Introduction au Traitement d'Images

Séance 3

Isabelle Debled-Rennesson
debled@loria.fr

Contenu

- Codage des images
- Histogrammes
- **Transformations géométriques sur les images**

Retour sur les exercices de la Séance 1

Exercice 3.a – images R, G et B

- Idée : créer les 3 fichiers en parallèle pendant la lecture du fichier source

```
s = bfreader.readLine();
while(s != null){
        //Ecriture de la composante rouge
        bfwR.write(s+"\n");
        bfwR.write("0\n");
        bfwR.write("0\n");
        //Ecriture de la composante verte
        s = bfreader.readLine();
        bfwG.write("0\n");
        bfwG.write(s+"\n");
        bfwG.write("0\n");
        //Ecriture de la composante bleue
        s = bfreader.readLine();
        bfwB.write("0\n");
        bfwB.write("0\n");
        bfwB.write(s+"\n");
        //lire une nouvelle ligne
        s = bfreader.readLine();
}
```

Exercice 3.b et c

- Transformation d’une image couleur en image en niveaux de gris

```
r = bfreader.readLine();
while(r != null){
        g = bfreader.readLine();
        b = bfreader.readLine();
        gris= Math.round(Float.parseFloat(r)*0.299+0.587*Float.parseFloat(g)
+0.114*Float.parseFloat(b) );
        bfwG.write(gris+"\n");
        r = bfreader.readLine();
}
```

- « Inversion » des couleurs d’une image

```
s = bfreader.readLine();
while(s != null){
        bfwN.write((255-Integer.parseInt(s))+"\n");
        s = bfreader.readLine();
}
```

Exercice3.d

- Un programme qui demande à l'utilisateur un nom de fichier image et qui crée le fichier image correspondant à l'image initiale avec une taille divisée par deux. Pour cela remplacer chaque carré de deux pixels sur deux pixels par un pixel dont la couleur est la moyenne de celle des quatre pixels qu'il remplace.

Exercice3.d

- Principe
  - Recopie dans h et l de la largeur et la hauteur de l’image source
  - Recopie dans Im[*ligne*][*colonne*] de tous les éléments de l’image

```
//Création de la nouvelle image à partir d’une image source en
niveaux de gris
    for(int i=0;i<h/2;i++)
            for(int j=0;j<l/2;j++){
            int pix =(int)((Im[2*i][2*j] + Im[2*i+1][2*j] +
Im[2*i][2*j+1]+ Im[2*i+1][2*j+1])/4);
    bfPetit.write(""+pix+"\n");
    }
```

Exercice3.d

- Principe
  - Recopie dans h et l de la largeur et la hauteur de l'image source
  - Recopie dans Im[*h*][*3*l*] de tous les éléments de l'image

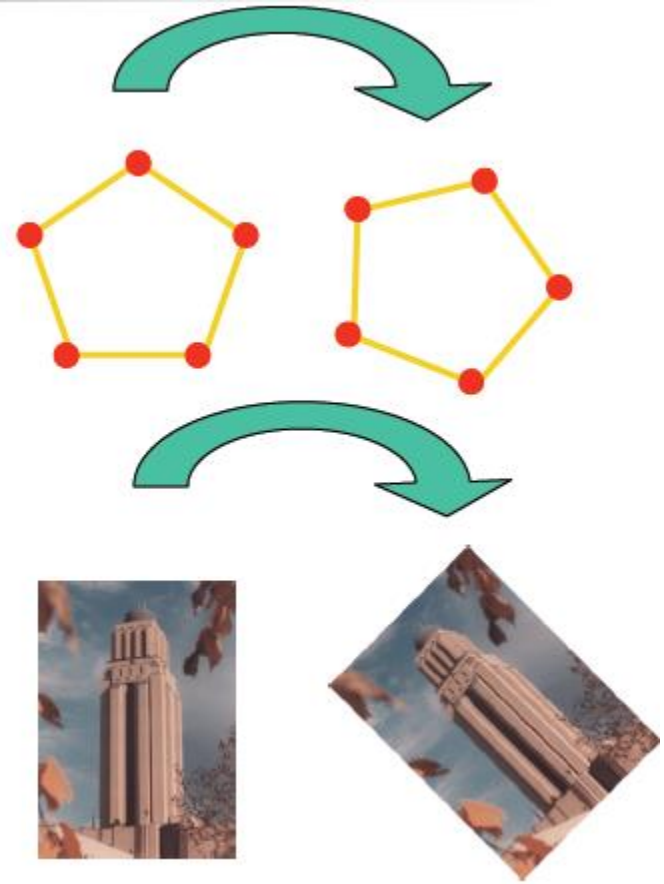
```
//Création de la nouvelle image à partir d'une image source en
couleur
for(int i=0;i<h/2;i++)
    for(int j=0;j<l/2;j++){
            pix = (int)((Im[2*i][6*j] + Im[2*i][6*j+3] +
Im[2*i+1][6*j] + Im[2*i+1][6*j+3])/4);
            bfPetit.write(""+pix+"\n");   //écriture compos rouge
                    pix = (int)((Im[2*i][6*j+1] + Im[2*i][6*j+4] +
Im[2*i+1][6*j+1] + Im[2*i+1][6*j+4])/4);
            bfPetit.write(""+pix+"\n");  //écriture compos verte
            pix = (int)((Im[2*i][6*j+2]+Im[2*i][6*j+5] +
Im[2*i+1][6*j+2] + Im[2*i+1][6*j+5])/4);
            bfPetit.write(""+pix+"\n");   //écriture compos bleue

}
```

Transformations géométriques sur les images

Sources : Anne Vialard, Alain Boucher, livre de Diane Lingrand

Transformations géométriques sur des images vectorielles/bitmaps

- Objet vectoriel : on transforme les sommets (ou points de contrôle) et on retrace
- Objet bitmap : calcul pour chaque pixel



Transformations géométriques

- Translation
- Changement d'échelle (homothétie) - scaling
- Rotation
- Symétrie - flip
- Cisaillement - shear







Changement d’échelle

- Première idée : agrandissement d’image par copie des pixels
- Exemple : multiplication par 2 de la taille de l’image

12	14
15	17

→

12	12	14	14
12	12	14	14
15	15	17	17
15	15	17	17

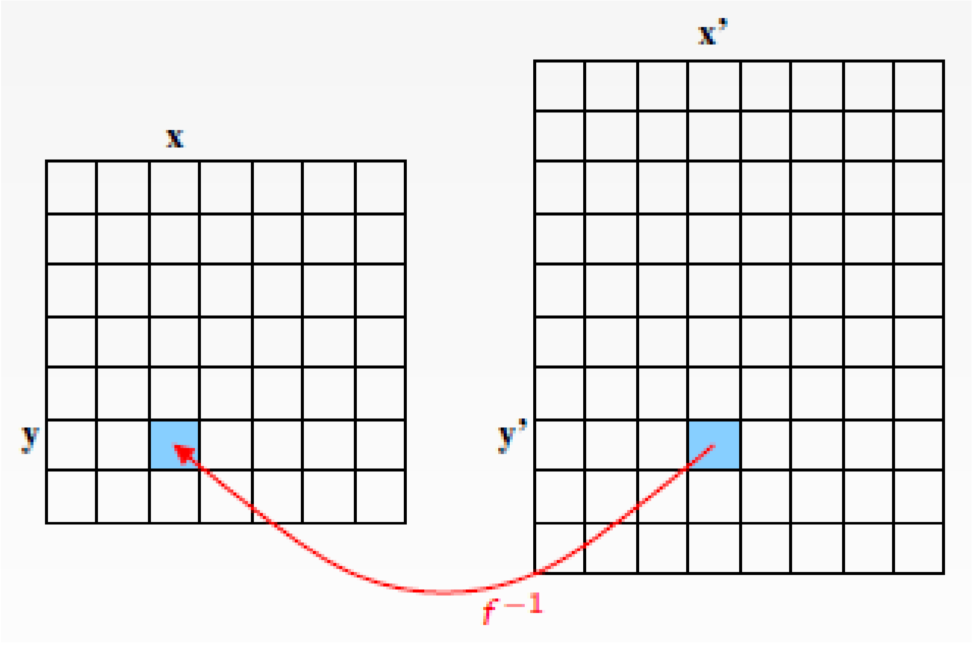
Limite de cette approche ?

Transformations géométriques – rappel 1

- Transformation géométrique
  - f: $(x,y) \rightarrow (x',y')$
  - Translation de vecteur $t(t_x,t_y)$ définie par :
    - $x' = x + t_x$
    - $y' = y + t_y$
  - Homothétie de rapport $\lambda(\lambda_x, \lambda_y)$ par rapport à l'origine
    - $x' = \lambda_x * x$
    - $y' = \lambda_y * y$
  - Homothétie de centre Mo(xo, yo)
    - $x' = xo + \lambda_x * (x - xo)$
    - $y' = yo + \lambda_y * (y - yo)$

Changement d'échelle

- Principe général
  - **Recherche du pixel antécédent** : pour chaque pixel de l'image résultat, on cherche le pixel correspondant dans l'image initiale.
    - Transformation géométrique $(x , y) \rightarrow (x' , y') = f (x , y)$
    - Transformation inverse $(x' , y') \rightarrow (x , y) = f^{-1} (x' , y')$



Changement d'échelle - calcul

- Changement d'échelle = homothétie de centre l'origine
- Soient Sx et Sy les facteurs d'échelle suivant chaque axe (agrandissement ou réduction)

  x' = Sx *x **x = 1/Sx *x'**

  y' = Sy *y **y = 1/Sy *y'**

- Algorithme

```
W' = Sx*W
H' = Sy*H
Créer l'image résultat R de taille W', H'
for(y=0; y<H'; y++)
    for(x=0; x<W'; x++)
        R(x, y) = I(x/Sx, y/Sy)
```

Changement d'échelle - calcul

- Changement d'échelle = homothétie de centre l'origine
- Soient Sx et Sy les facteurs d'échelle suivant chaque axe (agrandissement ou réduction)

  x' = Sx *x **x = 1/Sx *x'**

  y' = Sy *y **y = 1/Sy *y'**

- Algorithme

```
W' = Sx*W
H' = Sy*H
Créer l'image résultat R de taille W', H'
for(y=0; y<H'; y++)
    for(x=0; x<W'; x++)
        R(x, y) = I(x/Sx, y/Sy)
```

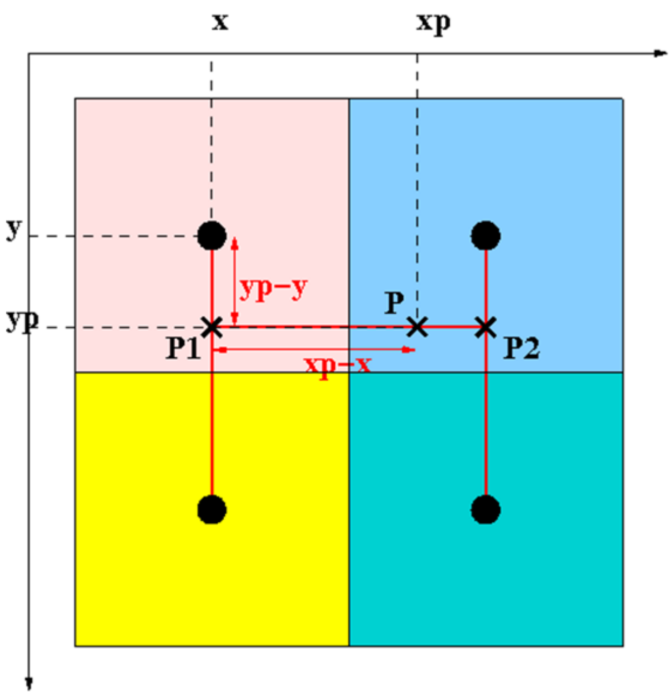
Problème : **x/Sx et y/Sy ne sont pas forcément des entiers**

Changement d’échelle - aliasing

- Changement d’échelle avec arrondi entier des coordonnées de l’antécédent = interpolation au plus proche voisin

Interpolation bilinéaire

- Permet d'atténuer l'aliasing
- L'**interpolation bilinéaire** = méthode d'interpolation pour les fonctions de 2 variables sur une grille régulière.
  - permet de calculer la valeur d'une fonction en un point quelconque, à partir de ses 2 plus proches voisins dans chaque direction.



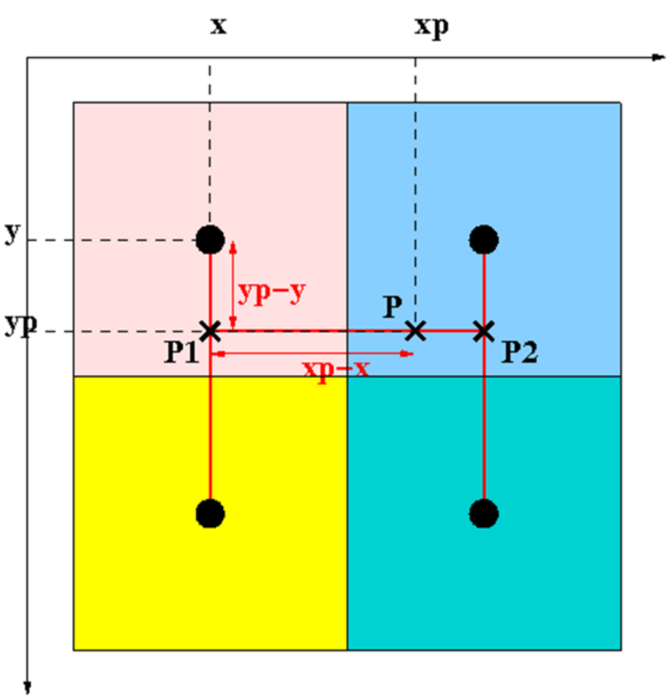
Objectif : obtenir une fonction bilinéaire
$f(x,y)= ax +by +cxy +d$

$f(x_P,y_P)$ est la valeur I_P d'intensité interpolée au point de coordonnées $P(x_P, y_P)$ et a,b,c,d sont des constantes déterminées à partir des intensité des 4 voisins de P : (x,y) ; (x+1,y) ; (x,y+1) ; (x+1,y+1)

→ 4 équations à 4 inconnues

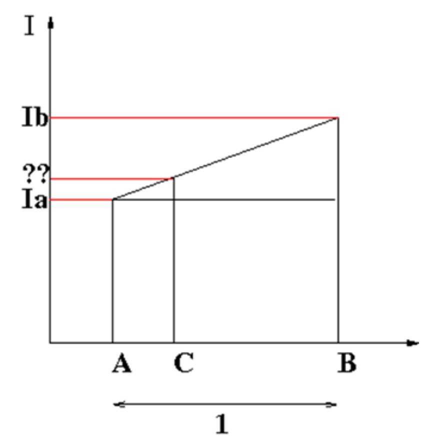
Interpolation bilinéaire

- Permet d'atténuer l'aliasing
- L'**interpolation bilinéaire** = méthode d'interpolation pour les fonctions de 2 variables sur une grille régulière.
  - permet de calculer la valeur d'une fonction en un point quelconque, à partir de ses 2 plus proches voisins dans chaque direction.



Autre méthode pour calculer I_P : succession de deux interpolations linéaires en considérant les points P1 et P2

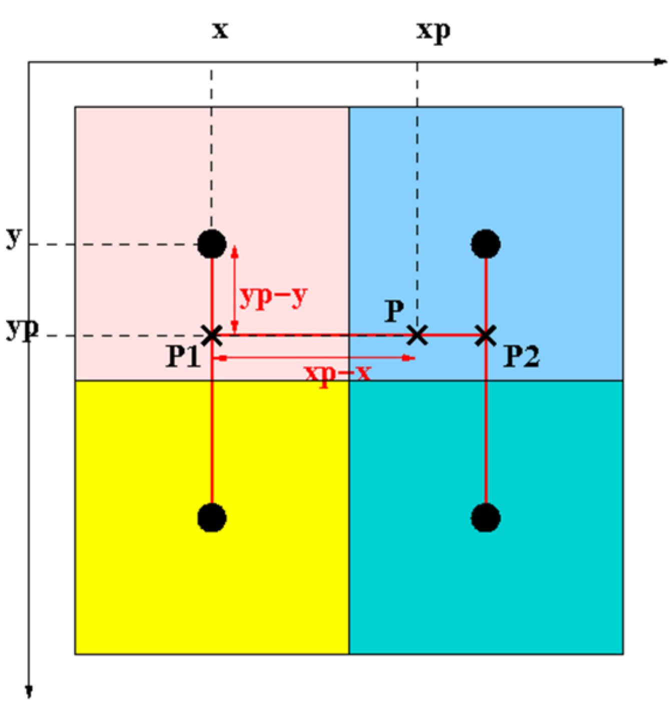
Calcul de I_{P1} en considérant I(x,y) et I(x,y+1)
Calcul de I_{P2} en considérant I(x+1,y) et I(x+1,y+1)
Calcul de I_P en considérant I_{P1} et I_{P2}



$I_C = I_A + d(AC)(I_B - I_A)$

Interpolation bilinéaire

- Permet d'atténuer l'aliasing
- L'**interpolation bilinéaire** = méthode d'interpolation pour les fonctions de 2 variables sur une grille régulière.
  - permet de calculer la valeur d'une fonction en un point quelconque, à partir de ses 2 plus proches voisins dans chaque direction.



Autre méthode pour calculer I_P : succession de deux interpolations linéaires en considérant les points P1 et P2

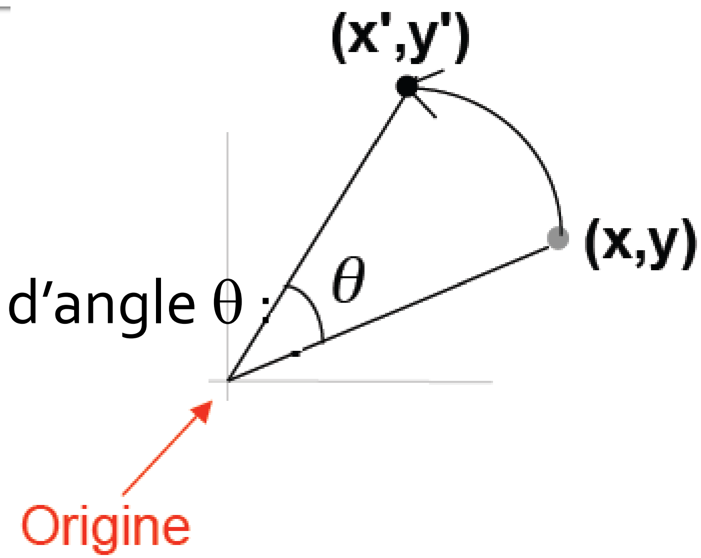
$$I_{P1} = I(x, y) + (y_P - y)(I(x , y + 1) - I(x , y))$$
$$I_{P2} = I(x + 1 , y) + (y_P - y)(I(x + 1 , y + 1) - I(x + 1 , y))$$

$$I_P = I_{P1} + (x_P - x)(I_{P2} - I_{P1})$$

Changement d'échelle avec interpolation bilinéaire

Transformations géométriques – rappel 2

- Transformation géométrique
  - f: $(x,y) \rightarrow (x',y')$
  - Rotation antihoraire autour de l'origine d'angle θ :
    - $x' = x\cos(\theta) - y\sin(\theta)$
    - $y' = y\cos(\theta) + x\sin(\theta)$



  - Démonstration

Coordonnées polaires : $x = r\cos(\alpha)$ et $y = r\sin(\alpha)$

Après rotation d'angle θ : $x' = r\cos(\alpha + \theta)$ et $y' = r\sin(\alpha + \theta)$

$$X' = r\cos(\alpha)\cos(\theta) - r\sin(\alpha)\sin(\theta)$$

$$y' = r\cos(\alpha)\sin(\theta) + r\cos(\theta)\sin(\alpha)$$

$$x' = x\cos(\theta) - y\sin(\theta)$$

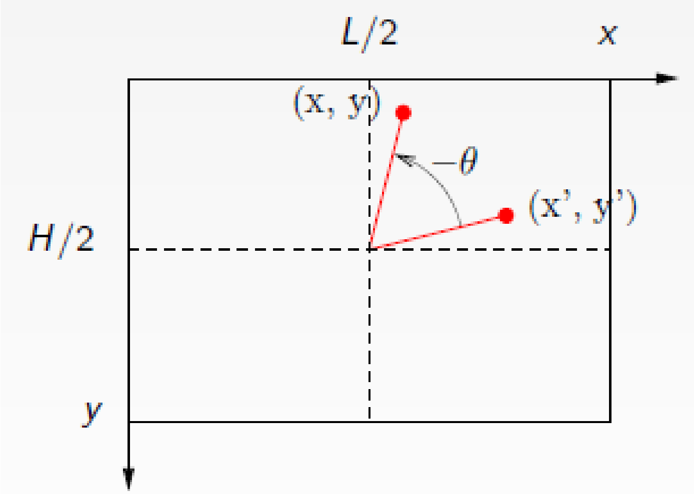
$$y' = y\cos(\theta) + x\sin(\theta)$$

Rotation de l'image autour de son centre

- Rotation **horaire** de centre (L/2,H/2)

$x' = L/2 + (x-L/2)\cos(\theta) + (y-H/2)\sin(\theta)$

$y' = H/2 + (y-H/2)\cos(\theta) - (x-L/2)\sin(\theta)$



- Recherche de l'antécédent

$x = L/2 + (x' - L/2)\cos(\theta) - (y' - H/2)\sin(\theta)$

$y = H/2 + (y' - H/2)\cos(\theta) + (x' - L/2)\sin(\theta)$

Rotation avec interpolation au plus proche voisin

Rotation avec interpolation bilinéaire

Cisaillement

- Transformation faisant penser à un étirement selon un axe
  - Cisaillement selon les abscisses

    $x' = x + sh_x * y$

    $y' = y$

  - Cisaillement selon les ordonnées

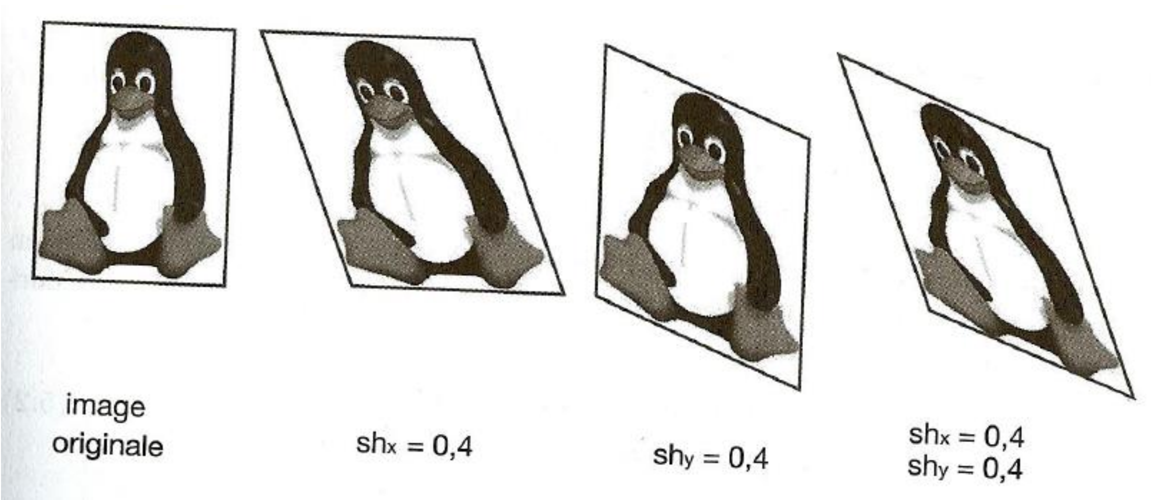
    $x' = x$

    $y' = y + sh_y * x$

  - Cisaillement général

    $x' = x + sh_x * y$

    $y' = y + sh_y * x$

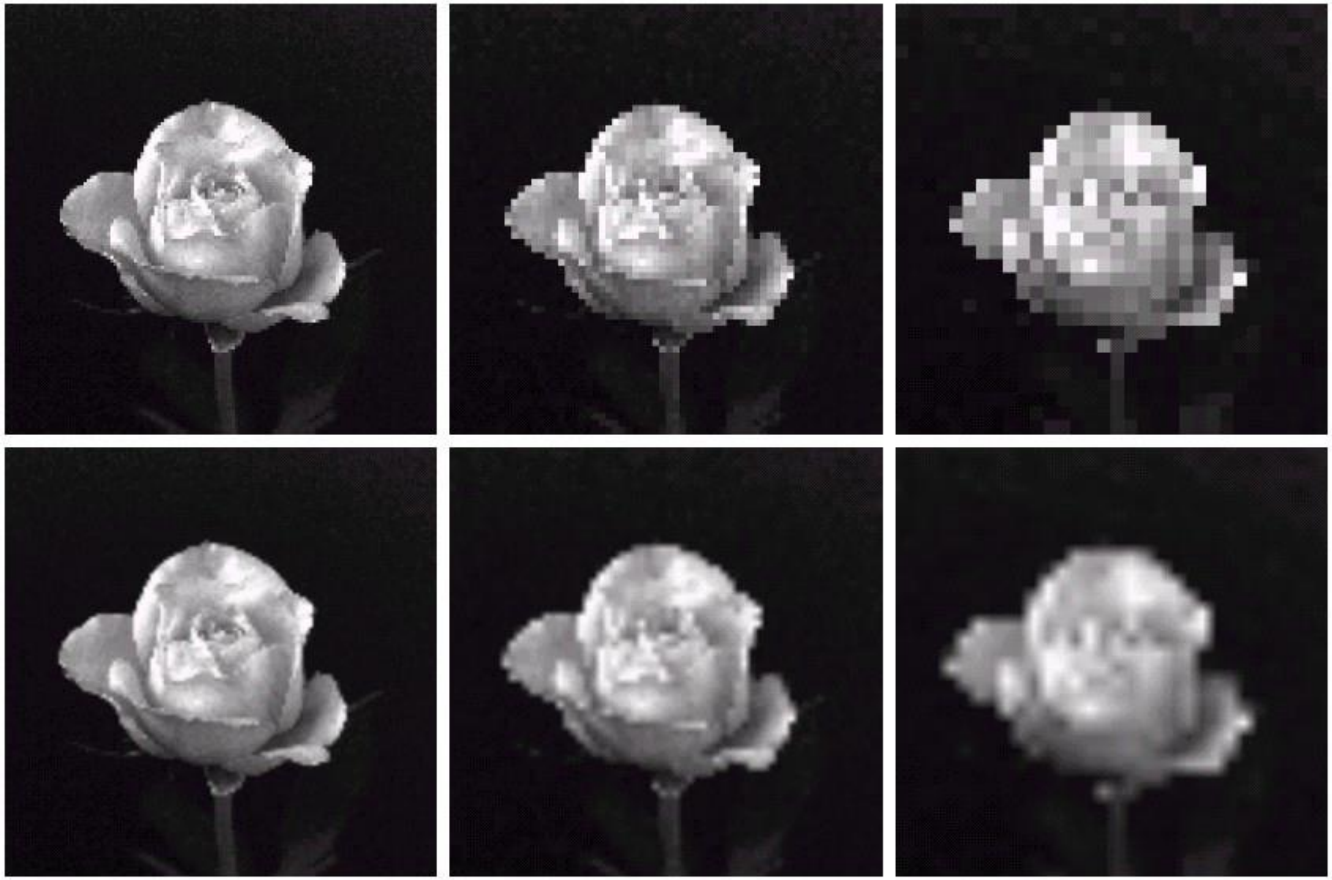


Changement d'échelle avec interpolation

FIGURE 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256.

Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall, 2002.

Exemples d’interpolation



a	b	c
d	e	f

FIGURE 2.25 Top row: images zoomed from 128 × 128, 64 × 64, and 32 × 32 pixels to 1024 × 1024 pixels, using nearest neighbor gray-level interpolation. Bottom row: same sequence, but using bilinear interpolation.

Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall, 2002.