TP 1. Chaîne de vision. Prétraitement

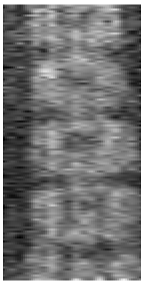
1. Négatif d'une image en 256 niveaux de gris

$I := \text{READ\_IMAGE}(\text{"vertebre.bmp"})$ Nombre colonnes: $Nx := \text{cols}(I)$ $Nx = 69$ Nombre lignes: $Ny := \text{rows}(I)$ $Ny = 138$

$$\text{nega}(I) := \left| \begin{array}{l} Nx \leftarrow \text{cols}(I) \\ Ny \leftarrow \text{rows}(I) \\ \text{for } y \in 0..Ny-1 \\ \quad \text{for } x \in 0..Nx-1 \\ \quad\quad J_{y,x} \leftarrow 255 - I_{y,x} \\ J \end{array} \right.$$

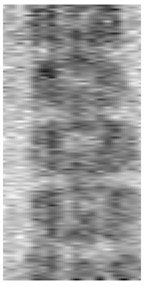
$J := \text{nega}(I)$

Image originale



I

Image traitée



J

2. Négatif d'une image RGB

$R := \text{READ\_RED}(\text{"mandril.bmp"})$ $G := \text{READ\_GREEN}(\text{"mandril.bmp"})$ $B := \text{READ\_BLUE}(\text{"mandril.bmp"})$

$r := \text{nega}(R)$ $g := \text{nega}(G)$ $b := \text{nega}(B)$

Image originale

R, G, B

Image traitée

r, g, b

3. Histogramme d'une image

```
I := READ_IMAGE("boat2.bmp")

Histo(I) := | for ng ∈ 0..255
            |    H_ng ← 0
            | for y ∈ 0..rows(I) − 1
            |    for x ∈ 0..cols(I) − 1
            |       | ng ← I_y,x
            |       | H_ng ← H_ng + 1
            | H
```

Nombre de colonnes : $Nx := cols(I)$ $Nx = 128$

Nombre de lignes : $Ny := rows(I)$ $Ny = 128$

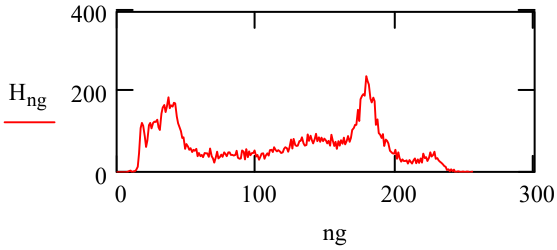
$ng := 0..255$

Image originale

I

$H := Histo(I)$

Histogramme

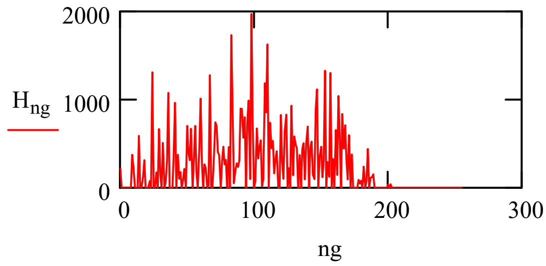


4. Binarisation d'une image

$I := READ\_IMAGE("mandy.bmp")$ $Nx := cols(I)$ $Nx = 256$ $Ny := rows(I)$ $Ny = 246$

```
Binarisation(I, seuil) := | Nx ← cols(I)
                          | Ny ← rows(I)
                          | for y ∈ 0..Ny − 1
                          |    for x ∈ 0..Nx − 1
                          |       | J_y,x ← 0 if I_y,x < seuil
                          |       | J_y,x ← 255 otherwise
                          | J
```

$H := Histo(I)$ *Histogramme de l'image initiale*



$seuil := 100$ $W := Binarisation(I, seuil)$ $H := Histo(W)$

Histogramme de l'image binarisée

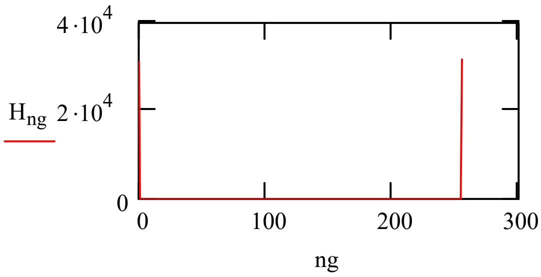
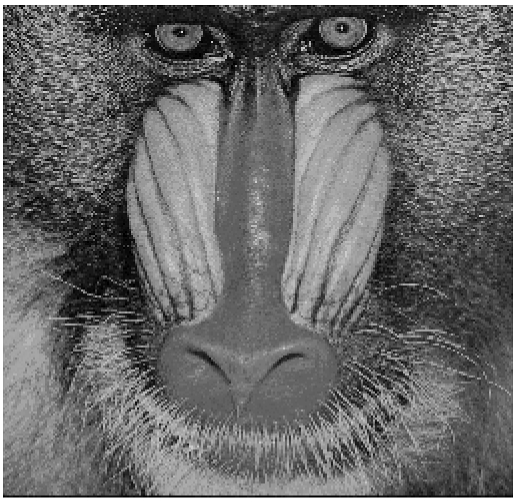


Image originale



I

Image binarisée

W

5. Retournement d'image (Flip) Horizontal et Vertical

A titre d'entraînement, coder sous Python l'algorithme de retournement d'image (Flip) Horizontal / Vertical

$I := \mathrm{READ\_IMAGE}(\text{"boat2.bmp"})$

$$\mathrm{Hflip}(I) := \left| \begin{array}{l} Nx \leftarrow \mathrm{cols}(I) \\ Ny \leftarrow \mathrm{rows}(I) \\ \text{for } \; y \in 0..Ny-1 \\ \quad \text{for } \; x \in 0..Nx-1 \\ \qquad J_{y,x} \leftarrow I_{y,Nx-1-x} \\ J \end{array} \right.$$

$$\mathrm{Vflip}(I) := \left| \begin{array}{l} Nx \leftarrow \mathrm{cols}(I) \\ Ny \leftarrow \mathrm{rows}(I) \\ \text{for } \; y \in 0..Ny-1 \\ \quad \text{for } \; x \in 0..Nx-1 \\ \qquad J_{y,x} \leftarrow I_{Ny-1-y,x} \\ J \end{array} \right.$$

$H := \mathrm{Hflip}(I)$

$V := \mathrm{Vflip}(I)$

Image originale HFlip VFlip



I H V

Image originale HFlip VFlip

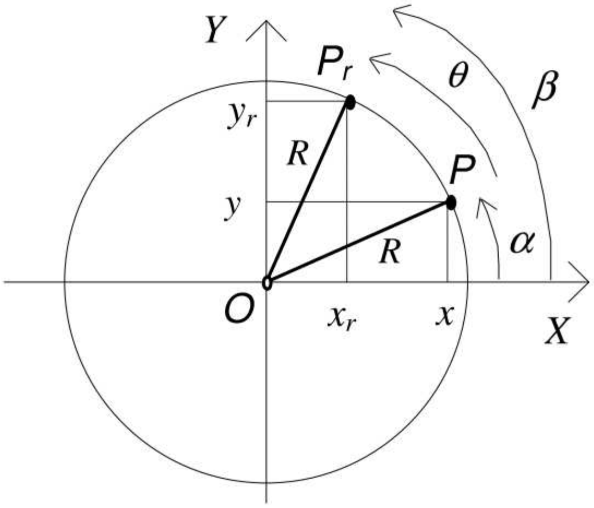


6. Rotation d'image

Rotation d'un angle θ d'un point *P* de coordonnées $(x\ ,y)$ vers un point P_r de coordonnées (x_r, y_r)

Rappels : **coordonnées cartésiennes de *P* :** $\begin{cases} x \\ y \end{cases}$ **coordonnées polaires de *P* :** $\begin{cases} R \\ \alpha \end{cases}$

trigonométrie : $\begin{cases} \cos(a+b) = \cos a \cdot \cos b - \sin a \cdot \sin b \\ \sin(a+b) = \sin a \cdot \cos b + \sin b \cdot \cos a \end{cases}$ $\begin{cases} x^2 + y^2 = R^2 \\ \tan(\alpha) = (y/x) \end{cases} \rightarrow \begin{cases} R = \sqrt{x^2+y^2} \\ \alpha = arc\tan(y/x) \end{cases}$



On a : $\begin{cases} x = R\cdot\cos\alpha \\ y = R\cdot\sin\alpha \end{cases} \rightarrow \begin{cases} \cos\alpha = (x/R) \\ \sin\alpha = (y/R) \end{cases}$ et $\begin{cases} x_r = R\cdot\cos\beta \\ y_r = R\cdot\sin\beta \end{cases} \rightarrow \begin{cases} \cos\beta = (x_r/R) \\ \sin\beta = (y_r/R) \end{cases}$

Soit $\beta = \alpha + \theta$: $\begin{cases} \cos\beta = (x_r/R) \\ \sin\beta = (y_r/R) \end{cases} \rightarrow \begin{cases} \cos(\alpha+\theta) = (x_r/R) \\ \sin(\alpha+\theta) = (y_r/R) \end{cases} \rightarrow \begin{cases} \cos\alpha\cdot\cos\theta - \sin\alpha\cdot\sin\theta = (x_r/R) \\ \sin\alpha\cdot\cos\theta + \sin\theta\cdot\cos\alpha = (y_r/R) \end{cases}$

$\rightarrow \begin{cases} (x/R)\cdot\cos\theta - (y/R)\cdot\sin\theta = (x_r/R) \\ (y/R)\cdot\cos\theta + \sin\theta\cdot(x/R) = (y_r/R) \end{cases} \rightarrow \begin{cases} x\cdot\cos\theta - y\cdot\sin\theta = x_r \\ y\cdot\cos\theta + \sin\theta\cdot x = y_r \end{cases}$

$\rightarrow \begin{cases} x_r = x\cdot\cos\theta - y\cdot\sin\theta \\ y_r = y\cdot\cos\theta + x\cdot\sin\theta \end{cases}$ $\rightarrow$ sous forme *matricielle :* $\boxed{\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}}$

6.1. Rotation en coordonnées cartésiennes (autour du centre (x0,y0))

Q1.1. Quelle est la cause des points noirs provoqués par l'algorithme de Rotation en coordonnées cartésiennes ?

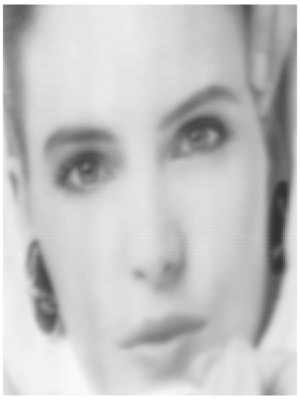
Q1.2. Proposer une solution pour améliorer l'algorithme de Rotation en coordonnées cartésiennes (disparition des points noirs). Expliquer la méthode, la coder sous Python, la tester sur différentes images, consigner les résultats (copies d'écran) et les interpréter (limites éventuelles de la méthode, autres améliorations ...)

$I := \text{READ\_IMAGE}(\text{"coco.bmp"})$ Nombre de colonnes : $Nx := \text{cols}(I)$ Nombre de lignes : $Ny := \text{rows}(I)$

```
RotateCart(I, angle_deg, x0, y0) := | Nx ← cols(I)
                                    | Ny ← rows(I)
                                    | angle_rad ← (angle_deg)·π/180
                                    | for y ∈ 0..Ny − 1
                                    |   for x ∈ 0..Nx − 1
                                    |     | xr ← round[(x − x0)·cos(angle_rad) − (y − y0)·sin(angle_rad) + x0]
                                    |     | yr ← round[(x − x0)·sin(angle_rad) + (y − y0)·cos(angle_rad) + y0]
                                    |     | J_{yr,xr} ← I_{y,x}  if (0 ≤ xr < Nx) ∧ (0 ≤ yr < Ny)
                                    | J
```

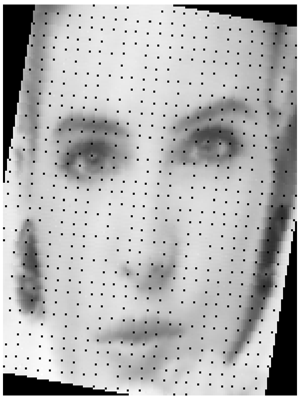
$angle\_deg := 10$ $J := \text{RotateCart}\left(I, angle\_deg, \frac{\text{cols}(I)}{2}, \frac{\text{rows}(I)}{2}\right)$

Image originale



I

Image issue de la rotation en coordonnées cartésiennes



J

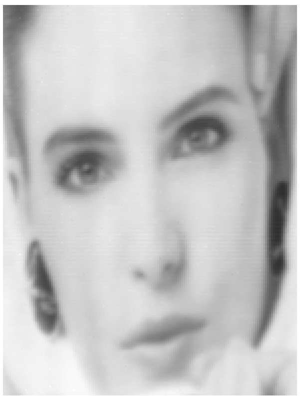
6.2. Rotation en coordonnées polaires (centre de rotation (0,0))

Q1.3. Tansposer sous Python l'algorithme de rotation en coordonnées polaires.
Pourquoi n'y-a-t-il pas de points noirs dans l'image issue de la rotation en coordonnées polaires ?

$$
\mathrm{RotatePol}(I, \mathrm{angle\_deg}) \equiv \left|
\begin{array}{l}
Nx \leftarrow \mathrm{cols}(I) \\
Ny \leftarrow \mathrm{rows}(I) \\
\theta \leftarrow \dfrac{\mathrm{angle\_deg}\cdot\pi}{180} \\
\text{for } y \in 0..Ny-1 \\
\quad \text{for } x \in 0..Nx-1 \\
\quad\quad \left|
\begin{array}{l}
r \leftarrow \sqrt{x^2+y^2} \\
\alpha \leftarrow \mathrm{atan}\left(\dfrac{y}{x}\right) \quad \text{if } x \neq 0 \\
\alpha \leftarrow \dfrac{\pi}{2} \quad \text{otherwise} \\
R \leftarrow r \\
\beta \leftarrow \alpha + \theta \\
X \leftarrow \mathrm{trunc}(R\cdot\cos(\beta)) \\
Y \leftarrow \mathrm{trunc}(R\cdot\sin(\beta)) \\
J_{y,x} \leftarrow I_{Y,X} \quad \text{if } (0 \le X < Nx) \wedge (0 \le Y < Ny)
\end{array}\right. \\
J
\end{array}\right.
$$

$I := \mathrm{READ\_IMAGE}(\text{"coco.bmp"})$ $\mathrm{angle\_deg} := 10$ $J := \mathrm{RotatePol}(I, \mathrm{angle\_deg})$

Image originale



I

Image issue de la rotation
en coordonnées polaires

J

6.3. Zoom en coordonnées polaires (centre de zoom (0,0))

Q1.4. Déduire de l'algorithme de rotation en coordonnées polaires, un algo. de zoom en coordonnées polaires

$$
\mathrm{zoomPol}(I,\mathrm{coeff}) \equiv
\begin{array}{|l}
Nx \leftarrow \mathrm{cols}(I) \\
Ny \leftarrow \mathrm{rows}(I) \\
\text{for } y \in 0..Ny-1 \\
\quad \text{for } x \in 0..Nx-1 \\
\quad\quad \begin{array}{|l}
r \leftarrow \sqrt{x^2+y^2} \\
\alpha \leftarrow \mathrm{atan}\left(\frac{y}{x}\right) \text{ if } x \neq 0 \\
\alpha \leftarrow \frac{\pi}{2} \text{ otherwise} \\
R \leftarrow \blacksquare \\
\beta \leftarrow \blacksquare \\
X \leftarrow \mathrm{trunc}(R\cdot\cos(\beta)) \\
Y \leftarrow \mathrm{trunc}(R\cdot\sin(\beta)) \\
J_{y,x} \leftarrow I_{Y,X} \text{ if } (0 \le X < Nx) \wedge (0 \le Y < Ny)
\end{array} \\
J
\end{array}
$$

<-- le paramètre coeff est le coefficient de zoom, avec la convention :
coeff > 1 pour un zoom avant
coeff < 1 pour un zoom arrière

mais la convention inverse est tout à fait possible :
(coeff < 1 pour un zoom avant,
coeff > 1 pour un zoom arrière).

<-- à compléter (R)
<-- à compléter (β)

J := READ_IMAGE("ZoomPol.bmp")

I := READ_IMAGE("coco.bmp") coeff := 2 J := zoomPol(I, coeff) WRITEBMP("ZoomPol.bmp") := J

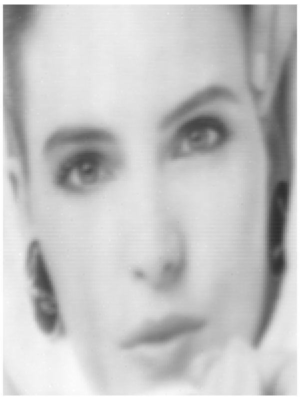


Image originale

I

Image issue du zoom en coordonnées polaires

J

6.4. Zoom en coordonnées cartésiennes (centre de zoom (0,0))

Q1.5. Compléter l'algorithme zoomCart pour réaliser un zoom en coordonnées cartésiennes

$$
\mathrm{zoomCart}(I,\mathrm{coeff}) \equiv
\begin{array}{|l}
Nx \leftarrow \mathrm{cols}(I) \\
Ny \leftarrow \mathrm{rows}(I) \\
\text{for } y \in 0..Ny-1 \\
\quad \text{for } x \in 0..Nx-1 \\
\quad\quad \begin{array}{|l}
X \leftarrow \blacksquare \\
Y \leftarrow \blacksquare \\
J_{y,x} \leftarrow I_{Y,X} \text{ if } (0 \le X < Nx) \wedge (0 \le Y < Ny)
\end{array} \\
J
\end{array}
$$

<-- le paramètre coeff est le coefficient de zoom, avec la convention :
coeff > 1 pour un zoom avant
coeff < 1 pour un zoom arrière

mais la convention inverse est tout à fait possible :
(coeff < 1 pour un zoom avant,
coeff > 1 pour un zoom arrière).

<-- à compléter (X)
<-- à compléter (Y)

$I := READ\_IMAGE("coco.bmp")$ $coeff := 0.7$ $J := zoomCart(I, coeff)$ $J := READ\_IMAGE("ZoomCart.bmp")$

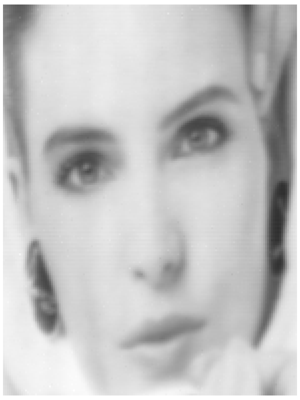


Image originale

I

Image issue du zoom en coordonnées cartésiennes

J

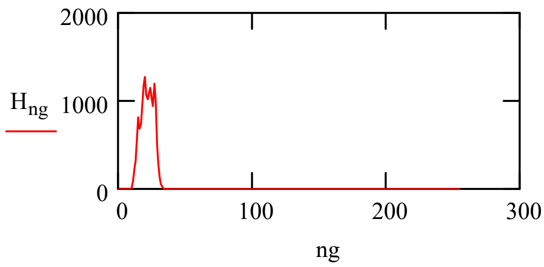
7. Recadrage de dynamique d'une image (sans perte de points aux bords)

Q1.6. Coder sous Python l'algorithme de recadrage de dynamique. Limites de l'algorithme ? Améliorations ?

$I := READ\_IMAGE("aquitaine.bmp")$ Nombre de colonnes : $Nx := cols(I)$ Nombre de lignes : $Ny := rows(I)$

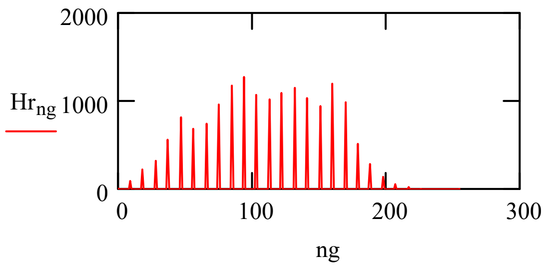
$H := Histo(I)$ $Nx = 128$ $Ny = 128$

$$
\begin{aligned}
RecadrageDyn(I) := \quad & mini \leftarrow min(I) \\
& maxi \leftarrow max(I) \\
& \Delta \leftarrow \frac{255}{maxi - mini} \\
& \text{for } y \in 0..rows(I)-1 \\
& \quad \text{for } x \in 0..cols(I)-1 \\
& \quad\quad J_{y,x} \leftarrow trunc\left[\left(I_{y,x} - mini\right)\cdot\Delta\right] \\
& J
\end{aligned}
$$



Histogramme de l'image brute initiale

$R := RecadrageDyn(I)$ $Hr := Histo(R)$



Histogramme de l'image recadrée dynamiquement

Image originale

I

Image de dynamique recadrée

R