TD – Traitement d'images

1ère partie - Corrigé

Introduction et objectifs

Dans cette première partie du TD consacré au traitement d'images, l'objectif principal est de se familiariser avec les codages des images (nuances de gris/couleurs) et leurs manipulations de base sous Scilab sous forme de matrices.

Quelques transformations élémentaires

Exercice N°1 – Conversion en nuances de gris

Ecrire une fonction Scilab `rvb2nb` recevant une image couleur (`M`) comme argument et créant une nouvelle image `Mnb` en nuances de gris correspondant à `M`.

On utilisera la fonction `imlincomb` de SVIP mais, naturellement, on n'utilisera pas la fonction `rgb2gray` ! ☺

L'Union Internationale des Télécommunications préconise (recommandation 601), pour les images visualisées sur un écran, de calculer le niveau de gris G (luminance) d'un pixel donné à partir de ses composantes RVB (Rouge Vert Bleu) selon la formule :

$$G = 0,299R + 0,587V + 0,114B$$

Votre fonction `rvb2nb` fera donc appel à la fonction `imlincomb` avec les coefficients appropriés.

Sous Scilab, on obtient simplement la fonction suivante :

```
function Mnb=rvb2nb(M)
    // On crée trois nouvelles matrices R, V et B qui
    reçoivent respectivement les composantes rouge, verte et
    bleue de chacun des pixels de la matrice M.
    R=M(:,:,1)
    V=M(:,:,2)
    B=M(:,:,3)
    Mnb=imlincomb(0.299,R,0.587,V,0.114,B)
endfunction
```

Avec l'image du fichier 'BalDesVampires2.jpg', on obtient :

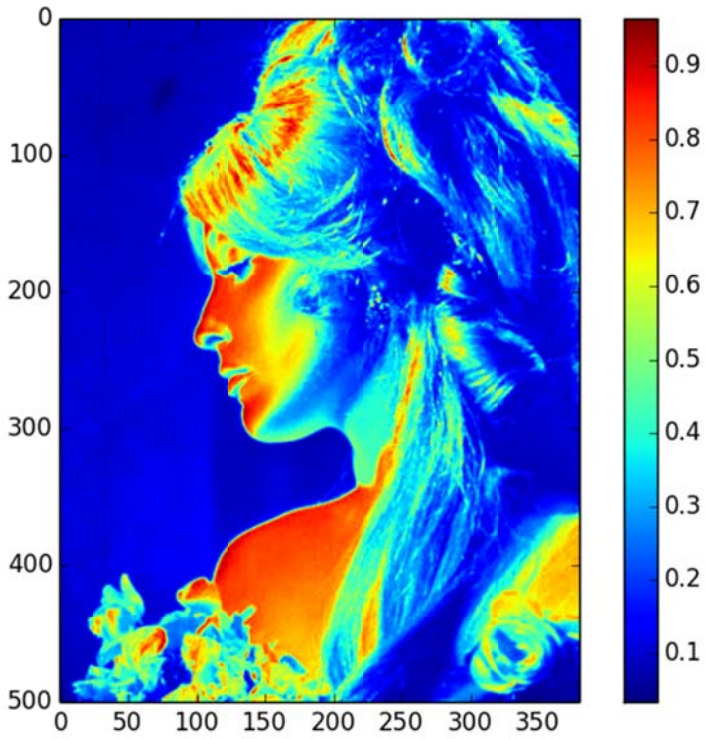

Sous Python, on pourra utiliser le script suivant :

```
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np

def rvb2nb(M):
     return 0.299*M[:,:,0]+0.587*M[:,:,1]+0.114*M[:,:,2]

# Nettoyage de la fenêtre d'affichage
plt.clf()

# Lecture du fichier image.
M = img.imread('BalDesVampires1.png')

# Rescaling pour obtenir une image en nuances de gris.
Mnb = rvb2nb(M)

# Affichage du nouveau tableau en tant qu'image.
M2 = plt.imshow(Mnb)

# Changement de la palette graphique pour obtenir une image en
nuances de gris.
M2.set_cmap('gray')

# Décommentez la dernière ligne ci-dessous si vous souhaitez
faire apparaître # l'échelle des nuances de gris.
#plt.colorbar(orientation="horizontal")
```

Rappelons qu'avec Python (et les bibliothèques idoines), les images sont manipulées sous forme de tableaux `numpy` dont les éléments sont des flottants compris entre 0 et 1. Ainsi, la combinaison linéaire requise pour convertir les trois indices de couleur en un seul indice de luminance s'écrit simplement, sans aucun problème de conversion :

```
0.299*M[:,:,0]+0.587*M[:,:,1]+0.114*M[:,:,2]
```

Par défaut cependant, la « `color map` » (palette graphique) permettant d'afficher des images dont les pixels sont codés sur un octet est une `color map` utilisant des couleurs chaudes pour les zones claires et froides pour les zones foncées. Ainsi, en n'utilisant pas la `colormap` '`gray`', on obtient :



Exercice N°2 – Négatif d'une image

Comment obtenir simplement le négatif d'une image (représentée matriciellement par `M`) en nuances de gris à l'aide de la seule fonction `imshow` ?

Rappels : pour les images en nuances de gris, les pixels sont codés, sous la forme d'entiers non signés (`uint8`) et les calculs sur de tels entiers sont ceux de $\mathbb{Z}/256\mathbb{Z}$ (arithmétique modulo 256).

Le négatif d'une image sera obtenu en remplaçant chaque nuance de gris (sou SIVP, rappelons qu'il s'agit d'un entier n non signé codé sur 8 bits, donc de type `uint8`) par son « complément à 255 », c'est-à-dire $255-n$. Matriciellement, le calcul s'écrit : $255U-M$ où U est une matrice ayant la dimension de la matrice image M mais ne comportant que des 1. Sous Scilab, ce calcul s'écrit simplement : `255-M` et comme les calculs s'effectuent modulo 256, on peut simplement écrire `-1-M`. D'où, finalement, la commande :

```
imshow(-1-M)
```

Avec Python, on pourra simplement considérer la fonction :

```
def neg(M):
    return 1.0-M
```

Après avoir transformé l'image du fichier 'Battersea1.jpg' en image en nuances de gris, on applique à cette nouvelle image la commande précédente et on obtient :

La Battersea en nuances de gris

Le négatif de l'image précédente.

Exercice N°3 – Floutage

Dans un premier temps, on appliquera à une image donnée (M) un filtre correspondant à un « floutage uniforme » (« uniform blur ») dont le noyau est une matrice *K* dont tous les coefficients sont égaux $k_{ij} = cte$).

Dans un second temps, on appliquera à une image donnée (M) un filtre correspondant à un « floutage gaussien » (« gaussian blur ») dont le noyau non normalisé est une matrice *K* dont tous les coefficients sont définis par $k_{ij} = \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right)$.

Remarque : la convention d'indexation des coefficients des noyaux est celle vue en cours.

On utilisera la fonction `fspecial` pour définir ces filtres puis la fonction `imfilter` pour les appliquer.

Pour chacun des deux filtres étudiés :

- On précisera les arguments possibles de la fonction `fspecial` et on testera l'effet du filtre pour différentes valeurs de ces arguments (pour les noyaux, on se limitera à des matrices carrées).
- On justifiera les valeurs des coefficients des noyaux normalisés.

Cas du floutage uniforme

Pour un noyau non normalisé correspondant à une matrice carrée d'ordre *n*, on a n^2 coefficients tous égaux à 1. Le coefficient de normalisation vaut donc simplement n^2.
Pour $n = 3$ et $n = 5$ les noyaux normalisés s'écrivent donc respectivement :

$$K_3 = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \text{ et } K_5 = \frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Dans la ligne de commande Scilab, on écrira respectivement
`K3=fspecial('average',3)` (le « 3 » peut être omis, il s'agit de la valeur par défaut) et `K5=fspecial('average',5)`.

Pour appliquer ces filtres, on utilise la fonction `imfilter`.

Par exemple si on a placé l'image du fichier BalDesVampires1.jpg dans la matrice M1, on pourra écrire : `M2=imfilter(M1,K3)` (cf. le résultat ci-dessous).

Avant le floutage…

Après le floutage.

On peut noter la perte de netteté générale. On pourra y voir un défaut ou… une qualité.
Ce type de filtrage peut cependant s'avérer utile pour « nettoyer » certaines images.

Avec le noyau K5, l'effet est encore plus prononcé…

Avant le floutage…

Après le floutage.

Cas du floutage gaussien

Par défaut, on a $\sigma = \frac{1}{2}$. Le coefficient du noyau non normalisé est donc égal à

$$k_{ij} = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) = \exp\left(-\frac{i^2+j^2}{2\times\frac{1}{4}}\right) = \exp\left(-2\left(i^2+j^2\right)\right).$$

Pour $n=3$, on obtient :

$$\begin{pmatrix} \exp\left(-2\left((-1)^2+(-1)^2\right)\right) & \exp\left(-2\left(0^2+(-1)^2\right)\right) & \exp\left(-2\left(1^2+(-1)^2\right)\right) \\ \exp\left(-2\left((-1)^2+0^2\right)\right) & \exp\left(-2\left(0^2+0^2\right)\right) & \exp\left(-2\left(1^2+0^2\right)\right) \\ \exp\left(-2\left((-1)^2+1^2\right)\right) & \exp\left(-2\left(0^2+1^2\right)\right) & \exp\left(-2\left(1^2+1^2\right)\right) \end{pmatrix} = \begin{pmatrix} e^{-4} & e^{-2} & e^{-4} \\ e^{-2} & 1 & e^{-2} \\ e^{-4} & e^{-2} & e^{-4} \end{pmatrix}$$

Le coefficient de normalisation *m* vaut donc : $m = 1+4\times e^{-2}+4\times e^{-4}$ et finalement le noyau normalisé K_3 s'écrit :

$$K_3 = \frac{1}{1+4\times e^{-2}+4\times e^{-4}}\begin{pmatrix} e^{-4} & e^{-2} & e^{-4} \\ e^{-2} & 1 & e^{-2} \\ e^{-4} & e^{-2} & e^{-4} \end{pmatrix} \simeq \begin{pmatrix} 0,011343737 & 0,083819506 & 0,011343737 \\ 0,083819506 & 0,619347031 & 0,083819506 \\ 0,011343737 & 0,083819506 & 0,011343737 \end{pmatrix}$$

Dans la ligne de commande Scilab, on écrira par exemple :

```
K3=fspecial('gaussian')
```

Pour $n=5$, on obtient :

$$\begin{pmatrix} \exp\left(-2\left((-2)^2+(-2)^2\right)\right) & \exp\left(-2\left((-1)^2+(-2)^2\right)\right) & \exp\left(-2\left(0^2+(-2)^2\right)\right) & \exp\left(-2\left(1^2+(-2)^2\right)\right) & \exp\left(-2\left(2^2+(-2)^2\right)\right) \\ \exp\left(-2\left((-2)^2+(-1)^2\right)\right) & \exp\left(-2\left((-1)^2+(-1)^2\right)\right) & \exp\left(-2\left(0^2+(-1)^2\right)\right) & \exp\left(-2\left(1^2+(-1)^2\right)\right) & \exp\left(-2\left(2^2+(-1)^2\right)\right) \\ \exp\left(-2\left((-2)^2+0^2\right)\right) & \exp\left(-2\left((-1)^2+0^2\right)\right) & \exp\left(-2\left(0^2+0^2\right)\right) & \exp\left(-2\left(1^2+0^2\right)\right) & \exp\left(-2\left(2^2+0^2\right)\right) \\ \exp\left(-2\left((-2)^2+1^2\right)\right) & \exp\left(-2\left((-1)^2+1^2\right)\right) & \exp\left(-2\left(0^2+1^2\right)\right) & \exp\left(-2\left(1^2+1^2\right)\right) & \exp\left(-2\left(2^2+1^2\right)\right) \\ \exp\left(-2\left((-2)^2+2^2\right)\right) & \exp\left(-2\left((-1)^2+2^2\right)\right) & \exp\left(-2\left(0^2+2^2\right)\right) & \exp\left(-2\left(1^2+2^2\right)\right) & \exp\left(-2\left(2^2+2^2\right)\right) \end{pmatrix}$$

$$= \begin{pmatrix} e^{-16} & e^{-10} & e^{-8} & e^{-10} & e^{-16} \\ e^{-10} & e^{-4} & e^{-2} & e^{-4} & e^{-10} \\ e^{-8} & e^{-2} & 1 & e^{-2} & e^{-8} \\ e^{-10} & e^{-4} & e^{-2} & e^{-4} & e^{-10} \\ e^{-16} & e^{-10} & e^{-8} & e^{-10} & e^{-16} \end{pmatrix}$$

Le coefficient de normalisation *m* vaut : $m = 1 + 4\times e^{-2} + 4\times e^{-4} + 4\times e^{-8} + 8\times e^{-10} + 4\times e^{-16}$ et, finalement, le noyau normalisé K_3 s'écrit :

$$K_5 = \frac{1}{1+4\times e^{-2}+4\times e^{-4}+4\times e^{-8}+8\times e^{-10}+4\times e^{-16}}\begin{pmatrix} e^{-16} & e^{-10} & e^{-8} & e^{-10} & e^{-16} \\ e^{-10} & e^{-4} & e^{-2} & e^{-4} & e^{-10} \\ e^{-8} & e^{-2} & 1 & e^{-2} & e^{-8} \\ e^{-10} & e^{-4} & e^{-2} & e^{-4} & e^{-10} \\ e^{-16} & e^{-10} & e^{-8} & e^{-10} & e^{-16} \end{pmatrix}$$

$$\simeq \begin{pmatrix} 0,000000070 & 0,000028089 & 0,000207549 & 0,000028089 & 0,000000070 \\ 0,000028089 & 0,01133176685 & 0,083731061 & 0,01133176685 & 0,000028089 \\ 0,000207549 & 0,083731061 & 0,618693507 & 0,083731061 & 0,000207549 \\ 0,000028089 & 0,01133176685 & 0,083731061 & 0,01133176685 & 0,000028089 \\ 0,000000070 & 0,000028089 & 0,000207549 & 0,000028089 & 0,000000070 \end{pmatrix}$$

Dans la ligne de commande Scilab, on écrira par exemple :

```
K5=fspecial('gaussian',5)
```

On note cette fois la présence d'un deuxième argument qui, lorsqu'il s'agit, comme ici, d'un nombre, indique l'ordre de la matrice carrée correspondant au noyau. Si le noyau n'est pas une matrice carrée, on peut fournir un vecteur de dimension 1×2 contenant le nombre de ligne et le nombre de colonnes du noyau.

Dans tous les appels à la fonction `fspecial`, on peut également fournir un dernier argument. Comme mentionné plus haut, il s'agit de la valeur de σ. Rappelons que l'on a :

$$k_{ij} = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

Ainsi, L'atténuation « loin » du pixel central sera d'autant plus forte que la valeur de σ sera plus faible. En d'autres termes, l'effet du floutage sera d'autant plus réduit que σ sera faible. A contrario, pour une valeur élevée de σ, la décroissance de l'exponentielle sera moindre et les effets du floutage seront beaucoup plus important.

Commençons par donner quelques noyaux pour diverses valeurs de σ.

$$K_3(0,2) \simeq \begin{pmatrix} 1,389\times 10^{-11} & 0,0000037 & 1,389\times 10^{-11} \\ 0,0000037 & 0,9999851 & 0,0000037 \\ 1,389\times 10^{-11} & 0,0000037 & 1,389\times 10^{-11} \end{pmatrix}$$

Pour $\sigma = 0,2$ on constate que la valeur central du noyau est très proche de 1 puis que les valeurs des autres coefficients « diminuent très vite avec la distance au centre » (les valeurs aux « sommets » peuvent être considérées comme nulles).

$$K_3(0,35) \simeq \begin{pmatrix} 0,0002666 & 0,0157954 & 0,0002666 \\ 0,0157954 & 0,935752 & 0,0157954 \\ 0,0002666 & 0,0157954 & 0,0002666 \end{pmatrix}$$

$$K_3(0,5) \simeq \begin{pmatrix} 0,0113437 & 0,0838195 & 0,0113437 \\ 0,0838195 & 0,6193470 & 0,0838195 \\ 0,0113437 & 0,0838195 & 0,0113437 \end{pmatrix}$$

En faisant croître la valeur de σ, on augmente les valeurs des coefficients « éloignés » du coefficient central : leur effet sur le résultat du filtrage en est augmenté. Continuons un peu :

$$K_3(0,65) \simeq \begin{pmatrix} 0,0360670 & 0,1177792 & 0,0360670 \\ 0,1177792 & 0,3846152 & 0,1177792 \\ 0,0360670 & 0,1177792 & 0,0360670 \end{pmatrix}$$

$$K_3(0,8) \simeq \begin{pmatrix} 0,0571183 & 0,1247577 & 0,0571183 \\ 0,1247577 & 0,2724960 & 0,1247577 \\ 0,0571183 & 0,1247577 & 0,0571183 \end{pmatrix}$$

Nous utilisons ce dernier noyau sur l'image en nuances de gris obtenue à partir du fichier BalDesVampires1.jpg (voir page suivante). On note que les effets obtenus sont assez doux. Remarquons également que pour des valeurs beaucoup plus élevées de σ, on a :

$$k_{ij} = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) \simeq 1$$

Ainsi, pour des valeurs élevées de σ, l'effet du flou gaussien sera comparable à celui d'un flou uniforme.

Image originale (après transformation en nuances de gris).

Flou gaussien (noyau 3×3 et $\sigma = 0,8$).

Exercice N°4 – Changement de contraste

On se donne une image en nuances de gris.

Rappelons que, d'un point de vue qualitatif :

- Augmenter le contraste de l'image revient à rendre plus foncés les gris foncés et plus clairs les gris clairs.
- Réduire le contraste de l'image revient à rendre moins foncés les gris foncés et moins clairs les gris clairs.

Ainsi, modifier le contraste revient à appliquer une fonction judicieusement choisie à la nuance de gris de chaque pixel. On pourrait ainsi d'emblée considérer une fonction de $[\![0\,;255]\!]$ dans lui-même mais on utilisera en fait une fonction de $[0\,;1]$ dans lui-même et les calculs seront menés en double précision.

On demande ici d'écrire et tester, à l'aide des fonctions tangente et arctangente, deux fonctions permettant de diminuer ou augmenter le contraste d'une image donnée (`M`).

Comme le suggère l'énoncé, on se ramène classiquement à l'intervalle $[0\,;1]$.

Nous traitons en détails le cas de l'augmentation du contraste à l'aide de la fonction arctangente.

Comme on l'a vu en cours, nous allons construire une fonction φ telle que l'allure de sa courbe représentative soit celle d'un « S ». Pour simplifier cette construction, nous imposons les « contraintes » (symétrie) : $\varphi\left(\frac{1}{2}\right)=\frac{1}{2}$, $\varphi(0)=0$ et $\varphi(1)=1$.

On part donc de la fonction : $x\mapsto \arctan(x)$.

Afin de pouvoir jouer sur la forme du « S », on introduit un paramètre α : $x\mapsto \arctan(\alpha x)$.

Pour que notre fonction vérifie $\varphi\left(\frac{1}{2}\right)=\frac{1}{2}$ (graphiquement parlant on effectue une translation de la courbe représentative de la fonction arctan de vecteur $\frac{1}{2}\left(\vec{i}+\vec{j}\right)$), on considère la fonction : $x\mapsto \frac{1}{2}+\arctan\left(\alpha\left(x-\frac{1}{2}\right)\right)$.

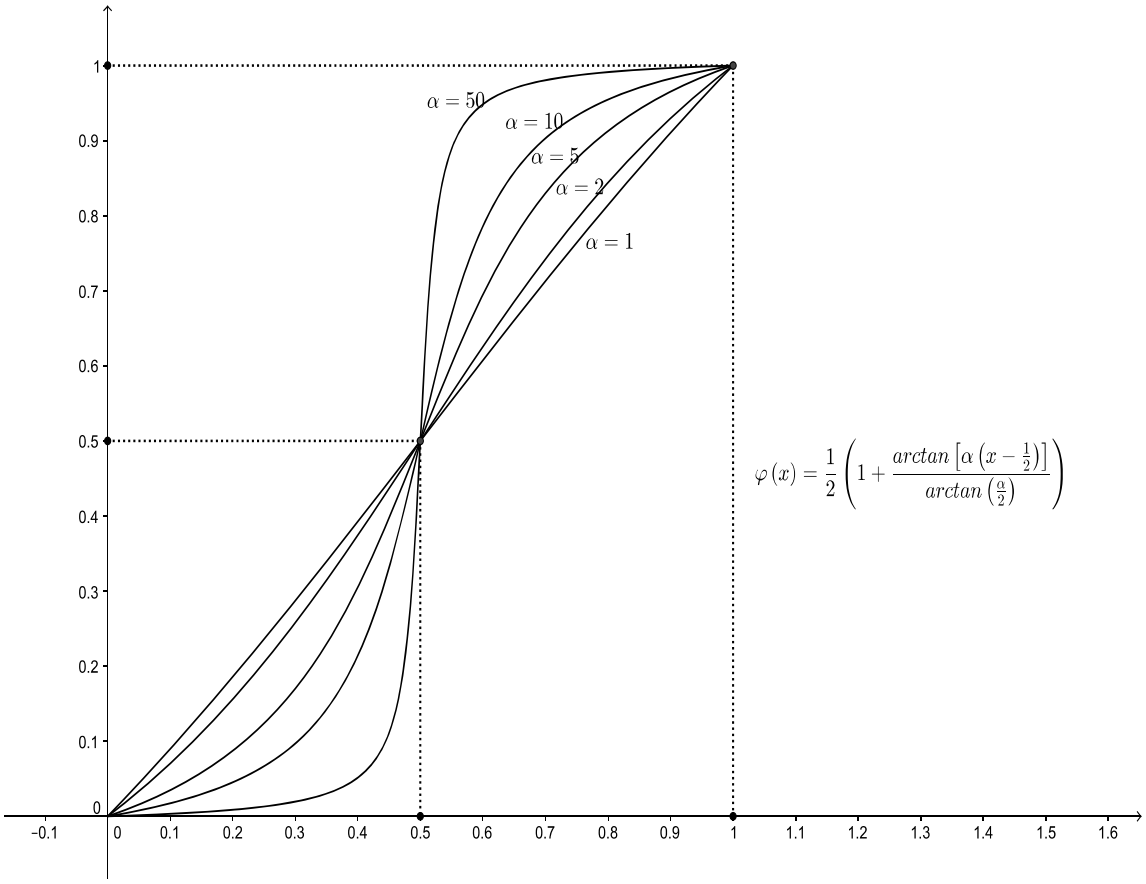
Pour $x=0$ et $x=1$, une telle fonction fournira les valeurs :

$$\frac{1}{2}+\arctan\left(\alpha\left(-\frac{1}{2}\right)\right)=\frac{1}{2}-\arctan\left(\frac{\alpha}{2}\right) \text{ et } \frac{1}{2}+\arctan\left(\frac{\alpha}{2}\right)$$

Comme nous voulons en fait les valeurs 0 et 1, il suffit de « normaliser » en divisant le second terme par $2\arctan\left(\frac{\alpha}{2}\right)$. On obtient finalement :

$$\varphi_\alpha : x \mapsto \frac{1}{2} + \frac{\arctan\left(\alpha\left(x-\frac{1}{2}\right)\right)}{2\arctan\left(\frac{\alpha}{2}\right)} = \frac{1}{2}\left[1 + \frac{\arctan\left(\alpha\left(x-\frac{1}{2}\right)\right)}{\arctan\left(\frac{\alpha}{2}\right)}\right]$$

Nous fournissons ci-après les courbes représentatives de fonctions φ_α pour diverses valeurs du paramètre α :



Clairement, l'augmentation du contraste sera d'autant plus marquée que la valeur du paramètre α sera grande.

En effet, on a facilement :

$$\varphi'_\alpha : x \mapsto \frac{1}{2\arctan\left(\frac{\alpha}{2}\right)} \times \alpha \times \frac{1}{1+\left[\alpha\left(x-\frac{1}{2}\right)\right]^2} = \frac{\alpha}{2\arctan\left(\frac{\alpha}{2}\right) \times \left[1+\alpha^2\left(x-\frac{1}{2}\right)^2\right]}$$

Puis :

$$\varphi''_\alpha : x \mapsto \frac{\alpha}{2\arctan\left(\frac{\alpha}{2}\right)} \times \frac{-2\alpha^2\left(x-\frac{1}{2}\right)}{\left[1+\alpha^2\left(x-\frac{1}{2}\right)^2\right]^2} = \frac{-2\alpha^3\left(x-\frac{1}{2}\right)}{2\arctan\left(\frac{\alpha}{2}\right)\times\left[1+\alpha^2\left(x-\frac{1}{2}\right)^2\right]^2}$$

On a donc : $\varphi'_\alpha\left(\frac{1}{2}\right) = \frac{\alpha}{2\arctan\left(\frac{\alpha}{2}\right)}$.

On a facilement : $\arctan\left(\frac{\alpha}{2}\right) \underset{\infty}{\sim} \frac{\pi}{2}$ et donc : $\varphi'_\alpha\left(\frac{1}{2}\right) \underset{\infty}{\sim} \frac{\alpha}{\pi}$.

Les variations de φ_α au voisinage de $\frac{1}{2}$ seront donc d'autant plus forte que α sera grand.

De plus $\varphi''_\alpha\left(\frac{1}{2}\right) = 0$. Ainsi, les variations de la dérivée φ'_α au voisinage de $\frac{1}{2}$, restent très faibles, même si la dérivée elle-même peut prendre de fortes valeurs.

Sous Scilab, le codage de la fonction `contraste` peut alors être :

```
function y=f(x)
      alpha=10
      y=0.5*(1+atan(alpha*(x-0.5))/atan(alpha/2))
endfunction

function MC=contraste(M,f)
      MC=uint8(f(double(M)/255.)*255)
endfunction
```

Remarques :

- On a séparé les définitions des fonctions `f` et `contraste` pour uniquement modifier la définition de `f` lorsque l'on souhaite changer celle-ci.
- Dans la définition de `f`, le paramètre `alpha` apparaît sous la forme d'une variable. On pourrait imaginer que cette valeur soit, par exemple, saisie par l'utilisateur ou corresponde à la position d'un curseur (sur une interface graphique).
- Les contraintes (de symétrie) imposées à la fonction `f` ne sont en rien une obligation. Avec des logiciels comme Gimp (www.gimp.org) ou Photoshop, par exemple, on a beaucoup de liberté pour définir/préciser la fonction φ.

Pour terminer, nous illustrons l'effet de la fonction `contraste` en l'appliquant à l'image BalDesVampires1.jpg avec, successivement, $\alpha = 2$ puis $\alpha = 4$ (voir page suivante).

Image d'origine en nuances de gris.

Augmentation du contraste ($\alpha = 2$).

Augmentation du contraste ($\alpha = 4$).

Sous Python, on pourra utiliser la fonction :

```
def contrast_inc(M,alpha):
     return 0.5*(1+arctan(alpha*(M-0.5))/arctan(alpha*0.5))
```

On notera que nous avons utilisé la fonction `arctan` de la bibliothèque `numpy` car la fonction `atan` du module `math` de Python n'est pas utilisable avec les tableaux `numpy`. En revanche, les fonctions « tangente » de `math` et `numpy` s'appellent toutes deux via `tan`.