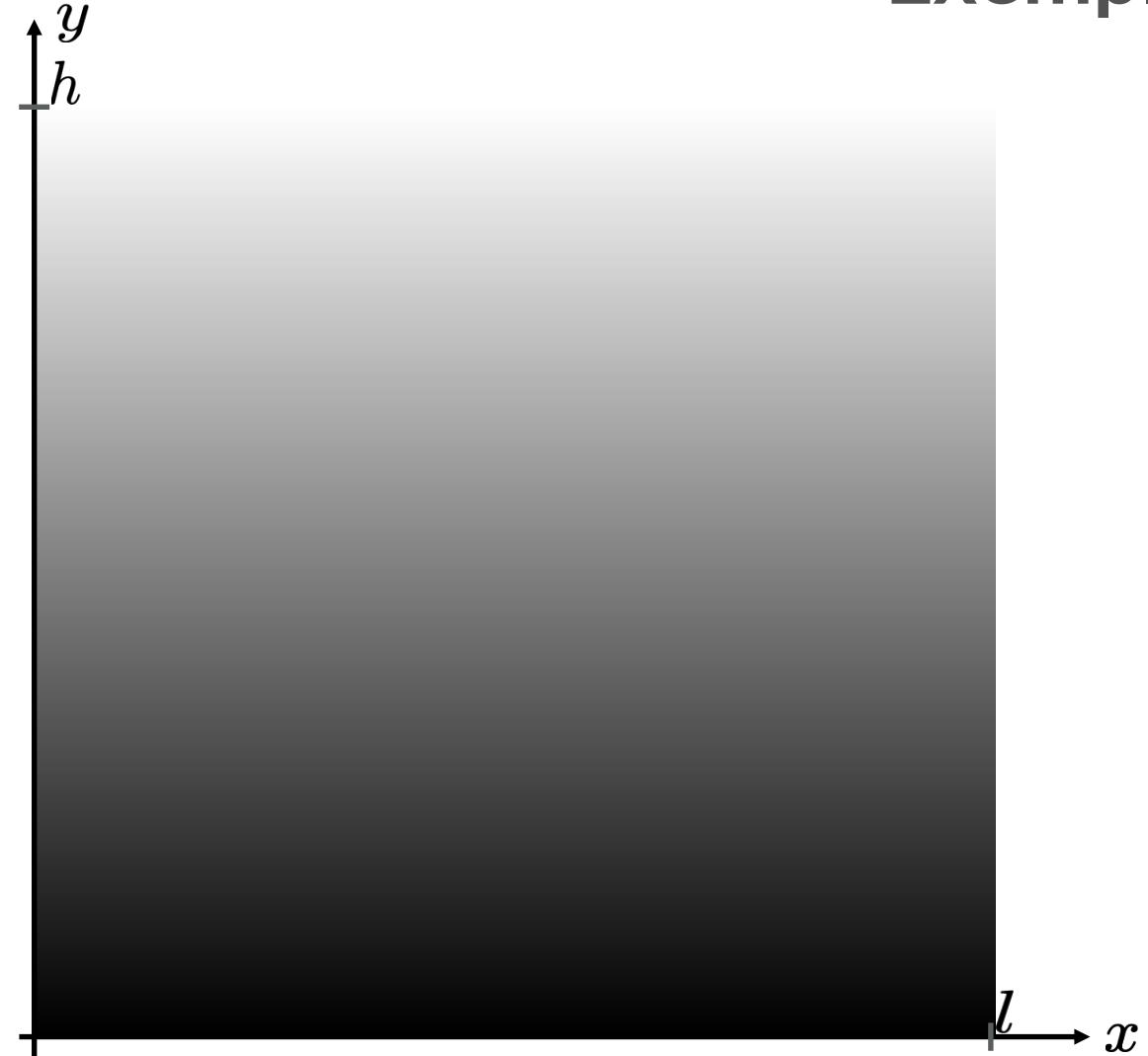
$$I : \begin{cases} \mathbb{R}^2 & \rightarrow [0, 1] \\ (x, y) & \mapsto I(x, y) \end{cases}$$



$$I(x, y) = \frac{x}{l}$$

$x$

# Exemple

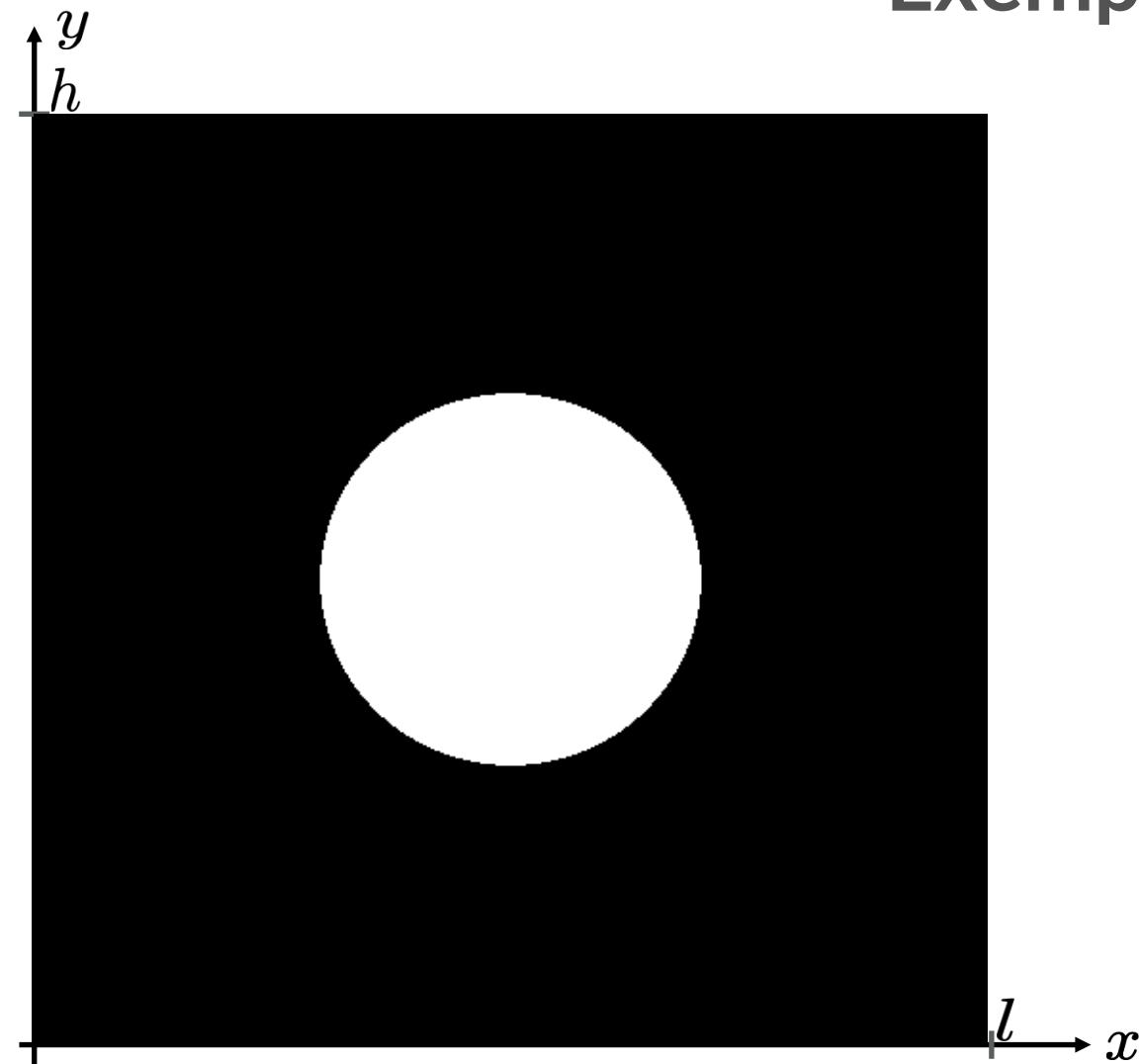


$$I : \begin{cases} \mathbb{R}^2 & \rightarrow [0, 1] \\ (x, y) & \mapsto I(x, y) \end{cases}$$



$$I(x, y) = \frac{y}{h}$$

# Exemple



$$I : \begin{cases} \mathbb{R}^2 & \rightarrow [0, 1] \\ (x, y) & \mapsto I(x, y) \end{cases}$$

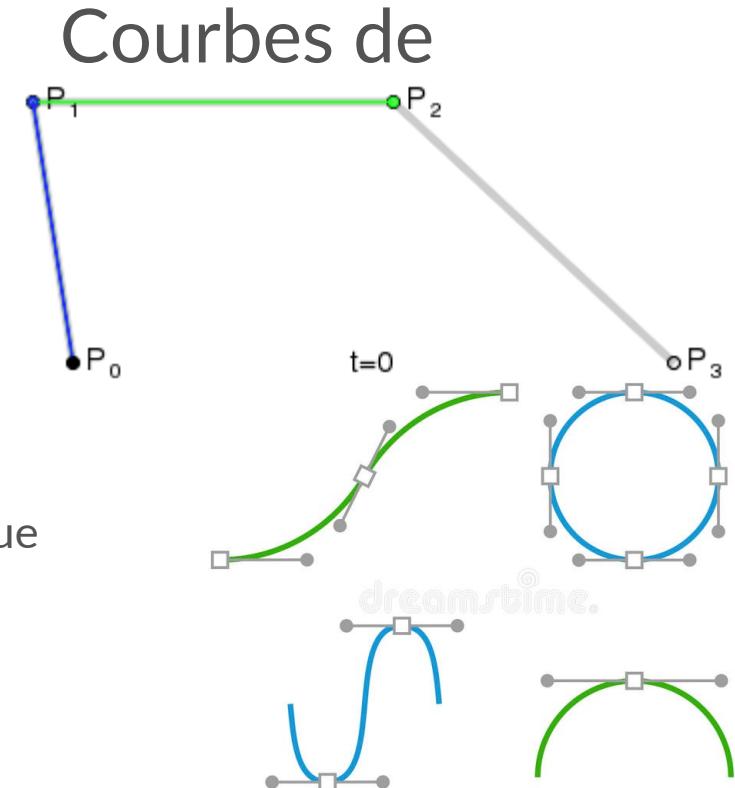


$$I(x, y) = \begin{cases} 1 & si\ r \leq 1 \\ 0 & sinon \end{cases}$$

$$avec r = \sqrt{(x - l/2)^2 + (y - h/2)^2}$$

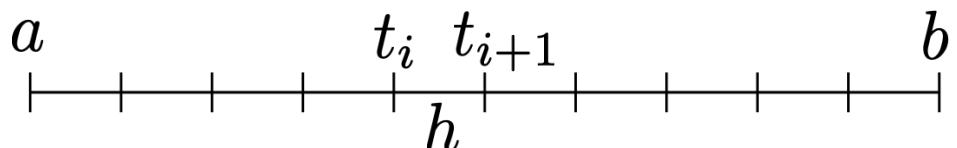
# Images vectorielles

- Utiles pour représenter des formes géométriques
  - ▶ Logos, polices TrueType
  - ▶ PDF, Images SVG
  - ▶ Canvas HTML5
  - ▶ Rendu 3D Blender
- Codées par “éléments de base”
  - ▶ Lignes, cercles, sphères
  - ▶ “arrondis” stockés comme courbes de Bézier
  - ▶ Redimensionnable sans perte de qualité car l’information de dépend pas de la position absolue mais relative des éléments



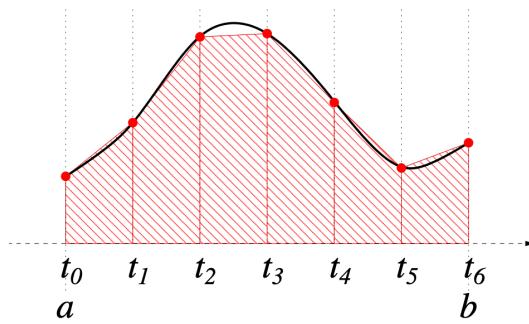
# Images numériques

- Besoin de discréteriser pour représenter/traiter
  - ▶ Découper un domaine continu
  - ▶ Découpage régulier : soit  $I = [a, b]$  un interval et  $n$  un entier naturel. On peut découper l'intervalle  $I$  en  $n$  intervalles  $I_i$  de taille  $h = \frac{b-a}{n}$  avec
    - ▶  $I_i = [t_{i-1}, t_i], i = 1, \dots, n$
    - ▶  $t_i = a + ih, i = 0, \dots, n$
  - ▶ Découpage irrégulier (peu utilisé en image)



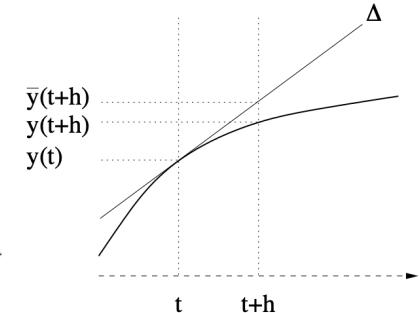
# Applications de la discrétisation au calcul numérique

- ▶ Approcher avec algorithmes la solution d'un problème mathématique
- ▶ Calcul d'intégrales
- ▶ Méthode d'Euler pour la résolution d'équation différentielles



$$\begin{cases} y'(t) = f(t) \\ y(t_{\min}) = y_0 \end{cases}$$

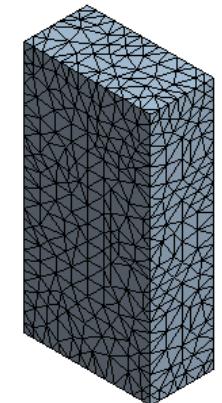
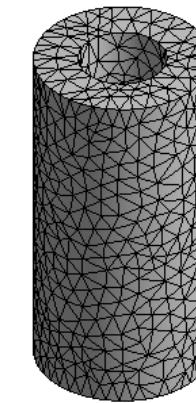
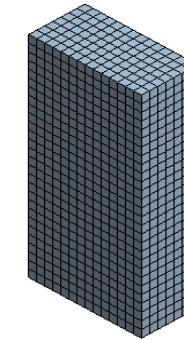
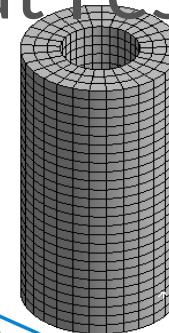
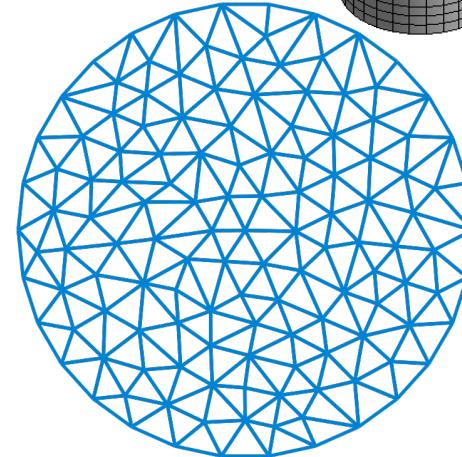
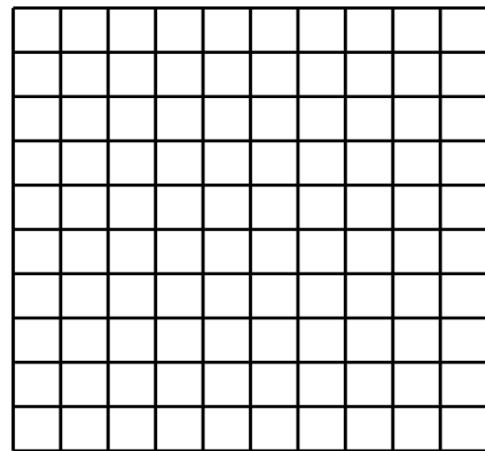
$$y'(t) = f(t) \simeq \frac{y(t+h) - y(t)}{h}$$



```
h ← (tmax − tmin)/n // calcul du pas h
// initialisation
tt(1) ← tmin
yy(1) ← y0
pour k de 1 à n faire
    tt(k + 1) ← tt(k) + h
    yy(k + 1) ← yy(k) + h (f(tt(k)) − a(tt(k)) yy(k))
fin_pour
```

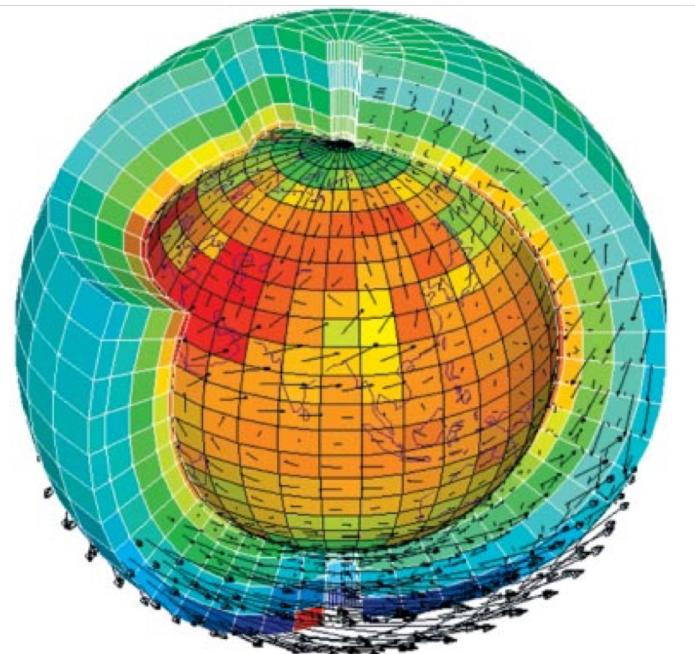
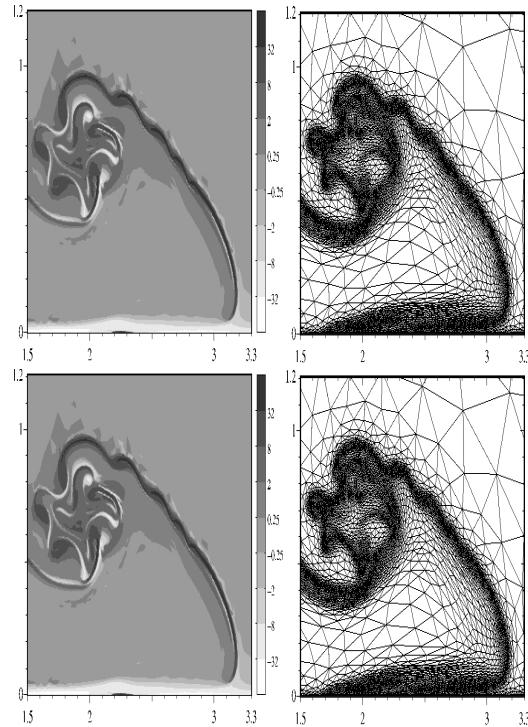
# Discrétisation en 2D/3D

- Le principe reste le même mais il faut faire attention à couvrir tout l'espace (pas de trous!)



# Applications de la discréétisation 2D/3D

- De nombreux problèmes physiques sont multi-dimensionnels
  - ▶ Mêmes problèmes qu'en 1D (résolution d'ED, intégrales, etc.)



# • Discrétisation régulière de l'espace

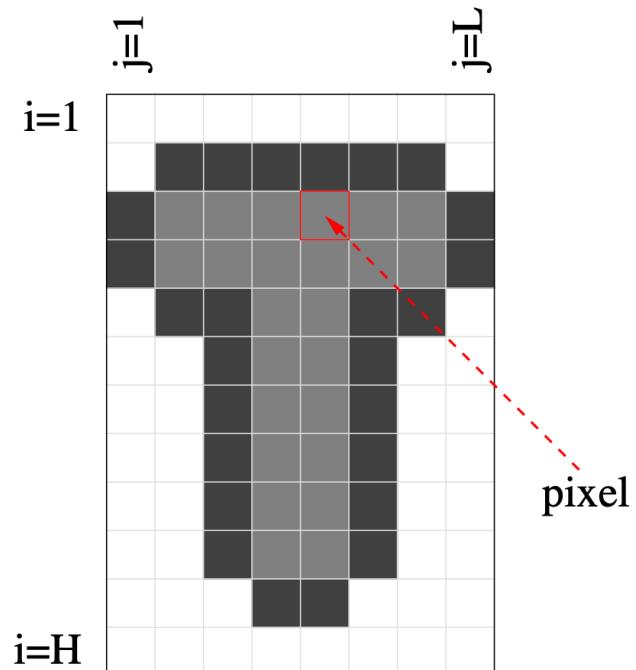
- ▶ Capteurs → ordinateurs → écrans
- ▶  $i, j$  indices lignes colonnes (pas un repère cartésien  $(x, y)$ )

Image rectangulaire découpée suivant une grille régulière

Elément de la grille :  
**PICTure ELelement - pixel**

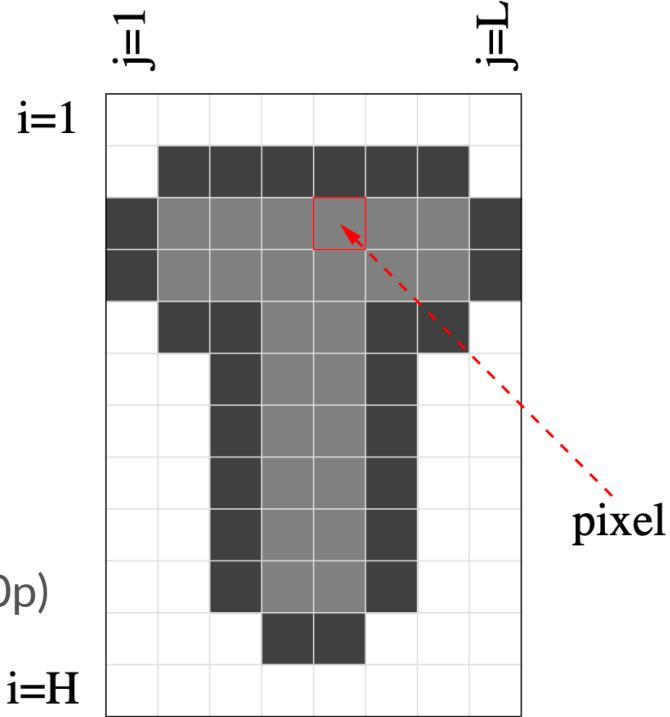
**Image numérique** :  
tableau de pixels  $p(i, j)$   
 $1 \leq i \leq H$  et  $1 \leq j \leq L$

pixel  $p(i, j) \equiv$  couleur



# Vocabulaire

- **Dimensions de l'image :**
  - Hauteur H (nombre de lignes)
  - Largeur L (nombre de colonnes)
- **Définition de l'image = L x H**
  - par ex : HD = 1280x720 (720p) , Full HD = 1920x1080 (1080p)
- **Résolution : rapport avec la taille réelle**
  - par ex : écrans : ppi (pixels per inch) = ppp (pixels par pouce)
  - 300 ppi = 118 pixels / cm (Image satellite : pixels / km)



# Image numérique 2

- Discrétisation régulière de l'intensité lumineuse

$$I \in [0, 1]$$



- ▶ Standard : 8 bits  $\Rightarrow 2^8 = 256$  niveaux de gris
- ▶ Binaire : 2 valeurs 0 et 1 (noir et blanc)
- ▶ Qualité pro, besoins spécifiques : 12, 14, 16 bits
- ▶ Biomédical, astrophysique : flottants



MAP201 : en TP, avec Python  
L'intensité est codée en **flottants**  
mais on affiche l'image  $\rightarrow$  256 niveaux de gris standard

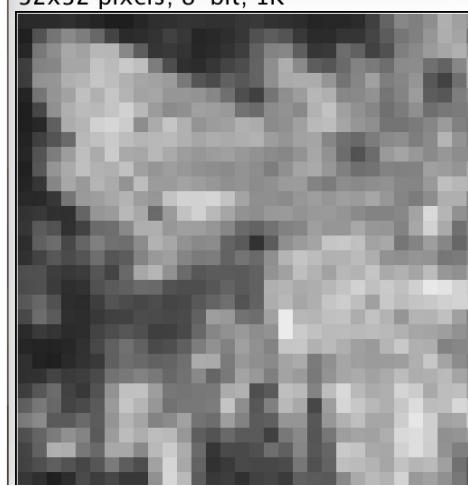
# Effets du changement de définition/résolution

256x256  
pixels  
= 65536 pixels



128x128  
pixels  
= 16384 pixels

64x64 pixels  
= 4096 pixels



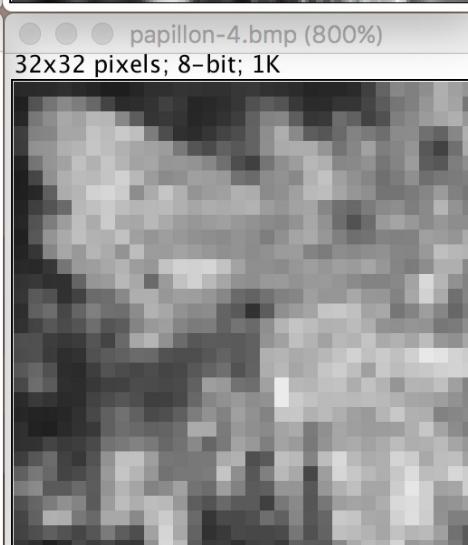
32x32 pixels  
= 1024 pixels

# Effets du changement de définition/résolution

256x256  
pixels  
= 65536 pixels

La valeur des pixels  
est stockée sur 8 bits  
(1 octet)

L'image est donc  
stockée sur 64 kPixels  
=  $64 \times (1024 \text{ pixels})$   
Et prend  
64 kB (64 koctets)  
=  $64 \times (1024 \text{ octets})$



32x32 pixels  
= 1024 pixels

La valeur des pixels  
est stockée sur 8 bits  
(1 octet)

L'image est donc  
stockée sur 1 kPixels  
=  $1 \times (1024 \text{ pixels})$   
Et prend  
1 kB (1 koctets)  
=  $1 \times (1024 \text{ octets})$

# Effet de l'échantillonnage



256



32



16



8



4

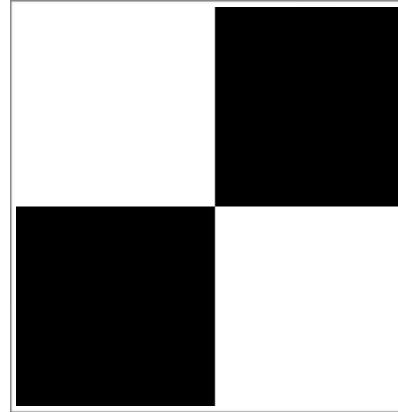


2

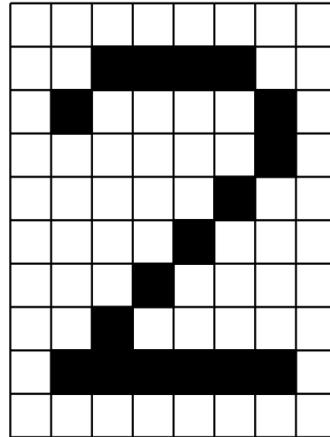
# Images binaires

- $im(i,j) = 0 \rightarrow \text{noir}$        $im(i,j) = 1 \rightarrow \text{blanc}$

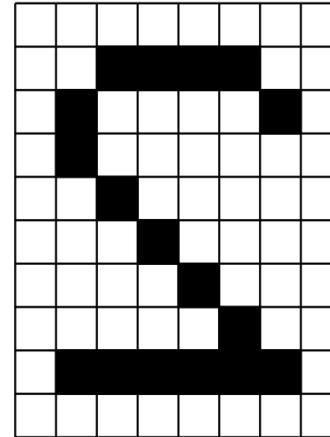
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1



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



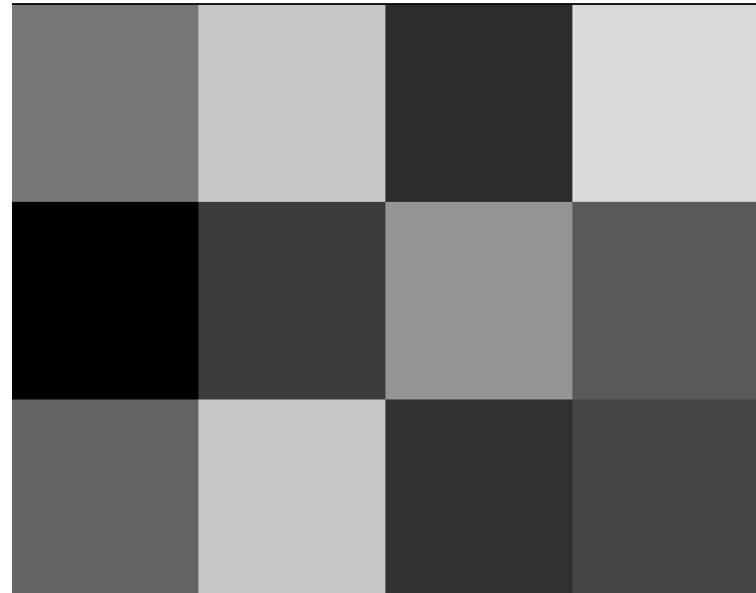
1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1	
1	0	1	1	1	1	0	1	
1	1	0	1	1	1	1	1	
1	1	1	0	1	1	1	1	
1	1	1	1	0	1	1	1	
1	1	1	1	1	0	1	1	
1	0	0	0	0	0	0	1	
1	1	1	1	1	1	1	1	



# Images en niveaux de gris 8 bits

- 8 bits =  $2^8 = 256$  valeurs de 0 à 255
  - ▶ 1 pixel = 8 bits = 1 octet
  - ▶  $im(i,j) = 0 \rightarrow$  noir     $im(i,j) = 255 \rightarrow$  blanc

```
7     im = np.array([[120,200,45,220],  
8                     [0,60,150,90],  
9                     [100,200,50,70]])
```



# Exemple

- Dégradé

0

255

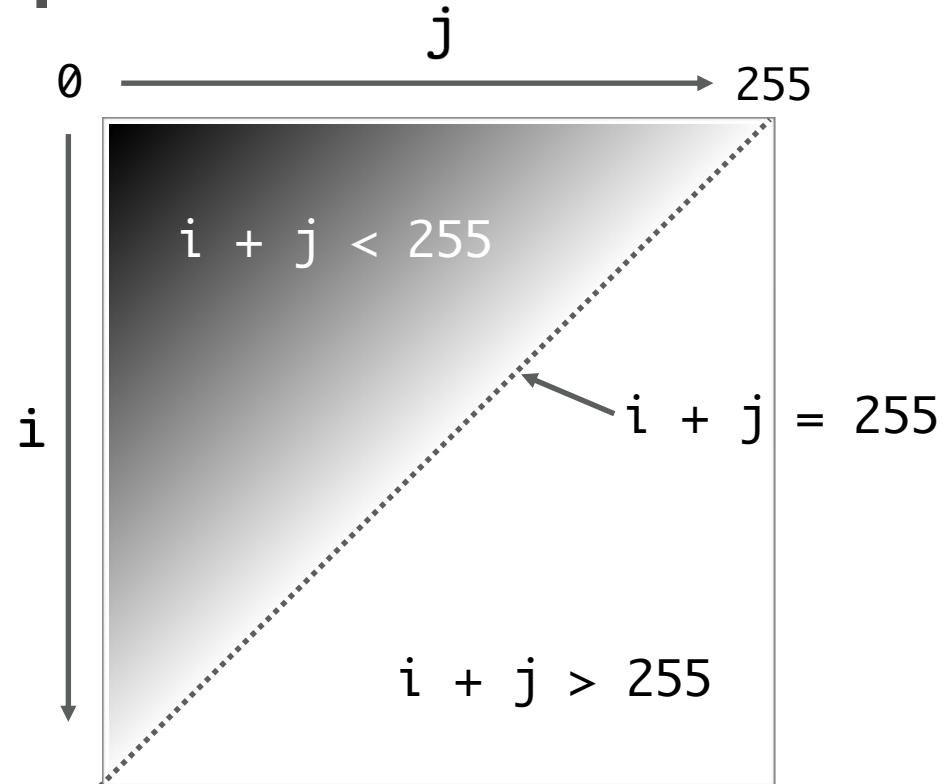


```
1  image = np.zeros([60,256])
2  for i in range(60):
3      for j in range(256):
4          image[i,j] = j-1
```

# Exemple

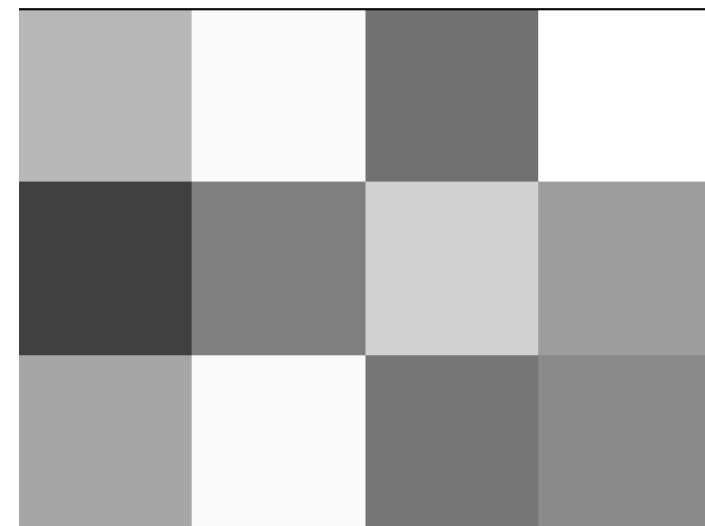
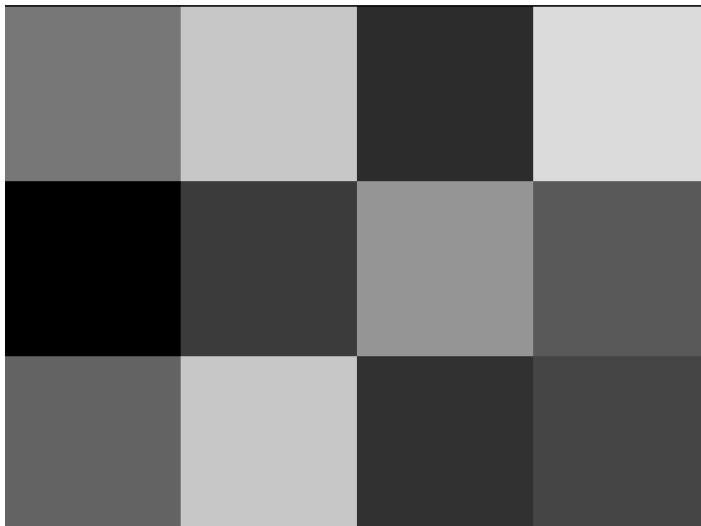
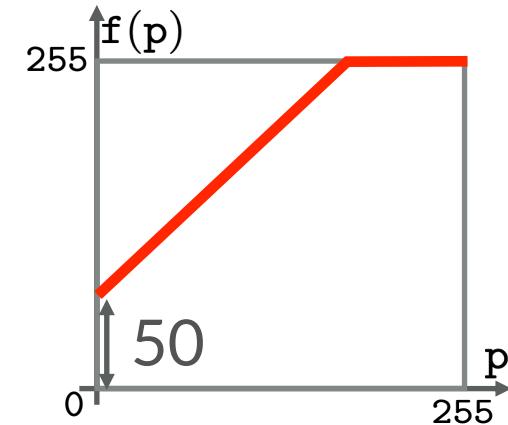
- Triangle dégradé

```
image = np.zeros([256, 256])
for i in range(256):
    for j in range(256):
        image[i, j] = min(i+j, 255)
```



# Transformation pixel par pixel

```
im = np.array([[120,200,45,220],  
...              [0,60,150,90],  
...              [100,200,50,70]])  
  
im2 = im + 50  
= np.array([[170,250,95,270],  
...              [50,110,200,140],  
...              [150,250,100,120]])
```

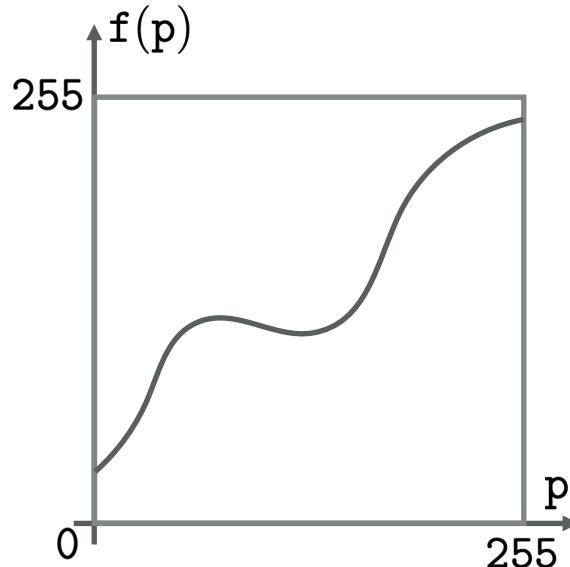


# Transformation pixel par pixel

- Fonctions sur les valeurs des pixels
  - Soit  $\mathbb{P} = \{0, \dots, 255\}$  l'ensemble des valeurs admissibles

$$f : \begin{cases} \mathbb{P} & \rightarrow \mathbb{P} \\ p & \mapsto f(p) \end{cases}$$

```
for i in range(256):
    for j in range(256):
        image[i,j] = f(image[i,j])
```



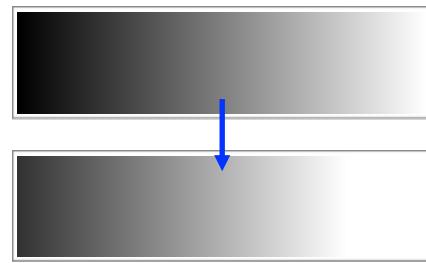
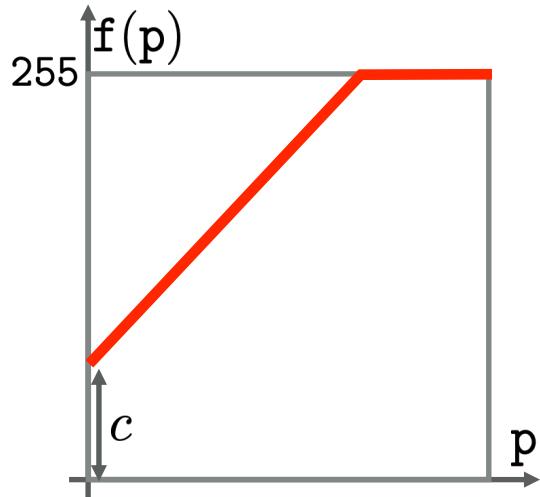
Contraintes :

- la courbe doit rester dans  $[0,255] \times [0,255]$
- les valeurs non-entières sont arrondies

# Exemple

- Augmenter la luminosité

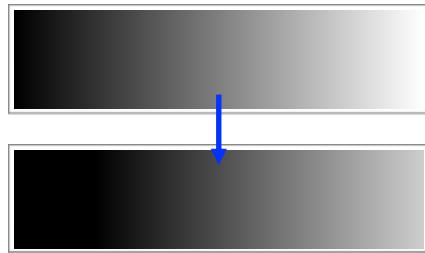
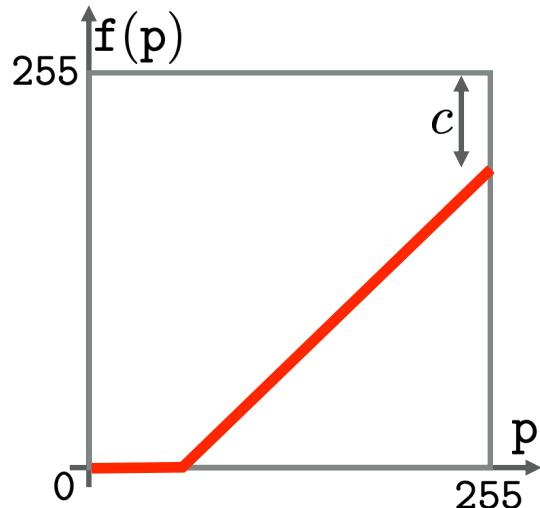
$$f(p) = \min(p + c, 255)$$



# Exemple

- Diminuer la luminosité

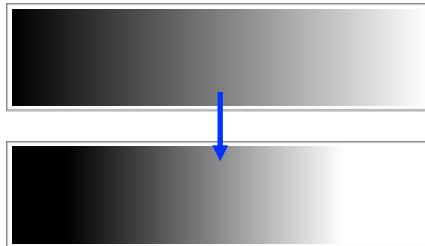
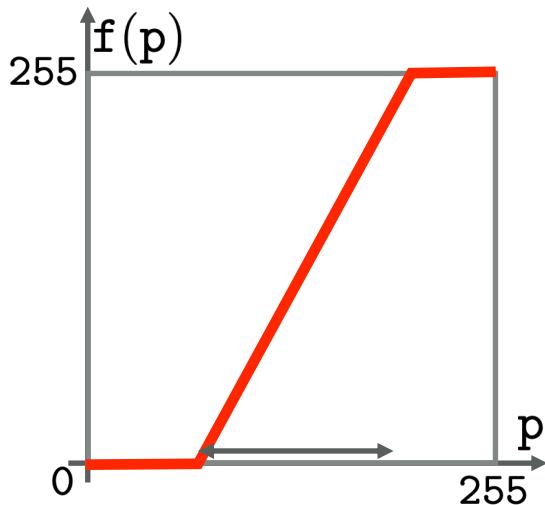
$$f(p) = \max(p - c, 0)$$



# Exemple

- Augmenter le contraste

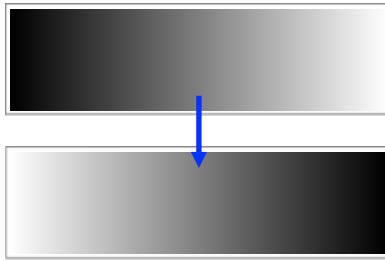
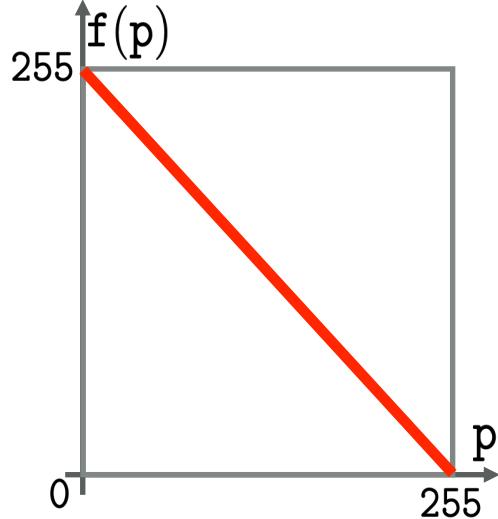
$$f(p) = \min(\max(a \times p - c, 0), 255)$$



# Exemple

- Négatif d'une image

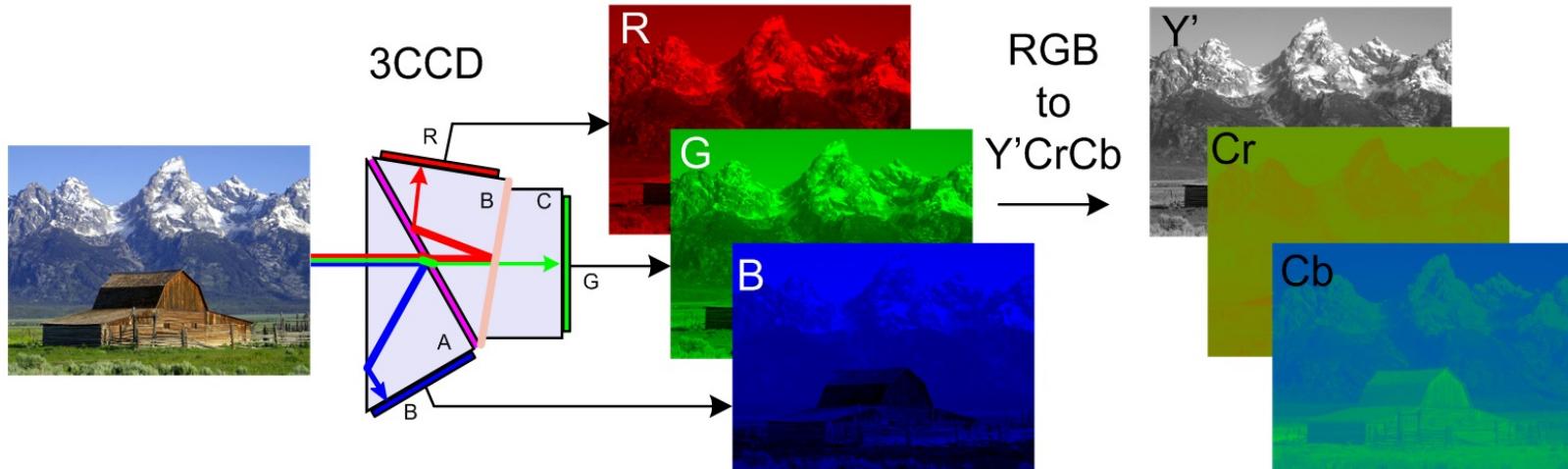
$$f(p) = 255 - p$$



# Images couleur

- RGB (red, green, blue) 8 bits/canal → 24 bits “true colors”  
utilisé par .bmp, .gif, .png
- “palettes” de couleurs 8 bits en tout → “look-up table”  
utilisé par .gif, .png
- Y'CbCr (Y': gris + 2 “couleurs”) 8 bits/canal → meilleur visuel si dégradation  
utilisé par .jpg

$$Y' = 0,299R + 0,587G + 0,114B \quad | \quad Cb = -0,1687R - 0,3313G + 0,5B + 128 \quad | \quad Cr = -0,1687R - 0,3313G + 0,5B + 128$$



# Exercices récapitulatifs

# Ex. 1: Retouche d'images



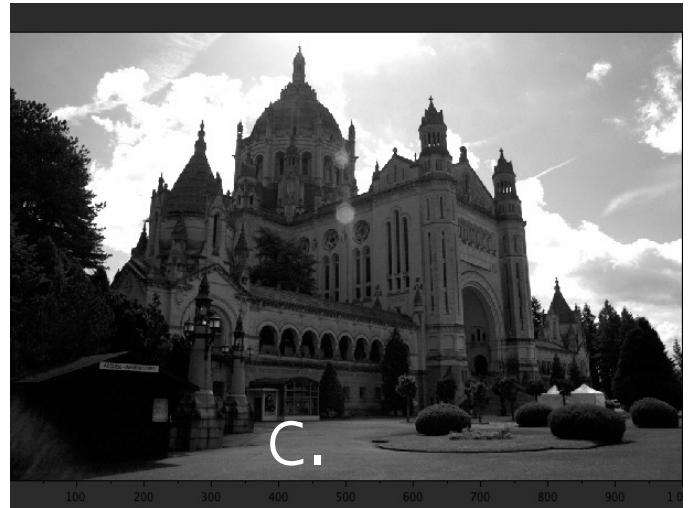
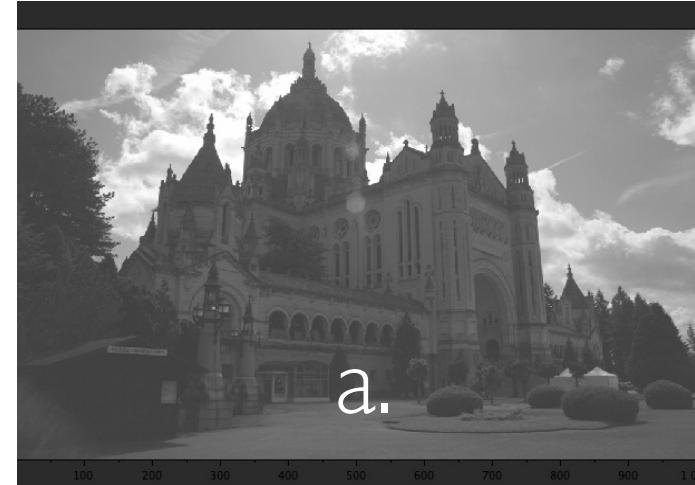
Image sous-exposée  
et sombre.  
Quelle transformation appliquer ?

# Ex. 1: Retouche d'images

Quelle image correspond à :

- $2 \times \text{im}$
- $\text{im} + 50$
- $\text{im} + 100$  ?

Pourquoi voit-on beaucoup de “blanc”?

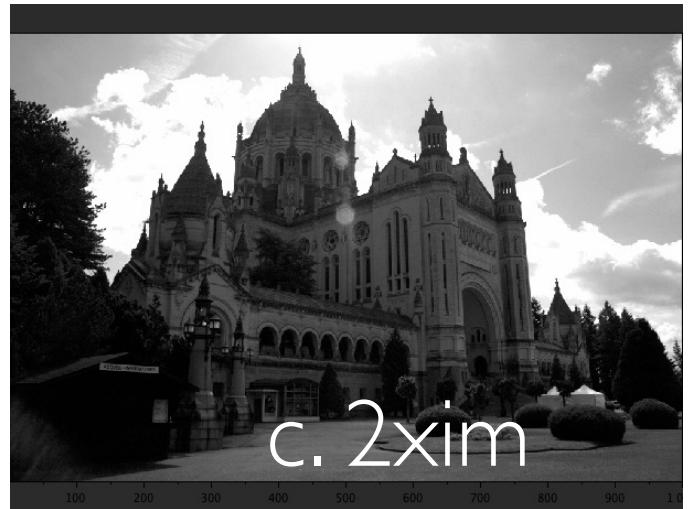
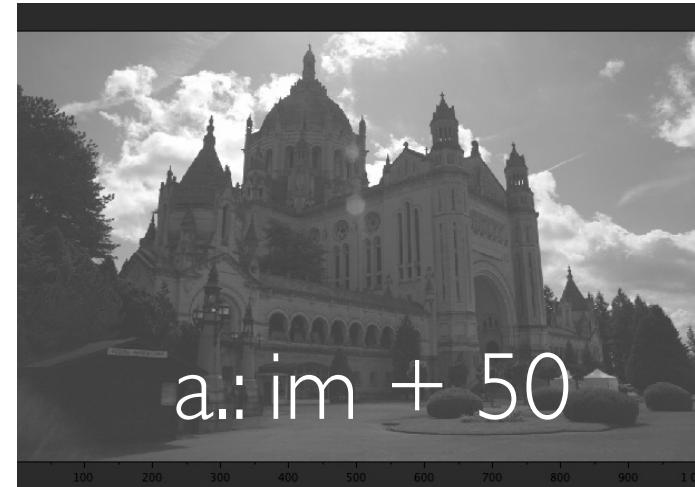


# Ex. 1: Retouche d'images

Quelle image correspond à :

- $2xim$
- $im + 50$
- $im + 100$  ?

Pourquoi voit-on beaucoup de “blanc”?



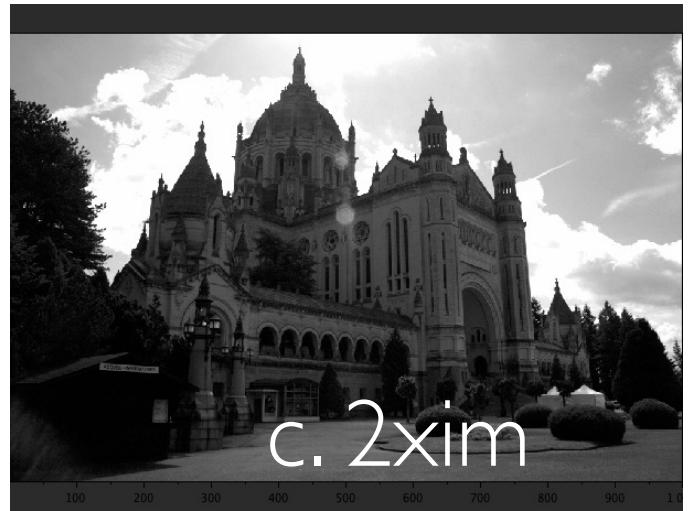
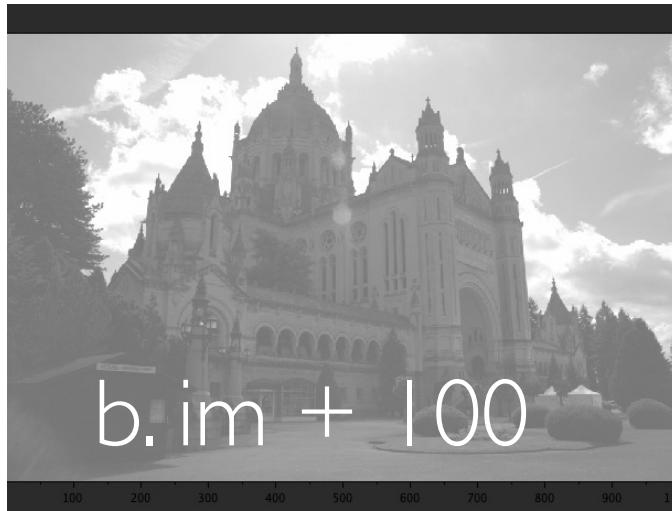
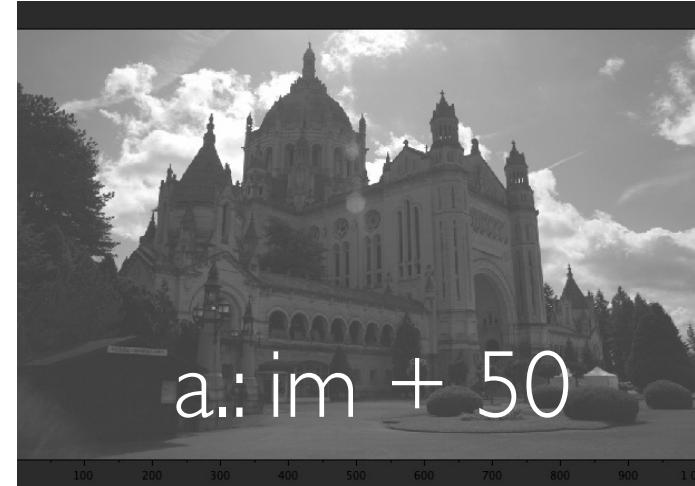
# Ex. 1: Retouche d'images

Quelle image correspond à :

- $2xim$
- $im + 50$
- $im + 100$  ?

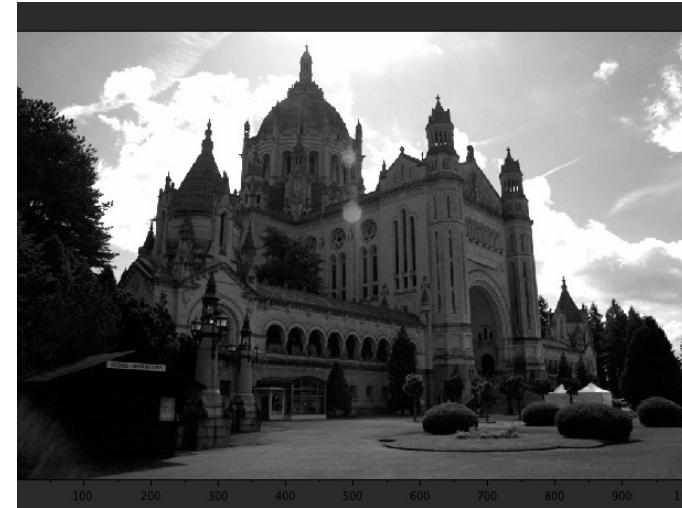
Pourquoi voit-on beaucoup de “blanc”?

→ seuillage à 255



## Ex. 2: Discrétisation d'image

- En passant de 720p (1280x720 pixels) à 1080p (1920x1080 pixels) par combien est multiplié:
  - la résolution verticale
  - la résolution horizontale
  - le nombre de pixels
- Si je veux que mon image 720p en 256 niveaux de gris ait la même taille mémoire qu'une image 1080p, combien de niveaux de gris choisir?



## Ex. 2: Discrétisation d'image

- En passant de 720p (1280x720 pixels) à 1080p (1920x1080 pixels) par combien est multiplié:
  - la résolution verticale  $1080 \div 720 = 1,5$
  - la résolution horizontale  $1920 \div 1280 = 1,5$
  - le nombre de pixels  $1920 \times 1080 \div (1280 \times 720) = 2,25$
- Si je veux que mon image 720p en 256 niveaux de gris ait la même taille mémoire qu'une image 1080p, combien de niveaux de gris choisir?  
 $8\text{Bits}/2,25 = 3,25 \rightarrow 3\text{bits/pixels} \rightarrow 2^3 = 8 \text{ couleurs !}$

