

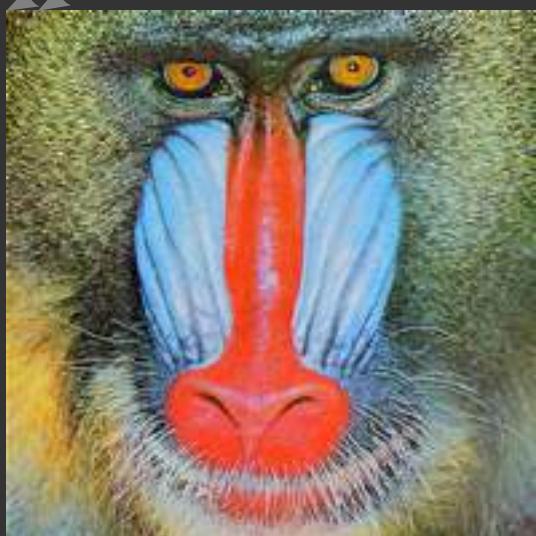
# Codage de l'image

# Codage de l'image

## Représentation d'une image couleur

- Codage d'une image par une matrice :

- L'image est une fonction discrète 2D, elle est souvent codée par une matrice.
- Pour une image codée en RGB, un point de l'image = un triplet (r,g,b) de valeurs dans la matrice
- Un point de l'image = un pixel. Que signifie pixel ?

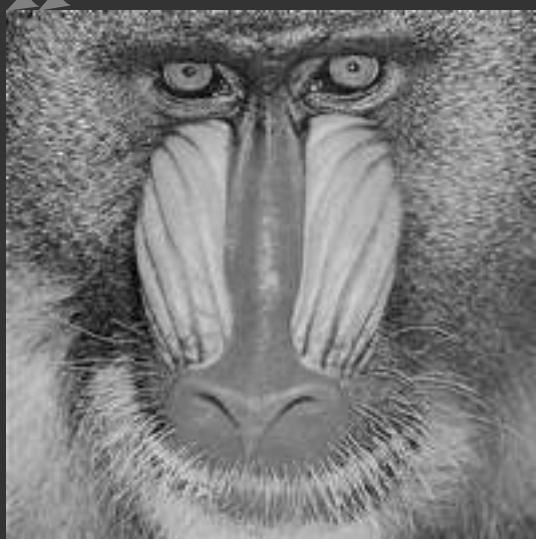


195, 167, 2	193, 87, 4	...
189, 154, 4	...	...
...	...	...

# Codage de l'image

## Représentation d'une image en niveaux de gris

- Codage d'une image par une matrice :
  - L'image est une fonction discrète 2D, elle est souvent codée par une matrice.
  - Pour une image codée en niveaux de gris, un point de l'image = une valeur dans la matrice codant la luminance



195	193	...
189	...	...
...	...	...

# Codage de l'image

## Accès aux pixels

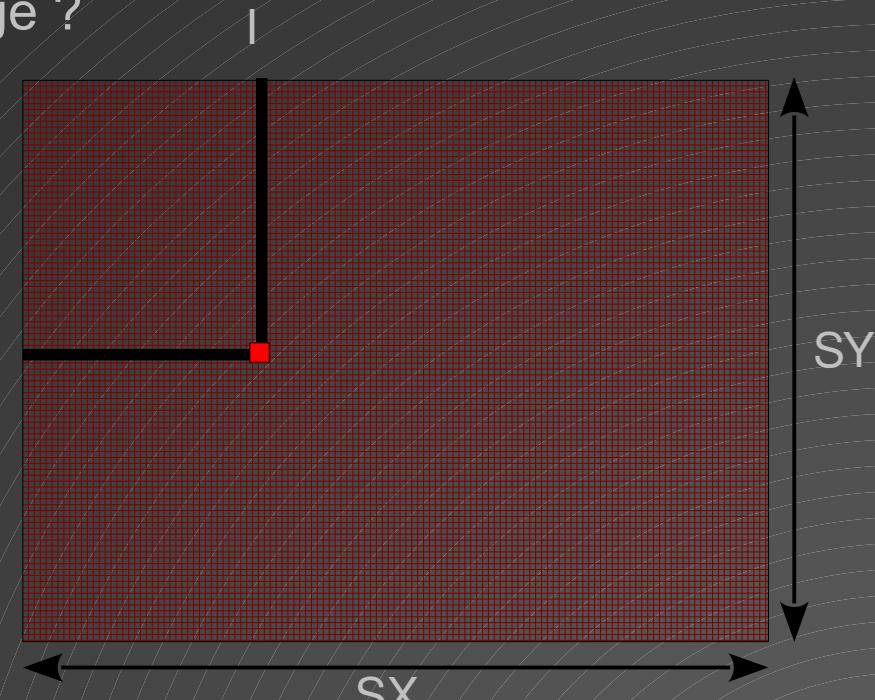
- Comment coder cette image en mémoire ?
  - Matrice ? Vecteur ?
- Comment accéder à un point de cette image ?
- Comment accéder à ses voisins ?
- Comment parcourir l'image ?

```
for(i=0;i<sx;i++)  
    for(j=0;j<sy;j++)
```

```
for(j=0;j<sy;j++)  
    for(i=0;i<sx;i++)
```

```
for(offset=0;offset<sx*sy;++offset)
```

- Utiliser un itérateur ?



# Codage de l'image

## Accès aux pixels

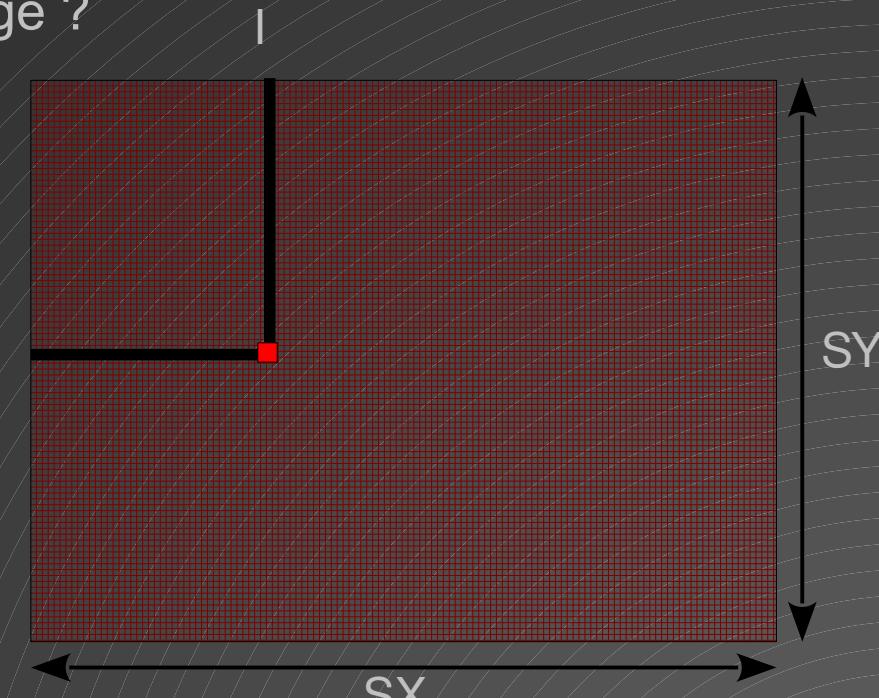
- Comment coder cette image en mémoire ?
  - Matrice ? Vecteur ?
- Comment accéder à un point de cette image ?
- Comment accéder à ses voisins ?
- Comment parcourir l'image ?

```
for(i=0;i<sx;i++)  
    for(j=0;j<sy;j++)  
        offset=i+j*sx;
```

```
for(j=0;j<sy;j++)  
    for(i=0;i<sx;i++)  
        offset=i+j*sx;
```

```
for(offset=0;offset<sx*sy;++offset)
```

- Utiliser un itérateur ?



# Codage de l'image

## Résolution/Échantillonnage

- Discrétisation spatiale (résolution)



- Échantillonnage (amplitude)



# Codage de l'image

## Nombre de couleurs - Échantillonnage

- Codage par palette (couleurs indexées)

- Bit(s) par pixel                      Couleurs

1 bpp	?
2 bpp	?
4 bpp	?
6 bpp	?
8 bpp	?

- Codage sans palette

- Bits par pixel                      Couleurs              Bits par canaux

16 bpp	?	?
24 bpp	?	?
32 bpp	?	?

# Codage de l'image

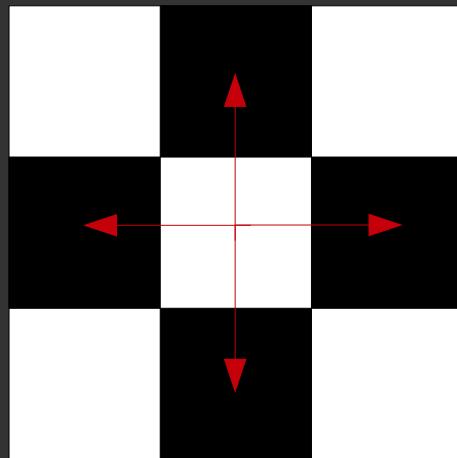
## Représentation de l'image

- Un moyen classique de représenter une image est d'utiliser une matrice. Y a t il d'autres approches ?
  - Arbres (max tree, min tree, tree of shape...)
  - Graphes
  - ...
- Maillage
  - On choisit intuitivement un maillage carré mais cela peut-il présenter des inconvénients ?
  - Y a t il d'autres maillages possibles ?

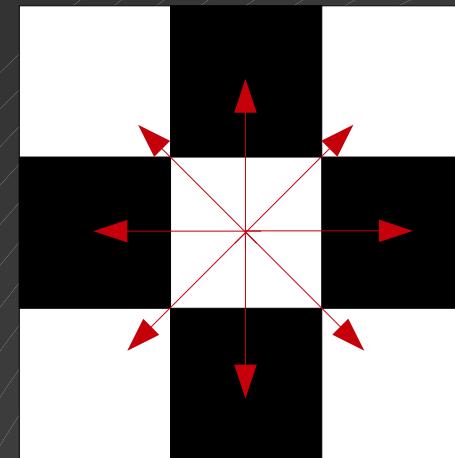
# Codage de l'image

## Topologie

- Choix de la connexité des pixels



4 - connexe

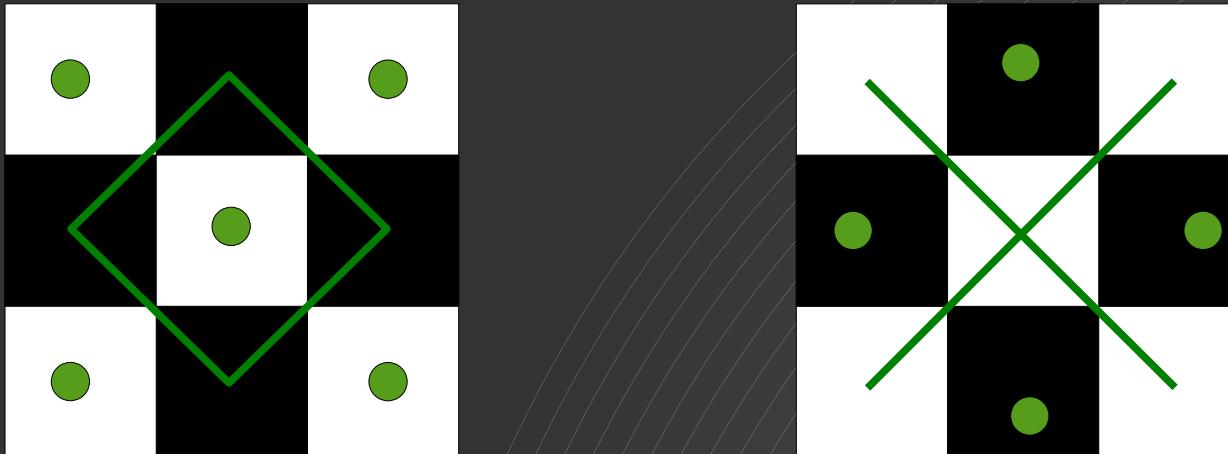


8 - connexe

# Codage de l'image

## Topologie

- Choix de la connexité des pixels
  - Cela pose un problème de topologie :

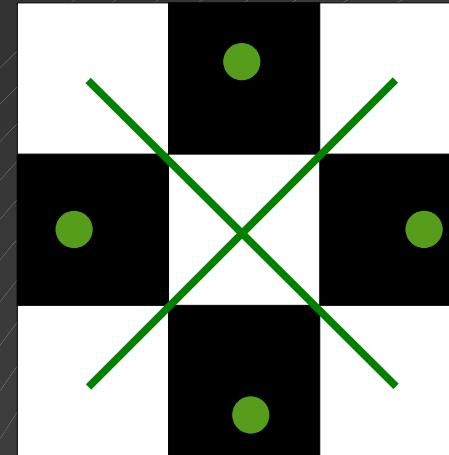
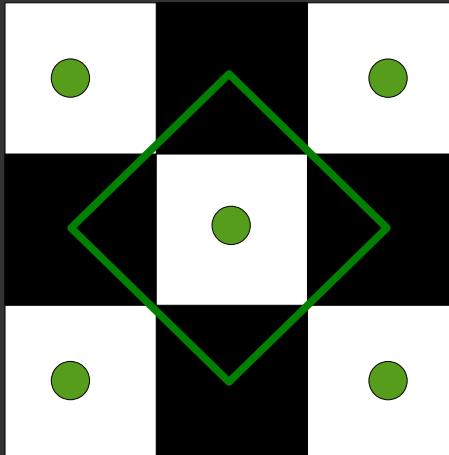


- Si le fond est 8-connexe (en noir), la forme (en blanc) est 4-connexe
- Si le fond est 4-connexe (en noir), la forme (en blanc) est 8-connexe
- Contradiction avec le théorème de Jordan

# Codage de l'image

## Topologie

- Choix de la connexité des pixels
  - Cela pose un problème de topologie :

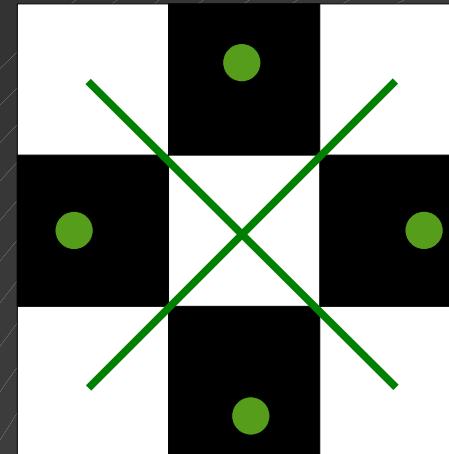
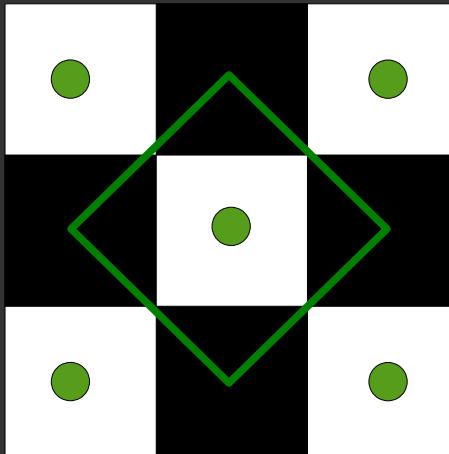


- Que faire ?

# Codage de l'image

## Topologie

- Choix de la connexité des pixels
  - Cela pose un problème de topologie :

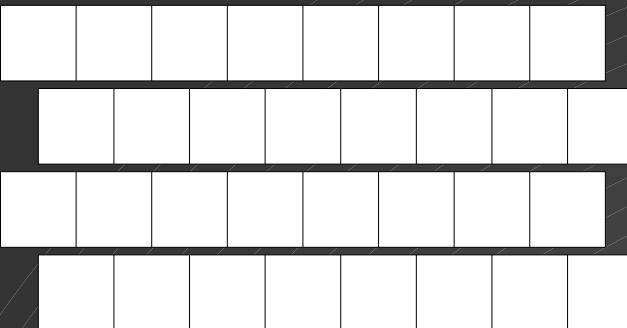
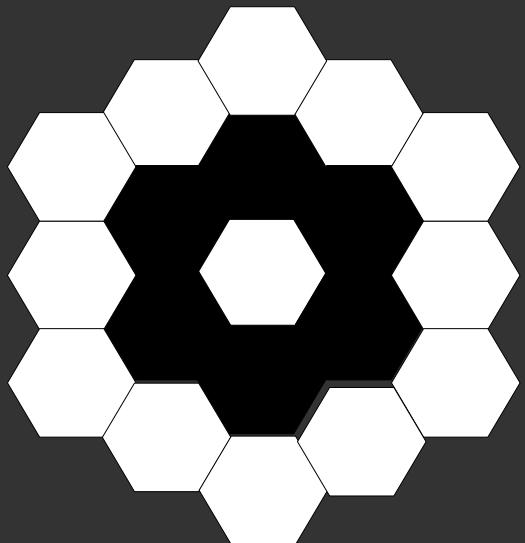


- Que faire ?
  - Vivre avec
  - Changer la forme de pixels
  - Intercaler des frontières entre les pixels
  - ...

# Codage de l'image

## Topologie

- Changer la forme de pixels :



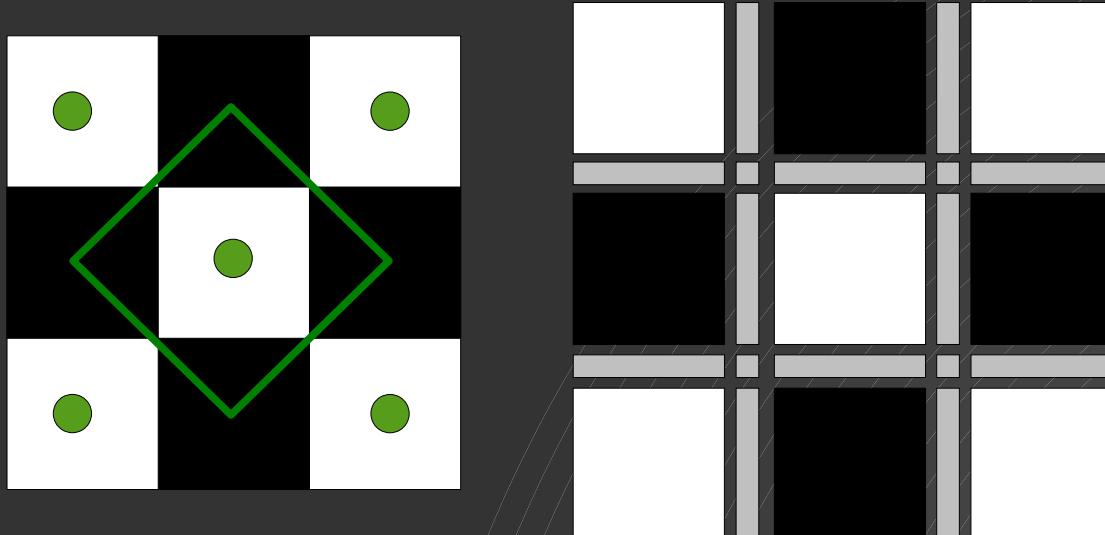
Codage en mémoire

- Pour :
  - Plus de problème de connexité
  - Plus de problème de distance
- Contre :
  - Gestion de la mémoire

# Codage de l'image

## Topologie

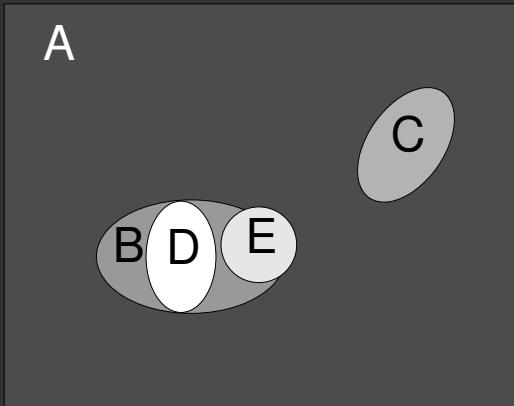
- Intercaler des frontières entre les pixels



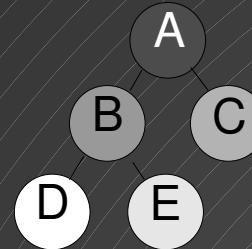
# Codage de l'image

## Représentation de l'image

- Exemple d'arbre : Max tree



Image

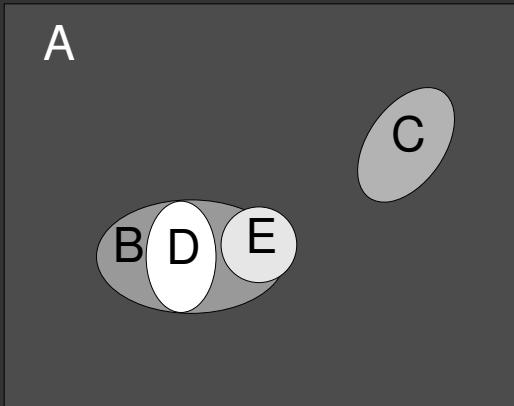


Max-Tree correspondant

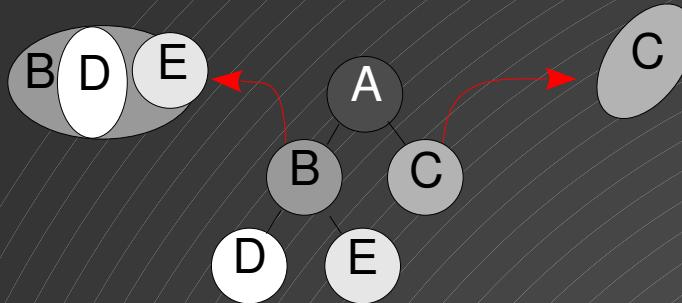
# Codage de l'image

## Représentation de l'image

- Exemple d'arbre : Max tree



Image

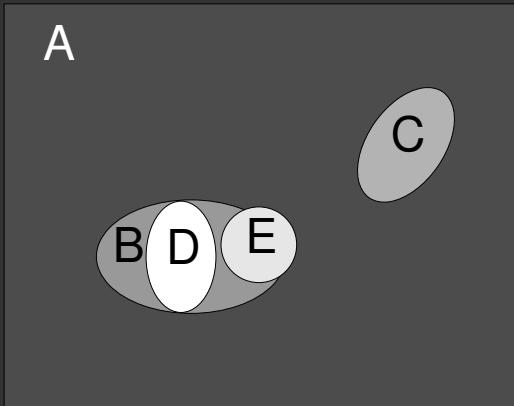


Max-Tree correspondant

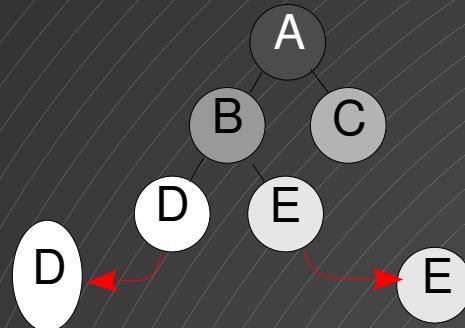
# Codage de l'image

## Représentation de l'image

- Exemple d'arbre : Max tree



Image



Max-Tree correspondant

# Codage de l'image

## Stockage/Transfert

- Différents formats :
  - JPEG, TIFF, PNM, PNG, BMP, GIF, TGA...
- Choix en fonction de critères
  - Avec ou sans compression (avec ou sans perte)
  - Avec ou sans couleur
  - Avec ou sans palette
  - Une seule image ou plusieurs
  - Optimisé pour une architecture ? (Ex. BMP sauvegardé à l'envers)
  - Libre ou pas (Ex. GIF et Compuserve)

# Codage de l'image

## Stockage/Transfert

- Format PNM :
  - PBM : noir et blanc
  - PGM : niveaux de gris
  - PPM : couleurs
- Deux variantes
- Format très simple (extrait de spec):

Each PPM image consists of the following:

A "magic number" for identifying the file type. A ppm image's magic number is the two characters "P6".

Whitespace (blanks, TABs, CRs, LFs).

A width, formatted as ASCII characters in decimal.

Whitespace.

A height, again in ASCII decimal.

Whitespace.

The maximum color value (Maxval), again in ASCII decimal. Must be less than 65536 and more than zero.

A single whitespace character (usually a newline).

A raster of Height rows, in order from top to bottom. Each row consists of Width pixels, in order from left to right. Each pixel is a triplet of red, green, and blue samples, in that order. Each sample is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.

# Applications

- On a vu pas mal de choses sur la formation d'une image
- On va l'appliquer
  - en changeant les couleurs ou l'illumination d'une image
  - en changeant l'organisation spatiale des pixels de l'image
  - en combinant des changements dans les couleurs et dans l'organisation spatiale des pixels

# Applications

## Changement d'illumination

- En tous points de la scène, la réponse du capteur est donné par :
- $L(x, y) = \int E(x, y, \lambda) S(x, y, \lambda) R(\lambda) d\lambda$ 
  - Avec
    - $E(x, y)$  l'éclairage.
    - $S(x, y, \lambda)$  la réflectance de la surface (fonction de la longueur d'onde  $\lambda$ )
    - $R(\lambda)$  la sensibilité du capteur qui (pour simplifier est supposé répondre à une seule longueur d'onde :  $R(\lambda) = \delta(\lambda - \lambda_k)$ )
  - On a donc :  $L(x, y) = E(x, y) S(x, y, \lambda_k)$

# Applications

## Changement d'illumination

- La même image prise avec deux niveaux d'illuminations différents :
  - $L_1(x, y) = E_1(x, y) S(x, y, \lambda_k)$
  - $L_2(x, y) = E_2(x, y) S(x, y, \lambda_k)$
- Donc :
  - $L_2(x, y) = (E_2(x, y) / E_1(x, y))^* L_1(x, y)$
- Et donc :
  - $L_2(x, y) = C * L_1(x, y)$
- Pour changer l'illumination il faut donc multiplier les valeurs des pixels par une constante (et non additionner/soustraire par une constante comme c'est usuellement fait).

# Applications

## Correction d'illumination non uniforme

- Soit une image acquise  $I_1$  avec un éclairage non uniforme.
- $I_1(x, y) = S(x, y, \lambda_k) E(x, y)$



Image issue du cours de C. Achard

# Applications

## Correction d'illumination non uniforme

- Soit l'image du fond  $I_f$
- $I_f(x, y) = F(x, y, \lambda_k) E(x, y)$



Image issue du cours de C. Achard

# Applications

## Correction d'illumination non uniforme

- La soustraction des deux donne :

$$I_1(x, y) - I_f(x, y) = [S(x, y, \lambda_k) - F(x, y, \lambda_k)] E(x, y)$$

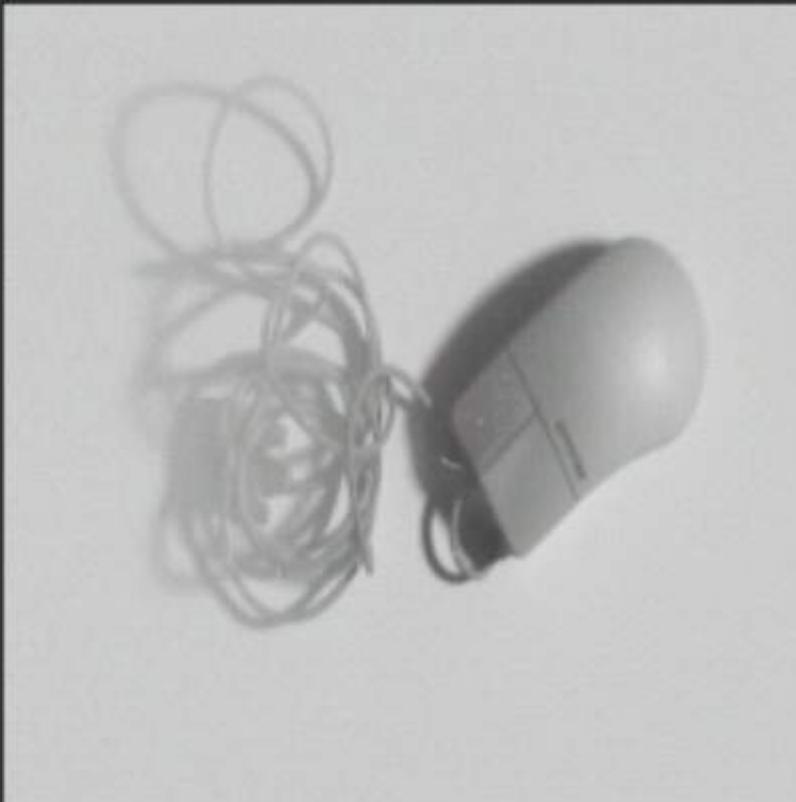


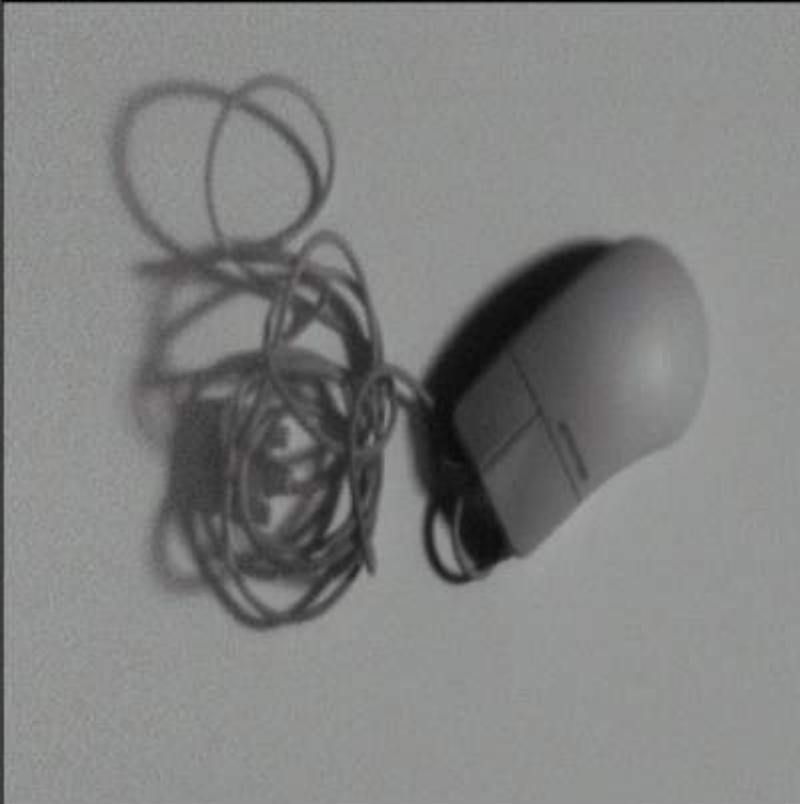
Image issue du cours de C. Achard

# Applications

## Correction d'illumination non uniforme

- Le ratio des deux donne :

$$I_1(x, y) / I_f(x, y) = S(x, y, \lambda_k) / F(x, y, \lambda_k)$$



→ indépendant de E

Image issue du cours de C. Achard

# Applications

## Correction d'illumination non uniforme

Différence vs Ratio

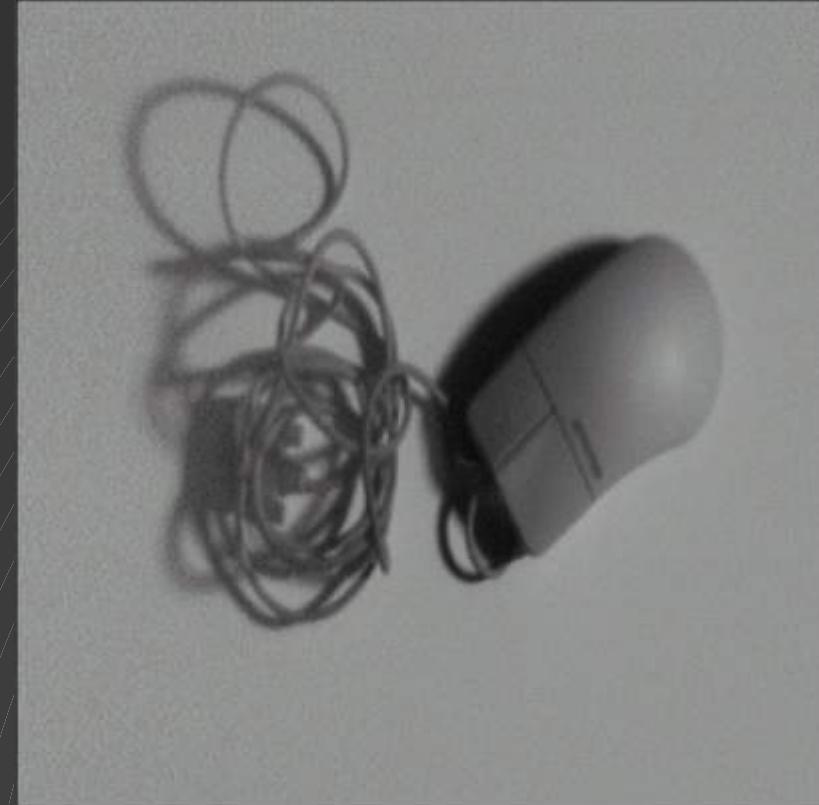
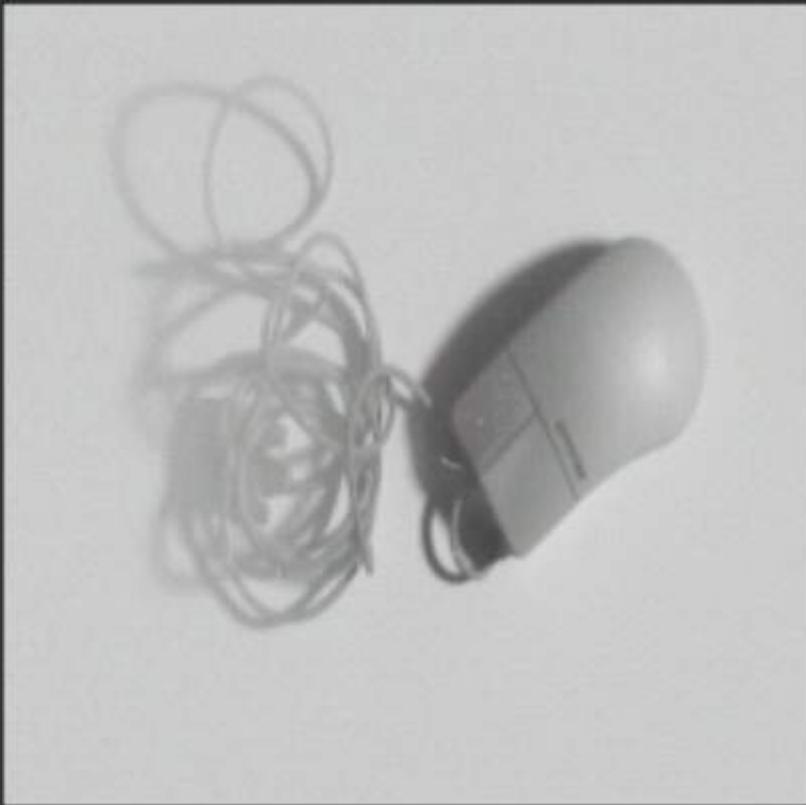


Image issue du cours de C. Achard

# Applications

## Modification des couleurs de l'image

- Application : effet artistique → effet sépia  
(passage du niveau de gris à la couleur)

# Applications

## Modification des couleurs de l'image

- Application : effet artistique → effet sépia
  - On associe à un niveau de luminance, une couleur



$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} 0,784 & 0 & 0 \\ 0 & 0,588 & 0 \\ 0 & 0 & 0,391 \end{pmatrix} \begin{pmatrix} l \\ l \\ l \end{pmatrix}$$

# Applications

## Modification des couleurs de l'image

- Application : effet artistique
  - effet sépia – résultat



# Applications

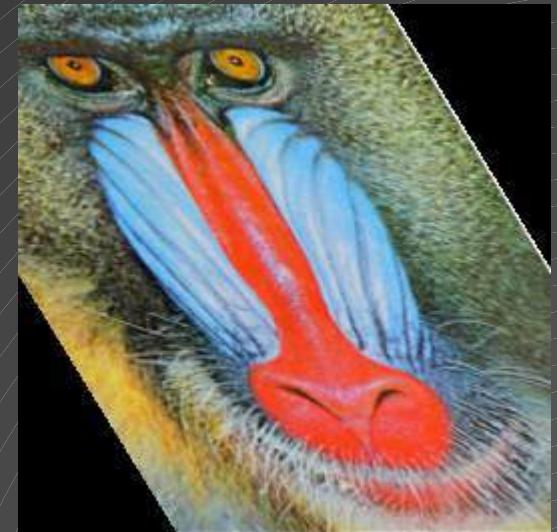
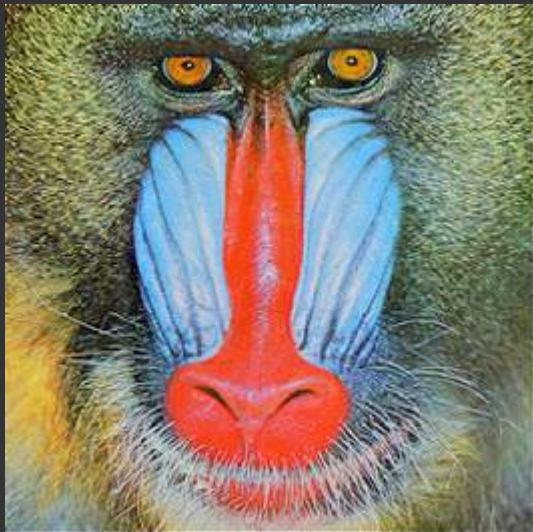
## Modification de l'organisation spatiale des pixels

- Application : effets artistiques
  - $\text{image\_resultat}(x,y) = \text{image\_origin}(g(x,y), h(x,y))$ 
    - Les fonctions g et h ne tiennent pas forcément compte de la valeur du pixel

# Applications

## Modification de l'organisation spatiale des pixels

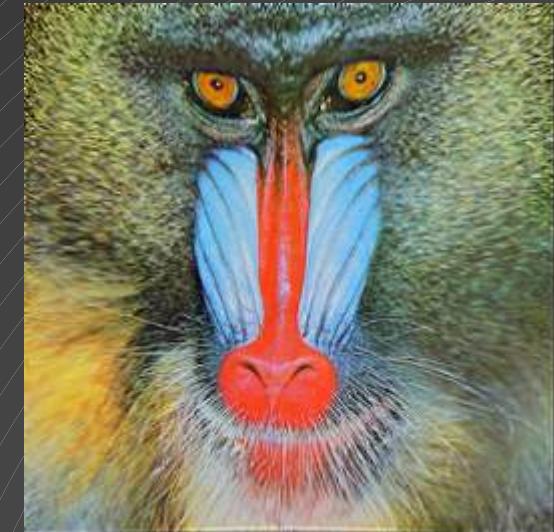
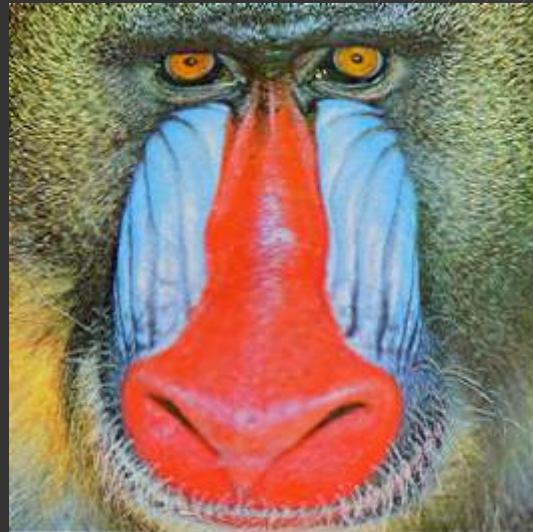
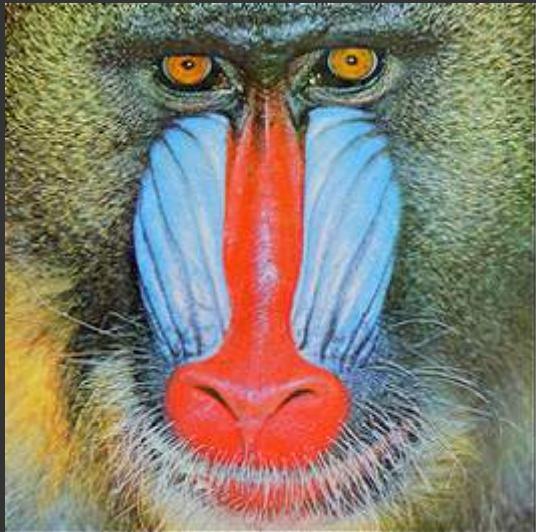
- Application : effets artistiques - exemples
  - Rotation – cisaillement...



# Applications

## Modification de l'organisation spatiale des pixels

- Application : effets artistiques – exemples
  - Étirement – rétrécissement...



# Applications

## Modification de l'organisation spatiale des pixels

- Application : effets artistiques – exemples
  - Ondulations
  - Spirale
  - Translations aléatoires
  - ...



# Applications

## Modification de l'organisation spatiale des pixels

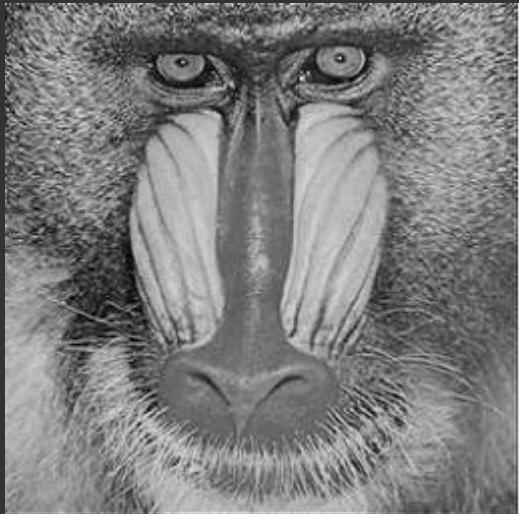
- Application : effets artistiques
  - $\text{image\_resultat}(x,y) = \text{image\_origin}(g(x,y), h(x,y))$ 
    - La transformation doit être appliquée dans ce sens !

# Application : Le morphing d'images

## Modification des couleurs et de l'organisation spatiale des pixels

- Application : Le morphing

- Vu la structure d'une image, il est possible d'appliquer des opérateurs sur ces images
- Exemple : la moyenne



# Application : Le morphing d'images

## Modification des couleurs et de l'organisation spatiale des pixels

- Application : Le morphing
  - En combinant
    - Une moyenne pondérée des images (dont les poids évoluent au cours du temps)
    - Un champ de vecteur translation

# Problèmes de précision

- Problèmes de précision :
  - Sur les fonctions colorimétriques
  - Sur les transformations spatiales

# Problème de précision

## La correction gamma

- Retour sur la perception
  - La perception de l'œil est logarithmique



- La répartition des niveaux d'énergie n'est donc pas linéaire mais exponentielle

# Problème de précision

## La correction gamma

- Retour sur la perception
  - La perception de l'œil est logarithmique

NIVEAU GRIS	Echelle linéaire	Signal
0	0.00	0.00
32	0.13	0.01
64	0.25	0.05
96	0.38	0.12
128	0.50	0.22
160	0.63	0.35
186	0.73	0.50
192	0.76	0.53
224	0.88	0.75
255	1.00	1.00

# Problème de précision

## La correction gamma

- Retour sur la perception

$r = (\text{number}/255)^{\text{gamma}}$

gamma = 2.2

NIVEAU GRIS	Echelle linéaire	Signal
0	0.00	0.00
32	0.13	0.01
64	0.25	0.05
96	0.38	0.12
128	0.50	0.22
160	0.63	0.35
186	0.73	0.50
192	0.76	0.53
224	0.88	0.75
255	1.00	1.00

- Les niveaux de gris ne sont que des numéros, faire des opérations (moyenne, addition, application de filtres, interpolation...) n'a pas vraiment de sens

# Problème de précision

## La correction gamma

- Dans la pratique, on omet souvent la correction gamma lors des étapes de filtrage
  - C'est faux
- Il y a un compromis entre précision du résultat et vitesse

# Problème de précision

## La correction gamma

- Retour sur le passage de la couleur en niveaux de gris
- Espace CIE XYZ 1931
  - $R' = r/255 \quad V' = v/255 \quad B' = b/255$
  - $Rsrvb = [(R'+0.055)/1.055]^{2.4}$
  - $Vsrvb = [(V'+0.055)/1.055]^{2.4}$
  - $Bsrvb = [(B'+0.055)/1.055]^{2.4}$
- $$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0,4124 & 0,3576 & 0,1805 \\ 0,2126 & 0,7152 & 0,0722 \\ 0,0193 & 0,1192 & 0,9505 \end{pmatrix} \begin{pmatrix} Rsrvb \\ Vsrvb \\ Bsrvb \end{pmatrix}$$
- Y donne la luminance.

# Problème de précision

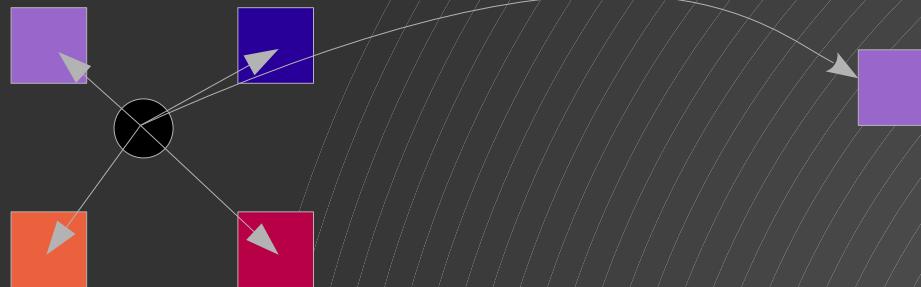
## L'interpolation

- Que faire lorsque l'on doit « chercher » la valeur d'un pixel mais que l'on ne tombe pas précisément sur un pixel ?

# Problème de précision

## L'interpolation

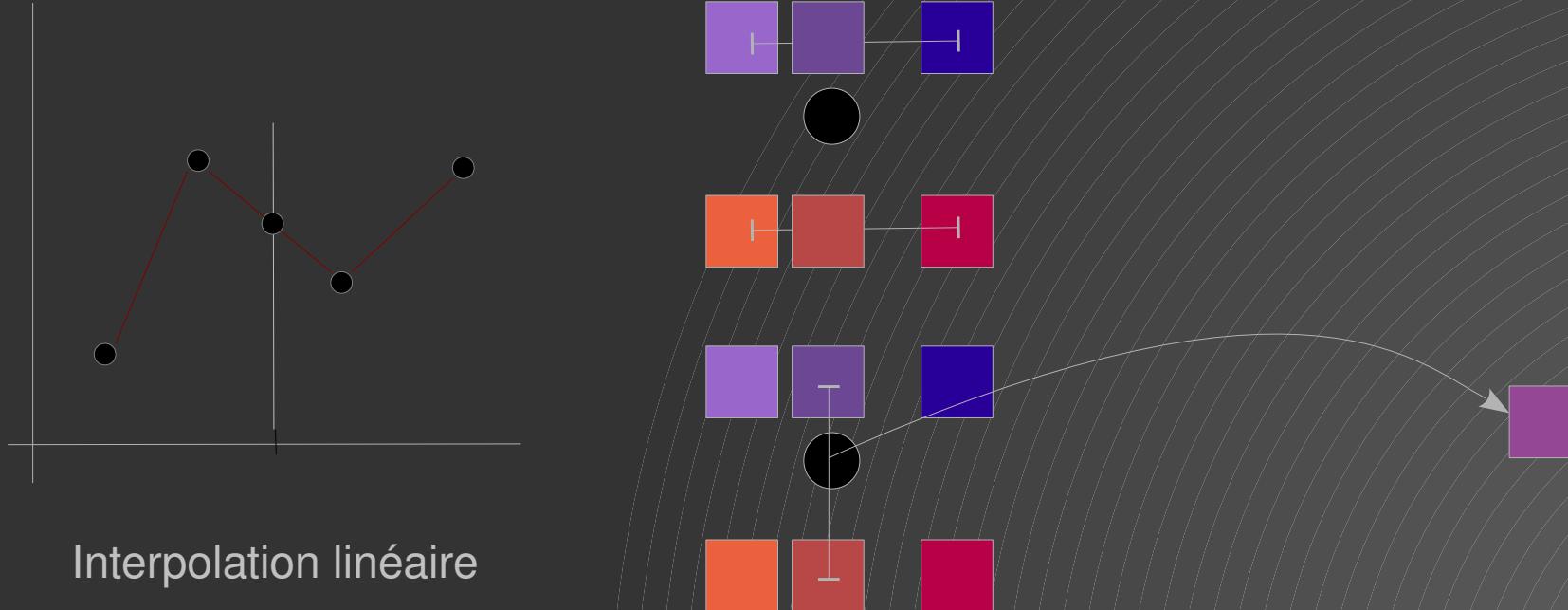
- Que faire lorsque l'on doit « chercher » la valeur d'un pixel mais que l'on ne tombe pas précisément sur un pixel ?
  - 1er solution (rapide) : prendre la valeur du pixel le plus proche



# Problème de précision

## L'interpolation

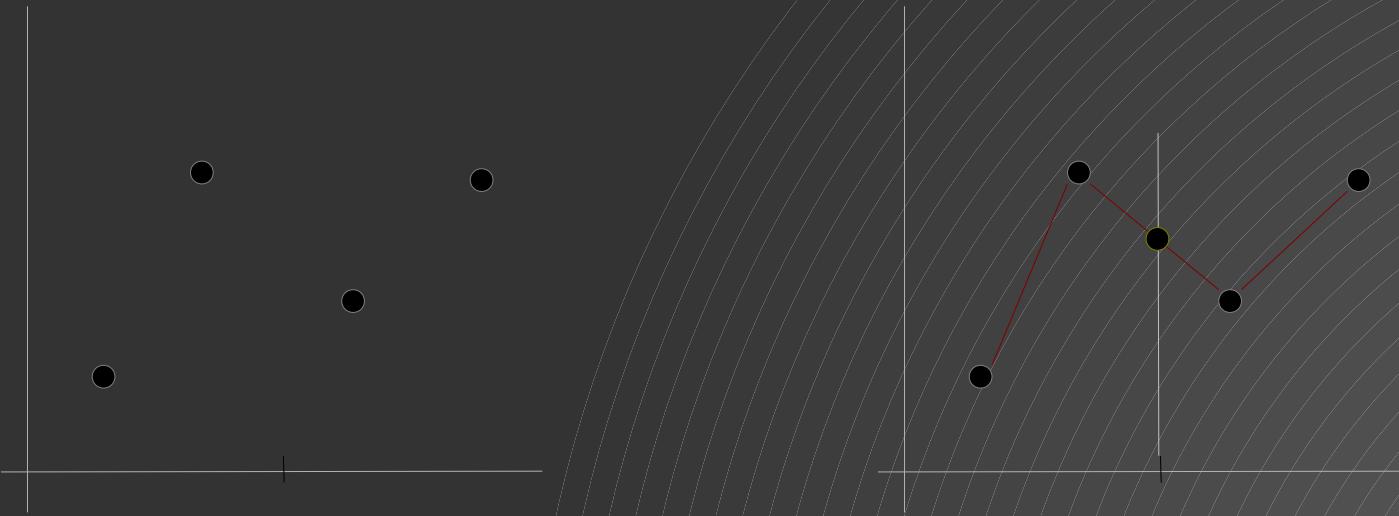
- Que faire lorsque l'on doit « chercher » la valeur d'un pixel mais que l'on ne tombe pas précisément sur un pixel ?
  - 2nd solution : Faire une interpolation bi-linéaire



# Problème de précision

## L'interpolation

- Peut-on faire mieux ?
  - Interpolation bic cubique
    - Tenir compte de la dérivée pour interpoler

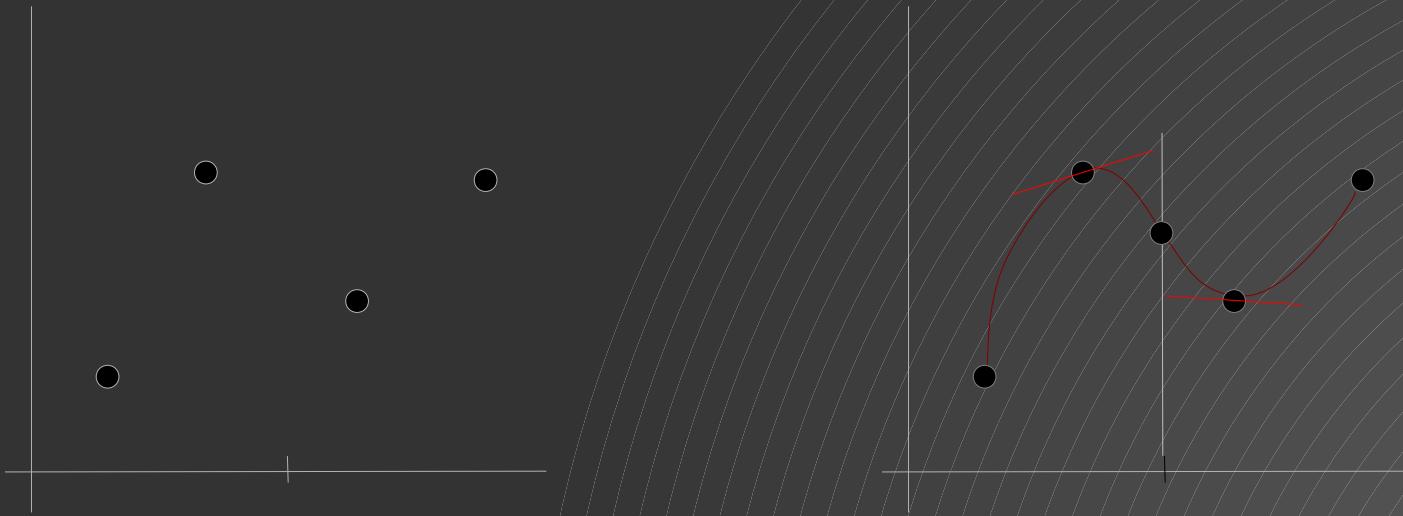


Interpolation linéaire

# Problème de précision

## L'interpolation

- Interpolation bicubique
  - Utilise 4 points (calcul de la dérivée)



# Problème de précision

## L'interpolation

- Interpolation bicubique

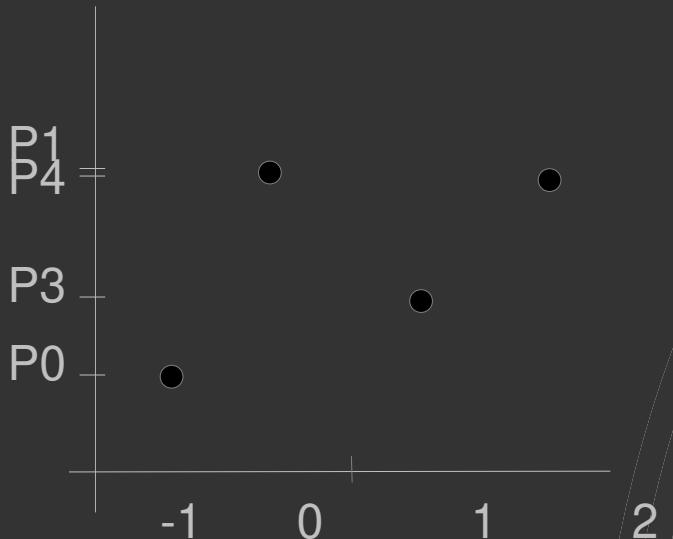
$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$

On connaît les valeurs pour  $x=-1$ ,  $x=0$ ,  $x=1$  et  $x=2$

$$f(-1) = p_0, f(0)=p_1, f(1)=p_2 \text{ et } f(2)=p_3$$

$$f'(0) = (p_2-p_0)/2 \text{ et } f'(1) = (p_3-p_1)/2$$



Mais aussi

$$f(0) = d$$

$$f(1) = a+b+c+d$$

$$f'(0) = c$$

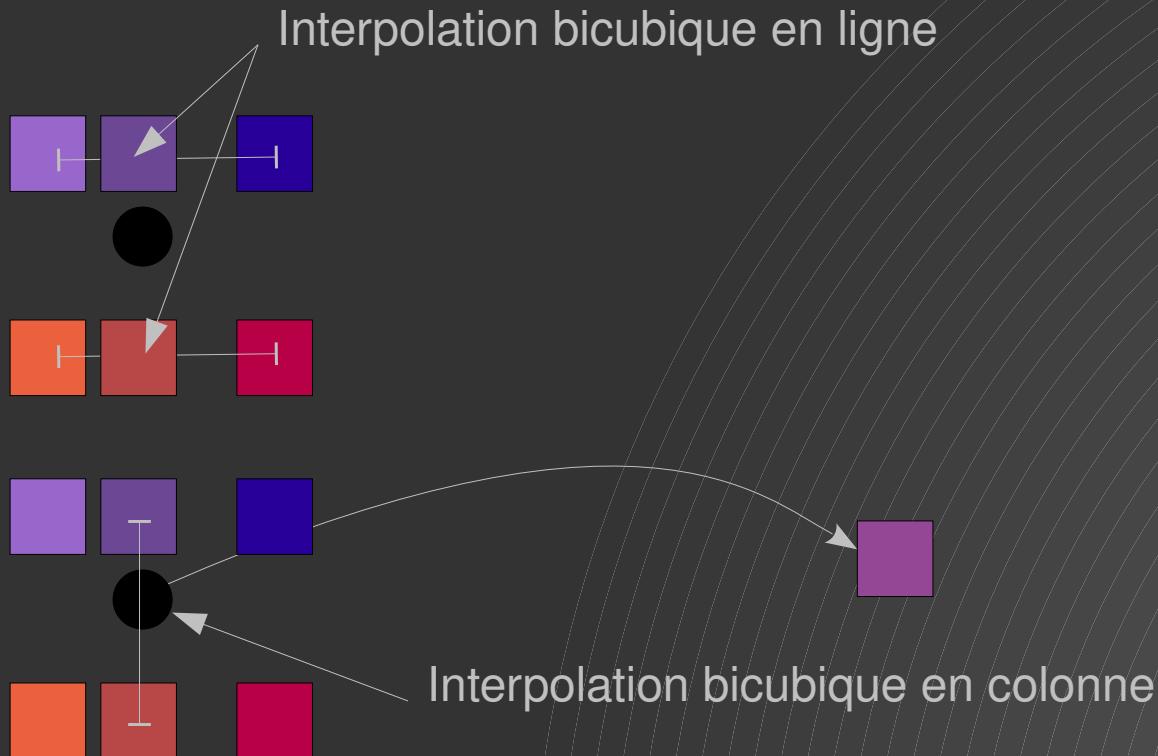
$$f'(1) = 3a+2b+c$$

On peut donc en conclure les coefficients  $a, b, c, d$  du polynôme et donc interpoler les valeurs intermédiaires du signal entre 0 et 1

# Problème de précision

## L'interpolation

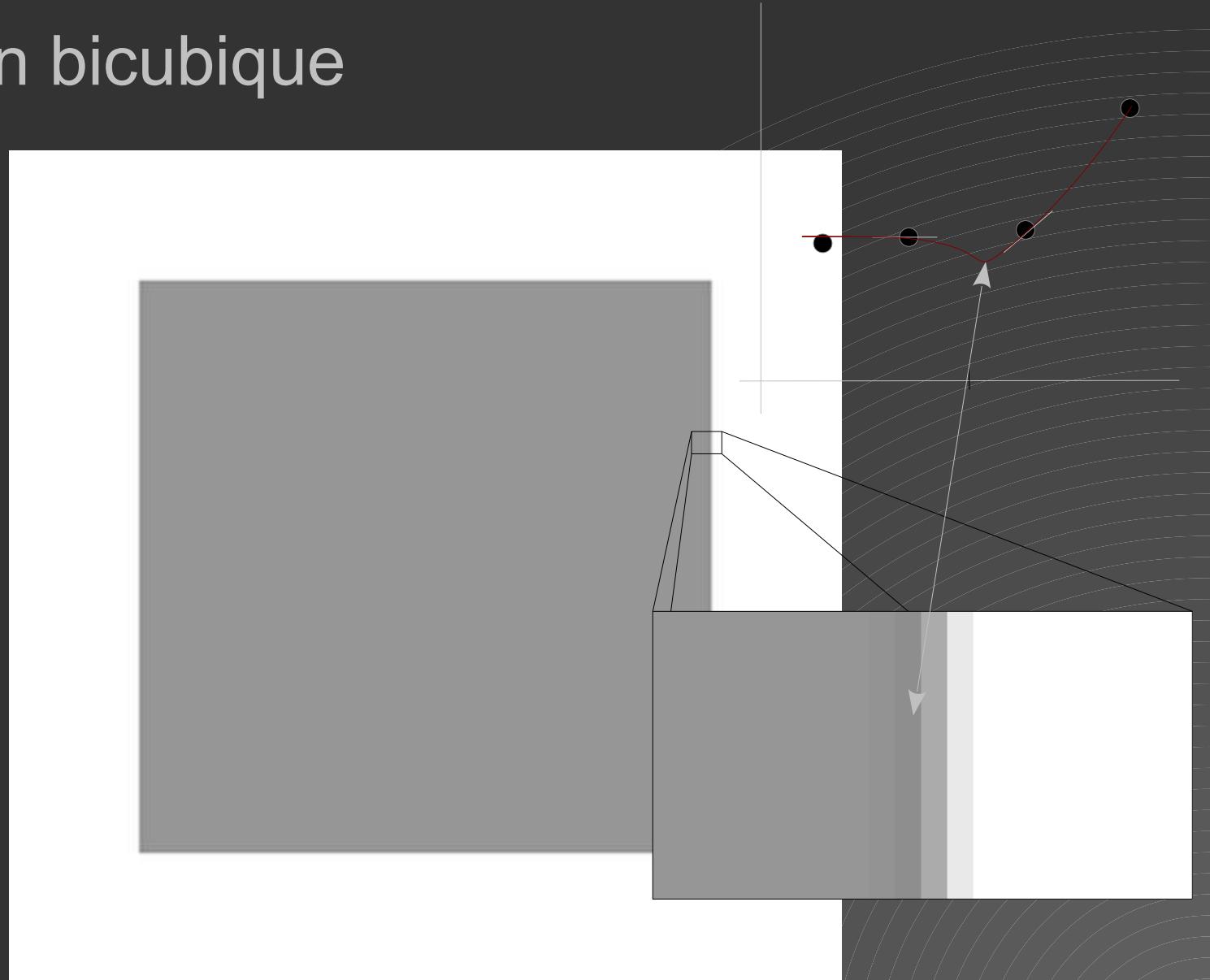
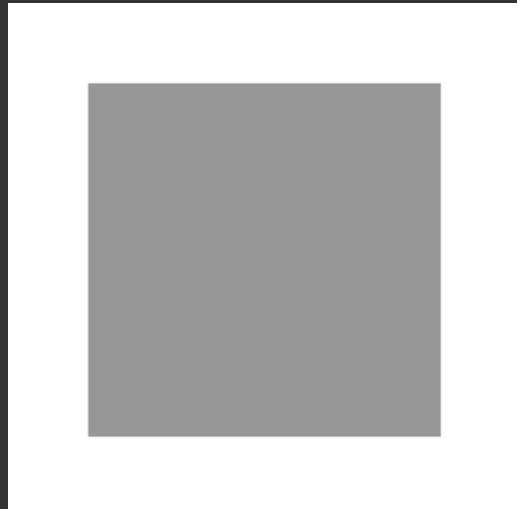
- Interpolation bicubique



# Problème de précision

## L'interpolation

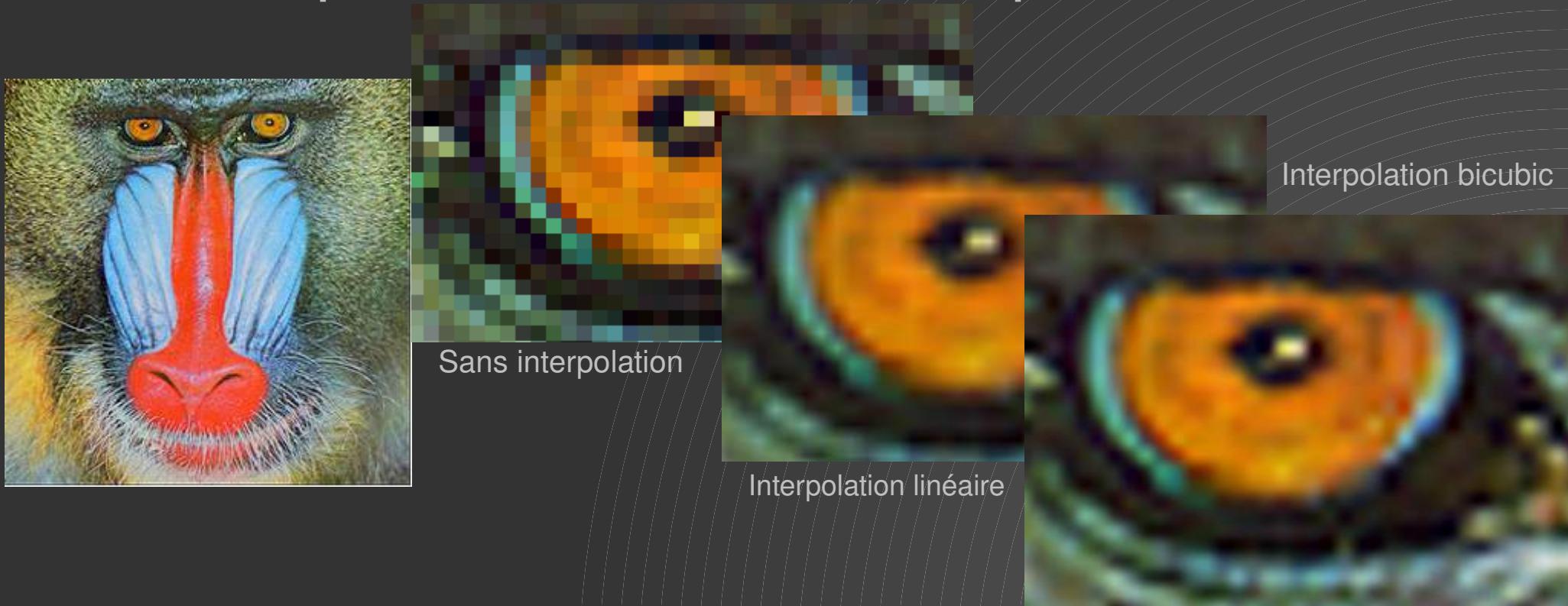
- Interpolation bicubique
  - Artéfact



# Problème de précision

## L'interpolation

- Il existe d'autres méthodes d'interpolation
- Pour faire le choix de l'interpolation, il faut faire un compromis entre vitesse et qualité



# Conclusion

- Codage de l'image et de la couleur
  - Espaces de couleurs, passage d'un espace à l'autre
- Applications :
  - Effet sépia
  - Transformation artistiques
  - Morphing
- Correction gamma
- Interpolation

# A Suivre...