TP 2. Filtrage

1. Convolution avec un masque 3x3. Application : Renforcement de contraste

Lecture N&B :

I := READ_IMAGE("coco.bmp")

Nombre de colonnes : Nx := cols(I)
Nx = 147

Nombre de lignes : Ny := rows(I)
Ny = 195

Q2.1. Ecrire sous Python la fonction de convolution Convol(I,Mask) ou utiliser la fonction intégrée signal.convolve2d(I, Mask) de la librairie scipy de Python.
Interprétation des résultats de l'opérateur de renforcement de contraste.

Fonction Convol(I,Mask)

```
Convol(I, Mask) := | for y ∈ 1 .. rows(I) − 2
                   |   for x ∈ 1 .. cols(I) − 2
                   |     | som ← 0
                   |     | for j ∈ 0 .. 2
                   |     |   for i ∈ 0 .. 2
                   |     |     som ← som + (I[y−j+1, x−i+1] · Mask[j, i])
                   |     | som ← 255 if som > 255
                   |     | som ← 0 if som < 0
                   |     | J[y, x] ← round(som)
                   | J
```

Renforcement de Contraste (k = 1)

$$M := \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

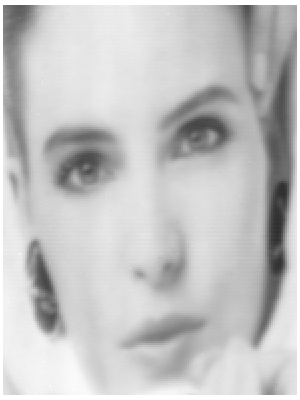
J := Convol(I, M)

J2 := Convol(J, M)

Fonction intégrée / **Gimp**

K := convol2d(I, M)

Image originale



I

Image à contraste renforcé

J

Contraste renforcé 2 fois

J2

Image à contraste renforcé

K

2. Dérivation d'une image

Lecture N&B :

$I := \mathrm{READ\_IMAGE}(\text{"cube.bmp"})$

Nombre de colonnes : $Nx := \mathrm{cols}(I)$ $Nx = 75$

Nombre de lignes : $Ny := \mathrm{rows}(I)$ $Ny = 120$

Q2.2. Ecrire la fonction Derivation(I,Mask3x3) basée sur la convolution. Déterminer le masque 2D de l'opérateur de dérivation ∇_{2D}. Interprétation des résultats de l'opérateur de dérivation.

Fonction Derivation(I,Mask3x3)

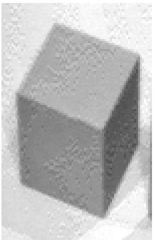
$\mathrm{Derivation}(I, \mathrm{Mask3x3}) :=$

$M :=$

$D := \mathrm{Derivation}(I, M)$

$R := \mathrm{RecadrageDyn}(D)$

Image originale



I

Image dérivée à dynamique recadrée

R

3. Renforcement de contraste : Opérateur de Chanda

$I := \mathrm{READ\_IMAGE}(\text{"coco.bmp"})$

Nombre de colonnes : $Nx := \mathrm{cols}(I)$ $Nx = 147$

Nombre de lignes : $Ny := \mathrm{rows}(I)$ $Ny = 195$

Q2.3. Coder sous Python l'algorithme de renforcement de contraste de Chanda et comparer ses résultats avec ceux de l'opérateur de renforcement de contraste basé sur la convolution.

Fonction Chanda(I,n) *n représente l'ordre du filtre (son intensité d'action) : (choix par défaut : n = 1)*

$$d_0 \equiv \begin{pmatrix} 0&0&0&0&1\\ 0&0&0&1&1\\ 0&0&1&1&1\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \quad d_1 \equiv \begin{pmatrix} 0&0&1&1&1\\ 0&0&1&1&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix}$$

$$d_4 \equiv \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 1&1&1&0&0\\ 1&1&0&0&0\\ 1&0&0&0&0 \end{pmatrix} \quad d_5 \equiv \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&1&1&0&0\\ 1&1&1&0&0 \end{pmatrix}$$

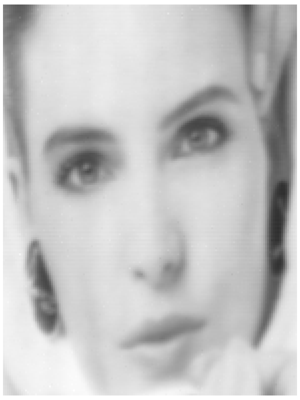
$$d_2 \equiv \begin{pmatrix} 1&1&1&0&0\\ 0&1&1&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \quad d_3 \equiv \begin{pmatrix} 1&0&0&0&0\\ 1&1&0&0&0\\ 1&1&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix}$$

$$d_6 \equiv \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&1&1&0\\ 0&0&1&1&1 \end{pmatrix} \quad d_7 \equiv \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&1&1\\ 0&0&0&1&1\\ 0&0&0&0&1 \end{pmatrix}$$

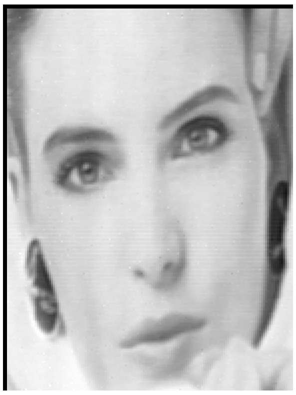
```
Chanda(I,n) := | for y ∈ 2 .. rows(I) − 3
               |   for x ∈ 2 .. cols(I) − 3
               |     | SDG ← 0
               |     | SG ← 0
               |     | for k ∈ 0 .. 7
               |     |   | S ← 0
               |     |   | for j ∈ 0 .. 4
               |     |   |   for i ∈ 0 .. 4
               |     |   |     S ← S + [I_(y+j−2, x+i−2) · (d_k)_(j,i)]
               |     |   | S ← S / 6
               |     |   | S2 ← 0
               |     |   | for j ∈ 0 .. 4
               |     |   |   for i ∈ 0 .. 4
               |     |   |     S2 ← S2 + [I_(y+j−2, x+i−2) · (d_k)_(j,i)]^2
               |     |   | S2 ← S2 / 6
               |     |   | Δ ← S − I_(y,x)
               |     |   | G ← (|S2 − S^2|)^(n/2)
               |     |   | SDG ← SDG + Δ·G
               |     |   | SG ← SG + G
               |     | J_(y,x) ← I_(y,x) − SDG / SG
               |     | J_(y,x) ← 0 if J_(y,x) < 0
               |     | J_(y,x) ← 255 if J_(y,x) > 255
               | J
```

$n := 1$ $C := Chanda(I, n)$ $E := Chanda(C, n)$

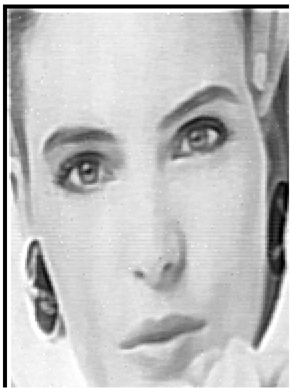
Image originale | Image à contraste renforcé | Image à contraste renforcé 2 fois



I



C



E

4. Lissage

$I := READ\_IMAGE("cube.bmp")$

Nombre de colonnes : $Nx := cols(I)$ $Nx = 75$

Nombre de lignes : $Ny := rows(I)$ $Ny = 120$

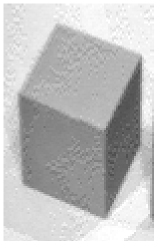
Lissage par la moyenne :

Q2.4. Déterminer le masque de l'opérateur de lissage 3x3 par la moyenne. Interprétation des résultats.

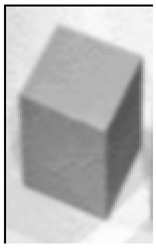
$$L := \begin{pmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{pmatrix}$$

$J := Convol(I, L)$

Image originale | Image lissée par la moyenne



I



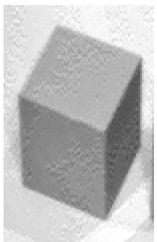
J

Lissage médian : **Q2.5. Programmer l'opérateur de lissage median 3x3. Interprétation des résultats.**

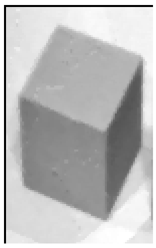
```
Median(I) ≡ | for y ∈ 1 .. rows(I) − 2
            |   for x ∈ 1 .. cols(I) − 2
            |     | for i ∈ 0 .. 2
            |     |   for j ∈ 0 .. 2
            |     |     v[3·i+j] ← I[y+i−1, x+j−1]
            |     | vtri ← sort(v)
            |     | J[y,x] ← vtri[4]
            | J
```

$J := Median(I)$

Image originale | Image lissée par le médian



I



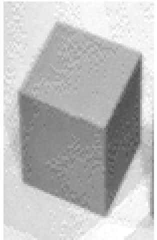
J

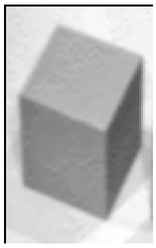
Lissage Gaussien :

Q2.6. Appliquer le masque 2D de l'opérateur de lissage Gaussien 3x3. Interprétation des résultats.

$$M := \begin{pmatrix} 0.075 & 0.124 & 0.075 \\ 0.124 & 0.204 & 0.124 \\ 0.075 & 0.124 & 0.075 \end{pmatrix} \qquad J := \mathrm{Convol}(I, M)$$

Image originale Image lissée par la Gaussienne





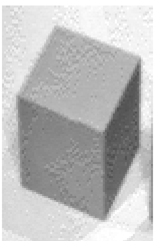
I J

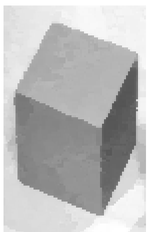
Lissage de Nagao : **Q2.7. Interprétation des résultats du lissage de Nagao.**

```
Nagao(I) ≡ | for y ∈ 2..rows(I) − 3
           |   for x ∈ 2..cols(I) − 3
           |     | for j ∈ 0..2
           |     |   for i ∈ 0..2
           |     |     | for m ∈ 0..2
           |     |     |   for k ∈ 0..2
           |     |     |     Tmp_{m·3+k} ← I_{y+j+m−2, x+i+k−2}
           |     |     | Moy_{3·j+i} ← mean(Tmp)  if i·j ≠ 1
           |     |     | Moy_{3·j+i} ← 0  otherwise
           |     |     | Var_{3·j+i} ← max(Tmp) − min(Tmp)  if i·j ≠ 1
           |     |     | Var_{3·j+i} ← 255  otherwise
           |     | Mini ← min(Var)
           |     | for i ∈ 0..8
           |     |   W_{y−2, x−2} ← Moy_i  if Var_i = Mini
           | W
```

$J := \mathrm{Nagao}(I)$

Image originale Image lissée par Nagao





I J

5. FFT : débruitage d'image - *image denoising*

$I := \text{READ\_IMAGE}(\text{"boat256noisy.bmp"})$

```
denoiseCFFT(I) := | for x ∈ 0 .. cols(I) − 1
                  |   | s ← I^⟨x⟩
                  |   | N ← length(s)
                  |   | S ← CFFT(s)
                  |   | for n ∈ trunc(N/4) .. trunc(3·N/4)
                  |   |   S_n ← 0
                  |   | y ← ICFFT(S)
                  |   | J^⟨x⟩ ← y
                  | J
```

```
denoiseDFT(I) := | for x ∈ 0 .. cols(I) − 1
                 |   | s ← I^⟨x⟩
                 |   | N ← length(s)
                 |   | S ← DFT(s)
                 |   | for n ∈ trunc(N/4) .. trunc(3·N/4)
                 |   |   S_n ← 0
                 |   | y ← iDFT(S)
                 |   | J^⟨x⟩ ← y
                 | J
```

débruitage FFT sans perte de résolution

débruitage FFT colonne par colonne $K := \text{denoiseCFFT}(I)$ ou $L := \text{denoiseDFT}(I)$

Q2.8. Ecrire sous Python la fonction de débruitage basée sur la Transformée de Fourier (TF) (fonctions numpy.fft.fft(s) (TF) et numpy.fft.ifft(S) (TF inverse) de la librairie numpy de Python). Tester sur l'image "kit256noisy.bmp". Interprétation des résultats.

I

K

L

Fonctions Bibliothèques :

$$\mathrm{Histo}(I) \equiv \left| \begin{array}{l} \text{for } ng \in 0..255 \\ \quad H_{ng} \leftarrow 0 \\ \text{for } y \in 0..\mathrm{rows}(I)-1 \\ \quad \text{for } x \in 0..\mathrm{cols}(I)-1 \\ \quad\quad \left| \begin{array}{l} ng \leftarrow I_{y,x} \\ H_{ng} \leftarrow H_{ng}+1 \end{array} \right. \\ H \end{array} \right.$$

$$\mathrm{RecadrageDyn}(I) \equiv \left| \begin{array}{l} mini \leftarrow \min(I) \\ maxi \leftarrow \max(I) \\ \Delta \leftarrow \dfrac{255}{maxi - mini} \\ \text{for } y \in 0..\mathrm{rows}(I)-1 \\ \quad \text{for } x \in 0..\mathrm{cols}(I)-1 \\ \quad\quad J_{y,x} \leftarrow \mathrm{trunc}\left[\left(I_{y,x} - mini\right)\cdot\Delta\right] \\ J \end{array} \right.$$

$$\mathrm{HFnoise}(N) \equiv \left| \begin{array}{l} \text{for } n \in 0..N-1 \\ \quad \left| \begin{array}{l} b_n \leftarrow 0.5 \ \text{ if } \mathrm{mod}(n,2) = 0 \\ b_n \leftarrow -0.5 \ \text{ otherwise} \end{array} \right. \\ b \end{array} \right.$$

$$\mathrm{HFnoise2}(N) \equiv \left| \begin{array}{l} \text{for } n \in 0..N-1 \\ \quad \left| \begin{array}{l} b_n \leftarrow 0.5\cdot\mathrm{trunc}(\mathrm{rnd}(2)) \ \text{ if } \mathrm{mod}(n,2) = 0 \\ b_n \leftarrow -0.5\cdot\mathrm{trunc}(\mathrm{rnd}(2)) \ \text{ otherwise} \end{array} \right. \\ b \end{array} \right.$$

$$\mathrm{downsampleX}(I,f) \equiv \left| \begin{array}{l} \text{for } x \in 0..\mathrm{cols}(I)-1 \\ \quad \left| \begin{array}{l} X \leftarrow \mathrm{trunc}\left(\dfrac{x}{f}\right) \\ J^{\langle X \rangle} \leftarrow I^{\langle x \rangle} \end{array} \right. \\ J \end{array} \right.$$

$$\mathrm{downsampleY}(I,f) \equiv \left| \begin{array}{l} \text{for } y \in 0..\mathrm{rows}(I)-1 \\ \quad \left| \begin{array}{l} Y \leftarrow \mathrm{trunc}\left(\dfrac{y}{f}\right) \\ J^{\langle Y \rangle} \leftarrow \left(I^T\right)^{\langle y \rangle} \end{array} \right. \\ J^T \end{array} \right.$$

$$\mathrm{downsample}(I,f) \equiv \left| \begin{array}{l} J \leftarrow \mathrm{downsampleX}(I,f) \\ K \leftarrow \mathrm{downsampleY}(J,f) \\ K \end{array} \right.$$

$$\mathrm{DFT}(x) \equiv \left| \begin{array}{l} N \leftarrow \mathrm{length}(x) \\ \text{for } k \in 0..N-1 \\ \quad X_k \leftarrow \dfrac{1}{N}\cdot\sum_{n=0}^{N-1}\left(x_n\cdot e^{-i\cdot 2\cdot\pi\cdot k\cdot\frac{n}{N}}\right) \\ X \end{array} \right.$$

$$\mathrm{bruitage}(I,A) \equiv \left| \begin{array}{l} \text{for } x \in 0..\mathrm{cols}(I)-1 \\ \quad \left| \begin{array}{l} v \leftarrow I^{\langle x \rangle} \\ N \leftarrow \mathrm{length}(v) \\ w \leftarrow A\cdot\mathrm{HFnoise}(N) \\ \text{for } n \in 0..N-1 \\ \quad \left| \begin{array}{l} u_n \leftarrow \mathrm{trunc}\left(v_n + w_n\right) \\ u_n \leftarrow 255 \ \text{ if } u_n > 255 \\ u_n \leftarrow 0 \ \text{ if } u_n < 0 \end{array} \right. \\ J^{\langle x \rangle} \leftarrow u \end{array} \right. \\ J \end{array} \right.$$

$$\mathrm{iDFT}(X) \equiv \left| \begin{array}{l} N \leftarrow \mathrm{length}(X) \\ \text{for } n \in 0..N-1 \\ \quad x_n \leftarrow \sum_{k=0}^{N-1}\left(X_k\cdot e^{i\cdot 2\cdot\pi\cdot k\cdot\frac{n}{N}}\right) \\ x \end{array} \right.$$