Traitement d'images

Prof: EL HADIQ Zouhair

Centre CPGE Moulay Youssef Rabat
Filières : MP et PSI

elhadiq.pro@gmail.com

January 16, 2023

Plan

- Représentation d'images.
- Inversion diimage.
- Passage en niveaux de gris.
- Rotation, réduction ou agrandissement de l'image.
- Calcul d'histogramme, seuillage fixe.
- Flou et Détection de contours.

Le pixel :
Une image numérique est constituée d'un ensemble de points appelés pixels (abréviation de PICture Element) pour former une image.
Le pixel
représente ainsi le plus petit élément constitutif d'une image numérique.
L'ensemble de ces pixels
est contenu dans un tableau à deux dimensions constituant l'image :

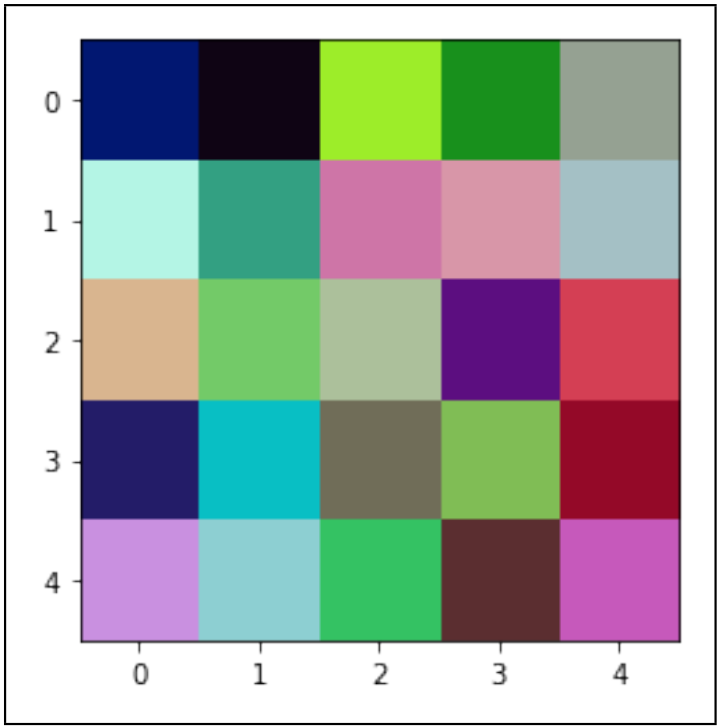


Figure: Exemple d'une image aléatoire

Caractéristiques d'une image: Image en Gris

Les images dites en "noir et blanc" sont représentées par des matrices : chaque élément de la matrice correspondra comme précédemment à un pixel, mais ces éléments seront des nombres entiers donnant l'intensité de gris voulue.
On utilise habituellement des nombres de 0 à 255, 0 pour le noir et 255 pour le blanc, soit $256 = 2^8$ niveaux de gris, ce qui permet de coder un pixel par 8 bits.

Figure: Image de Youssef en-nesyri en gris

Caractéristiques d'une image: Image Coulours

Les
images en couleur sont représentées par trois matrices de taille identique, donnant l'intensité de Rouge, Vert, et Bleu pour chaque pixel. Cette méthode est désignée par le sigle RGB (Red, Green, Blue). Les éléments de ces trois matrices sont encore des entiers de 0 à 255.

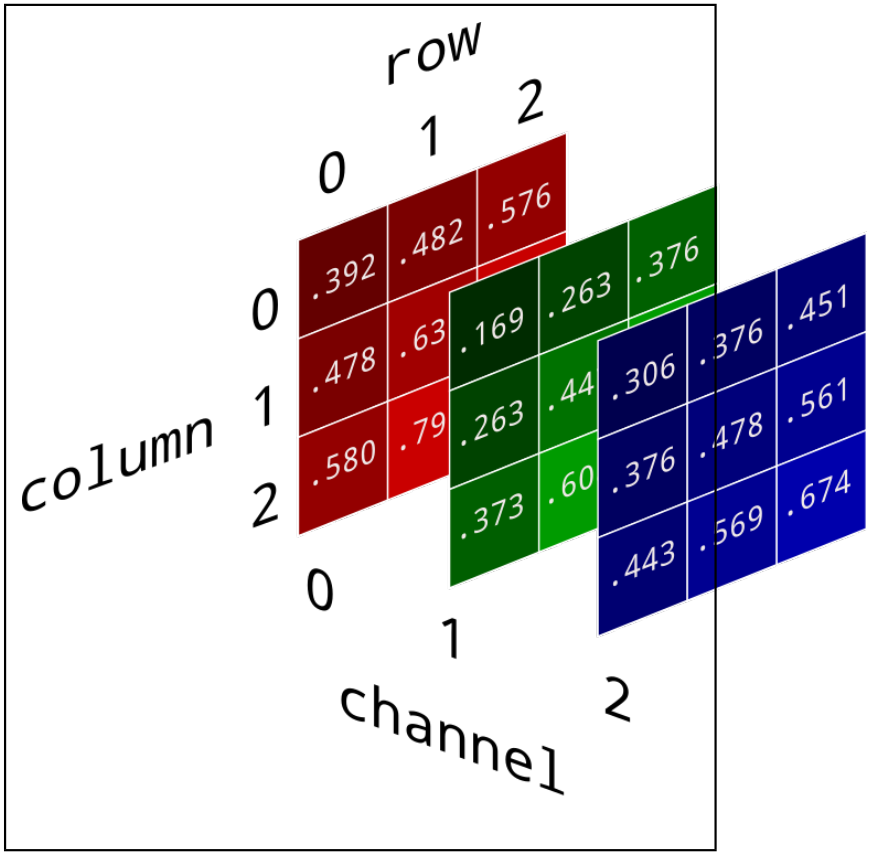
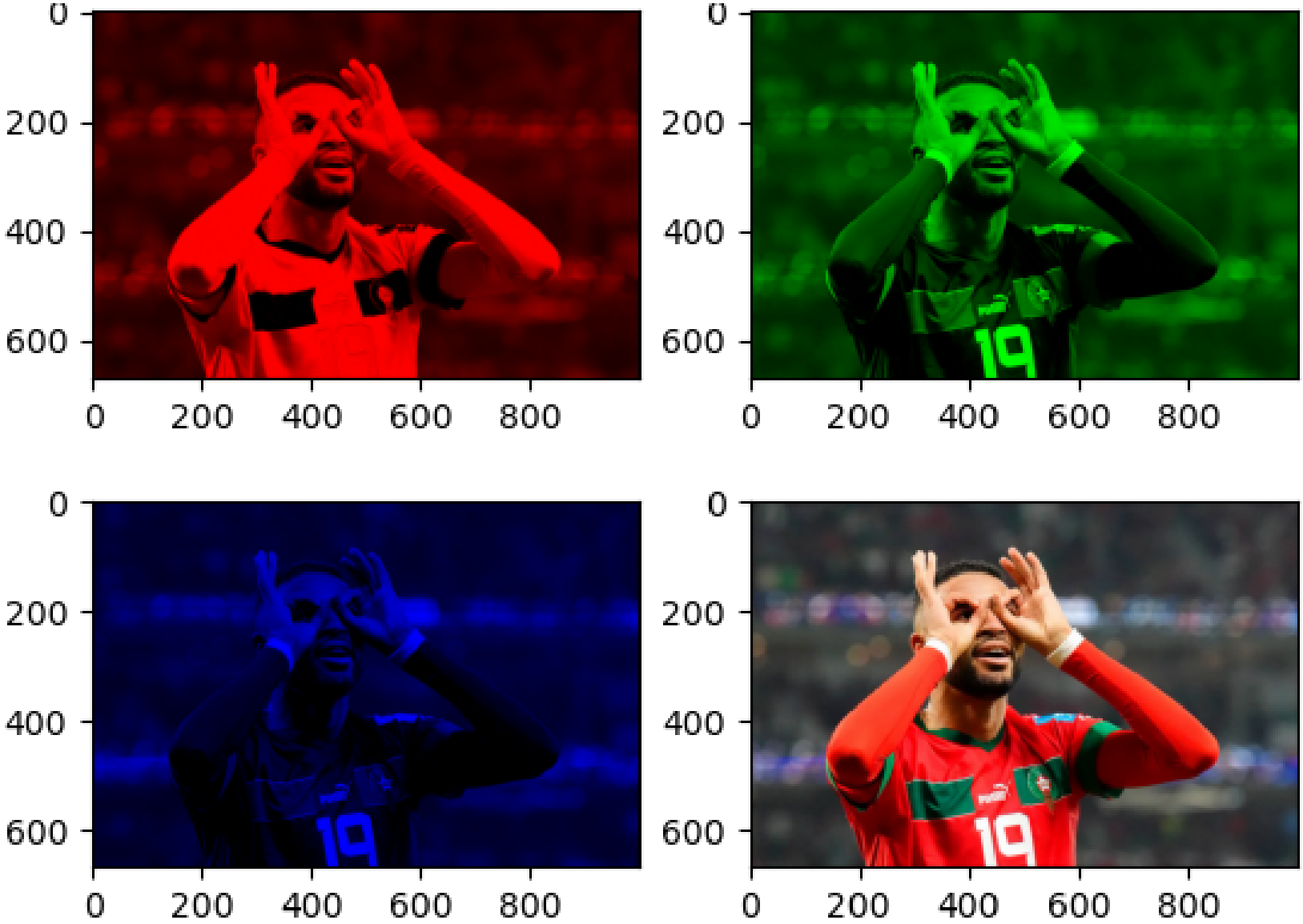


Figure: Exemple d'une image

Caractéristiques d'une image: Image en Coulours

L'image finale est obtenue en combinant les trois images.
Ci-dessous le résultat final.



Chargement de l'image

Nous allons utiliser le module pyplot de matplotlib pour charger en mémoire une image.
On peut procéder de la sorte pour charger une image en-nesyri.png qui se trouverait en répertoire local:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 ######
4 img = plt.imread('en-nesyri.png')
```

La taille de l'image

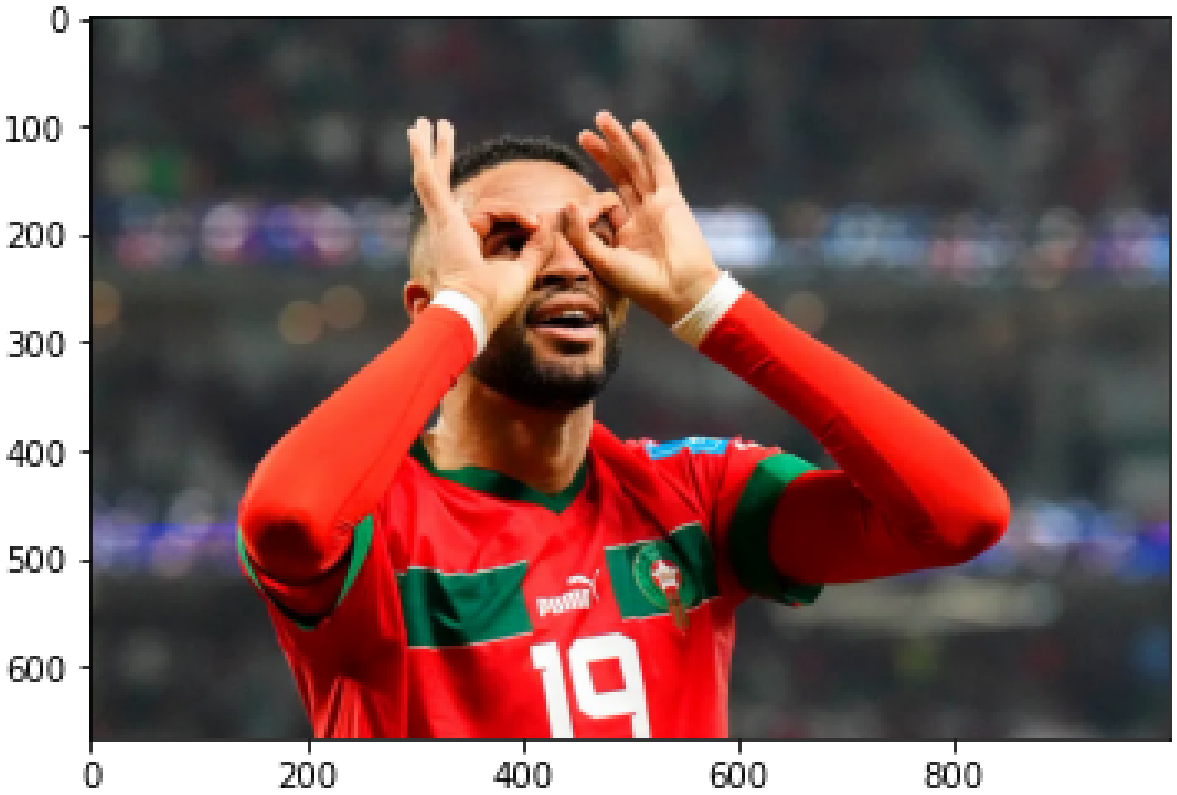
Si, dans un algorithme, il est besoin de connaître la taille de l'image, on rappelle que image.shape fournit un tuple de trois éléments, conprenant la hauteur de l'image, sa largeur, et le nombre de canaux de couleur (3, en principe).

```
1 nblignes,nbcolones,canals=img.shape
```

Affichage d'une image

Pour afficher une image ayant trois canaux de couleur, il suffit de faire appel à plt.imshow:

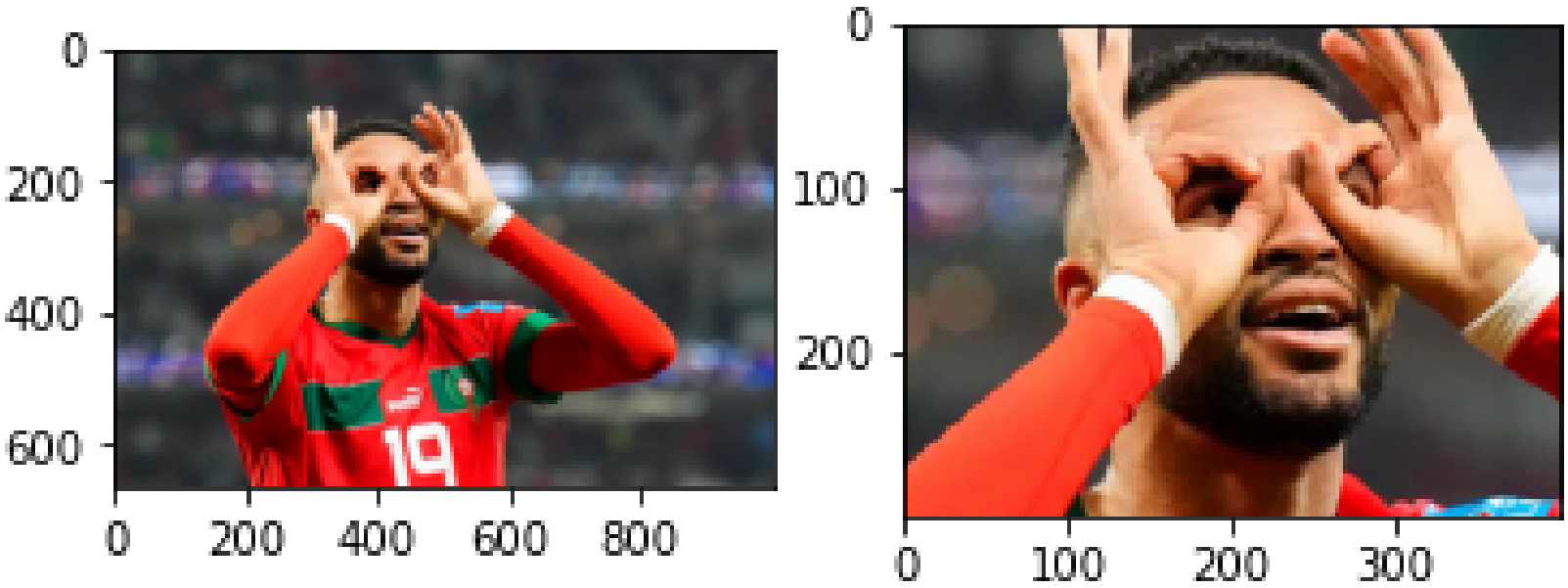
```
1 from matplotlib import pyplot as plt
2 plt.imshow(img)
```



Recadrer l'image

Pour selectionner une partie spécifique en peut utiliser le slicing offert par les objets ndarray de numpy, ainsi ce code permet par exemple de selectionner la partie visage du joeur Youssef En-nesyri.

```
1 plt.imshow(img[100:400,200:600])
```



Modification d'une image

On peut modifier la valeur correspondant à l'intensité d'une couleur d'un pixel par mutation de la case correspondante du tableau, par exemple de la sorte :

```
1 img[4, 11, 2] = 0.2
```

Cette commande attribue, pour le pixel situé sur la ligne d'index 4 (la cinquième ligne), dans la colonne d'index 11, une intensité égale à 0.2 pour le canal d'index 2 (le troisième canal de couleur, donc, correspondant au bleu).

Traitements basiques de l'image: Inversion

L'inversion d'une image consiste à remplacer, pour chaque pixel et pour chaque canal de couleur, une valeur v par $1.0 - v$ (ou $255 - v$ si l'on travaillait avec des valeurs entières). Cela a pour effet de transformer le blanc en noir (et inversement), le rouge en cyan, le vert en magenta, etc.

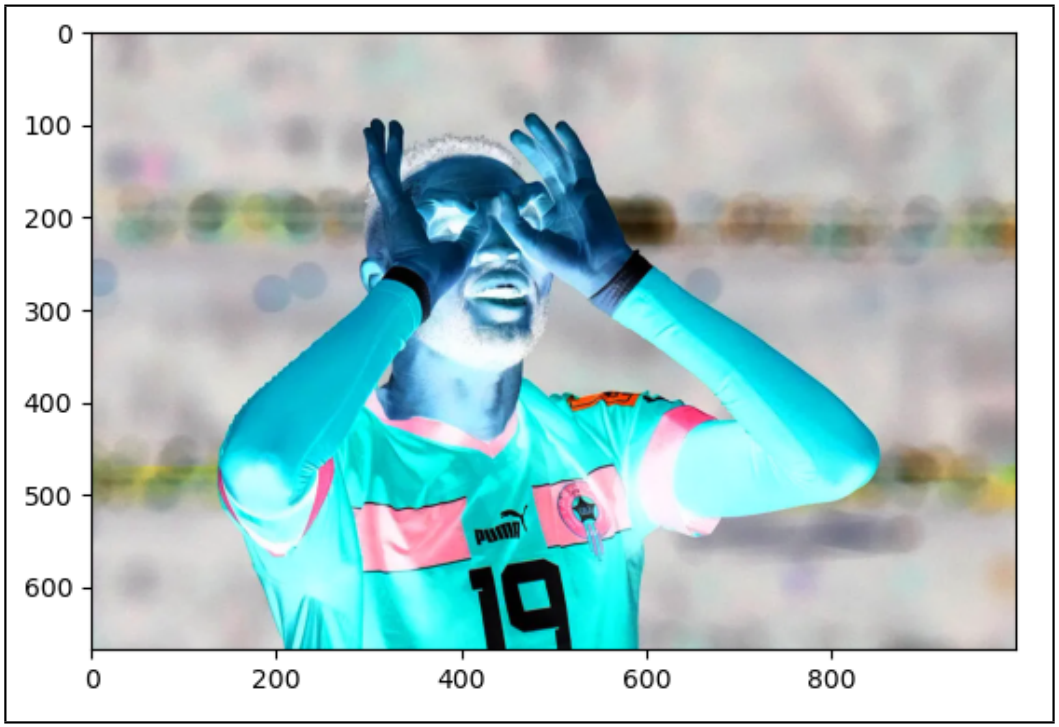


Figure: L'image inversée(Négatif) de Youssef en-nesyri

Traitements basiques de l'image: Inversion

Exercice

Écrire une fonction **Inverse(img)** qui prend en argument une image et l'inverse, et tester son bon fonctionnement.

Traitements basiques de l'image: Inversion

Exercice

Écrire une fonction **Inverse(img)** qui prend en argument une image et l'inverse, et tester son bon fonctionnement.

Réponse

```
1 def inversion1(image):
2     return 1-image
3
4 def inversion2(image):
5     return np.ones_like(image)-image
```

Traitements basiques de l'image: Ajouter un Cadre

Exercice

Écrire
une fonction AjouteCadre(img)
qui prend en argument une image,
et modifie cette image de façon à
ajouter sur celle-ci un Cadre blue.

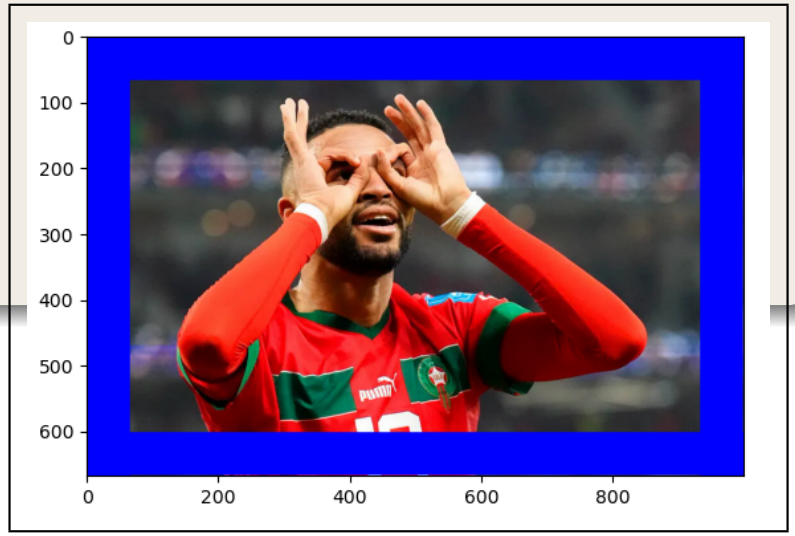
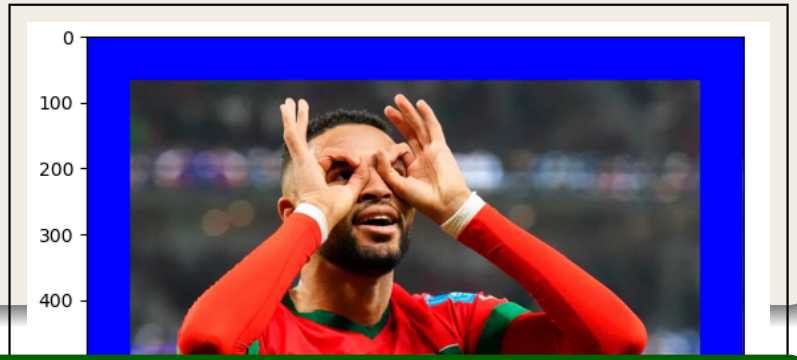


Figure: L'ajout d'un cadre à l'image de Youssef en-nesyri

Traitements basiques de l'image: Ajouter un Cadre

Exercice

Écrire
une fonction AjouteCadre(img)
qui prend en argument une image,
et modifie cette image de façon à
ajouter sur celle-ci un Cadre blue.



Réponse

```
1 def AjouteCadre(image):
2     n=len(image)
3     e=int(0.1*n)
4     blue=np.array([0,0,1])
5     image[0:e,:]=blue
6     image[-e:-1:]=blue
7     image[:,0:e]=blue
8     image[:,-e:-1]=blue
```

Traitements basiques de l'image: Ajouter cercle

Exercice: Ajouter cercle

Ecrire une fonction AjouteDisque (img) qui prend en argument une image, et modifie cette image de façon a ajouter sur celle-ci un cercle bleu de rayon est la moitie de minimaum entre le nombre des lignes et le nombre des colones dont le centre se trouve au centre de l'image, puis tester son bon fonctionnement

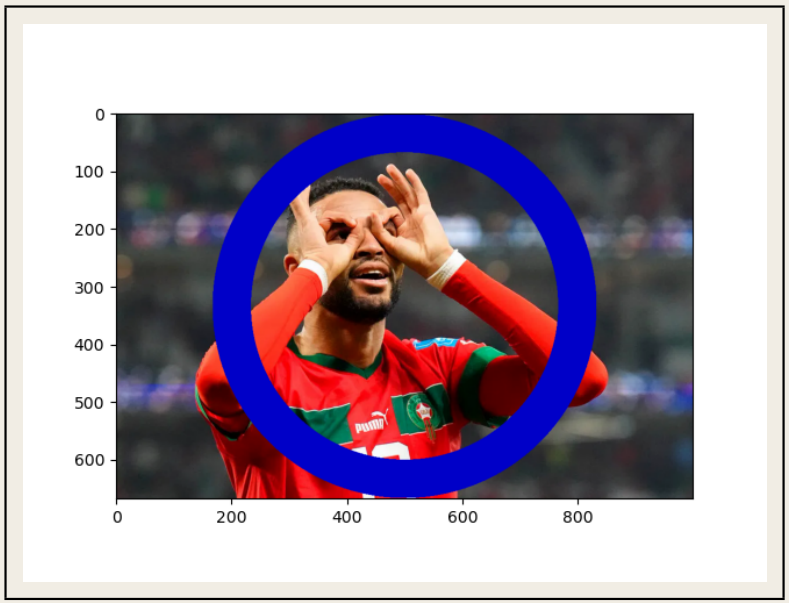


Figure: L'ajout d'une cercle à l'image de Youssef en-nesyri

Ajouter cercle

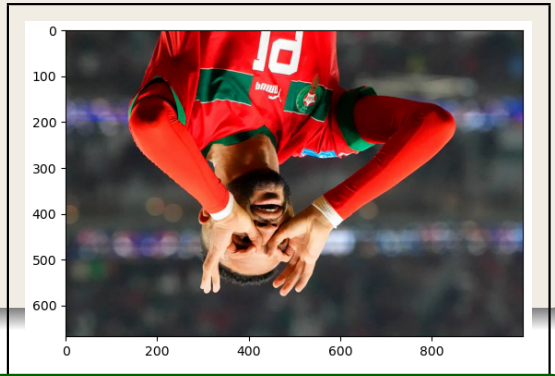
Réponse

```
1 def norm2(i,j):
2     return np.sqrt(i**2+j**2)
3 def cercle(image,ep=1):
4     mx,my=len(image)//2,len(image[0])//2
5     rayon=min(mx,my)
6     epaiseur=ep*rayon
7     blue=np.array([0,0,1])
8     masque=[[norm2(i-mx,j-my)<rayon and
    norm2(i-mx,j-my)>rayon-epaiseur for j in
    range(len(image[0]))]for i in range(len(
    image))]
9     image[masque]=blue
```

Traitements basiques de l'image: Retournement

Définition

Le retournement d'une image consiste à effectuer une symétrie autour de la ligne médiane de l'image. Les pixels de la dernière ligne se retrouvent sur la première ligne (et inversement), et ainsi de suite.



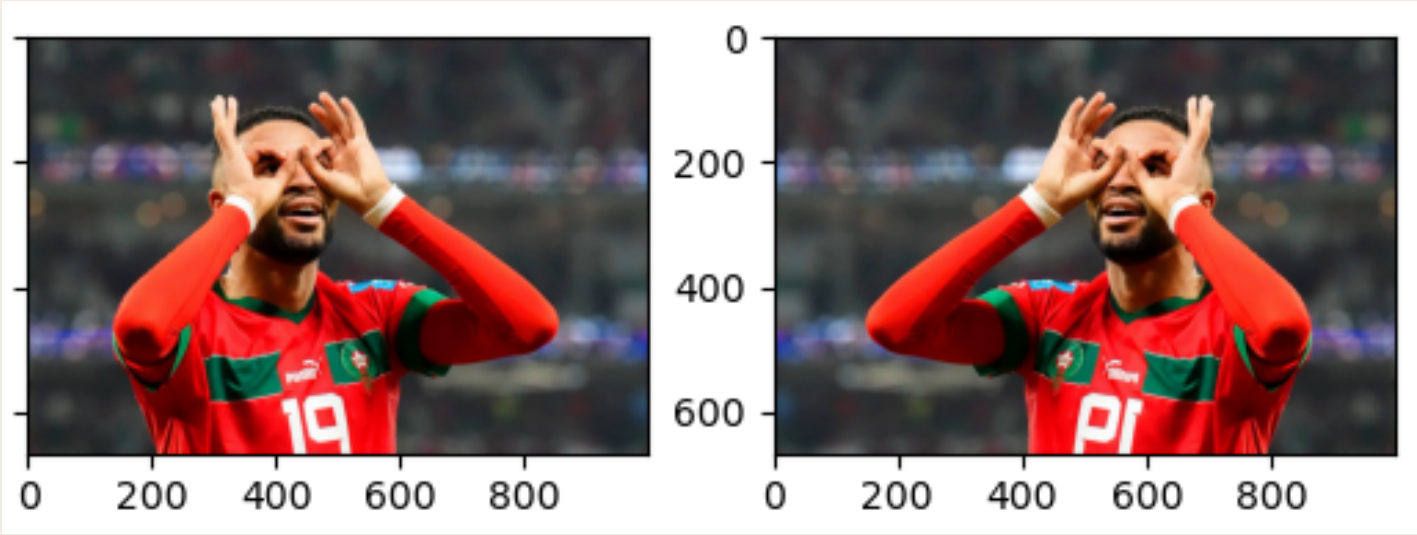
Code

```
1 def retournemet(image):
2     n,_,_=image.shape
3     for i in range(n//2):
4         image[i],image[n-1-i]=copy(image[n
    -1-i]),copy(image[i])
5         #essayer d'enlever copy
```

Miroire

Exercice l'image miroire

Ecrire une fonction Miroire(img) qui renvoie l'image miroire de img.



Réponse

```
1 def miroir(image):
2     return [[l[i] for i in range(len(l)
    -1,-1,-1)] for l in image]
```

Passage en niveaux de gris

Pour convertir une image couleur en niveaux de gris, il faut donc remplacer les valeurs des trois canaux par une unique valeur représentant la luminosité du pixel. on peut utiliser la formule suivante:

$$0.2126 \times Rouge + 0.7152 \times Vert + 0.0722 \times Bleu$$

Passage en niveaux de gris

Exerice: Image Grise

Écrire une fonction Monochrome(img) qui prend en argument une image en couleurs et la transforme en une image en niveaux de gris selon la formule précédente, et vérifier son bon fonctionnement.

Passage en niveaux de gris

Exerice: Image Grise

Écrire une fonction Monochrome(img) qui prend en argument une image en couleurs et la transforme en une image en niveaux de gris selon la formule précédente, et vérifier son bon fonctionnement.

Réponse

```
1 def Monochrome(image):
2     return 0.2126*image[:,:,0]+0.7152 *image
    [:,:,1]+0.0722 *image[:,:,2]
3 def triplet(image):
4     return np.array([[[c]*3 for c in l]for l
    in image])
```

Courbe Tonale

Retoucher
une image revient à modifier les valeurs de certains pixels. On peut le faire localement (à un endroit bien précis de l'image) ou globalement. Dans ce dernier cas, on utilise un outil appelé "courbe tonale", qui ressemble au dessin ci-contre.

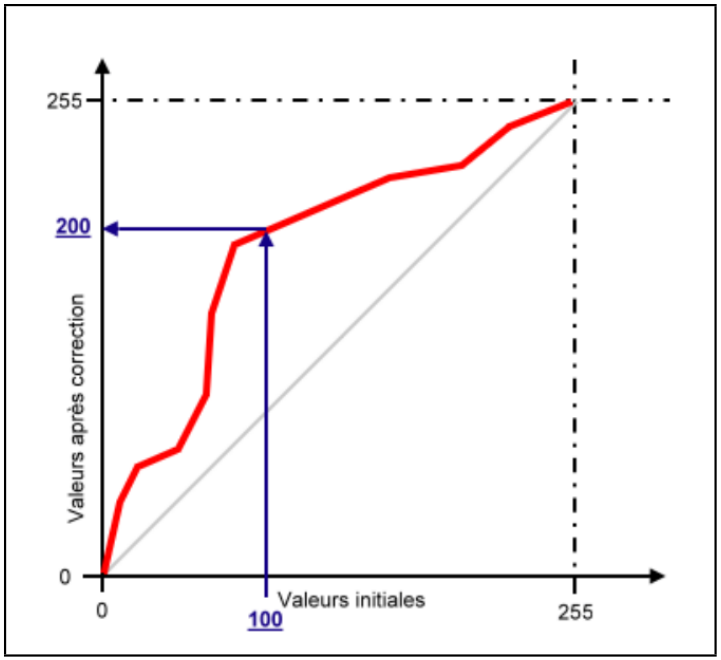
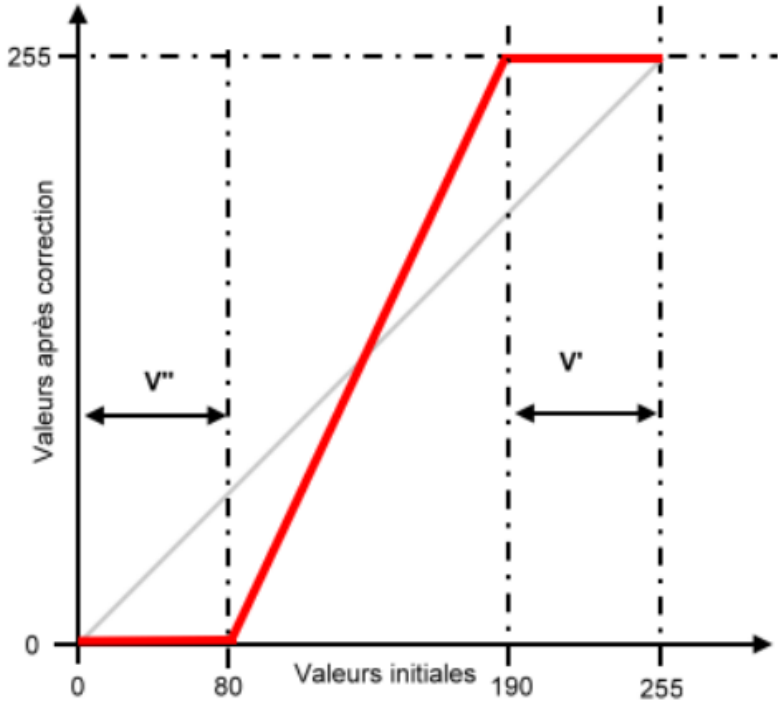
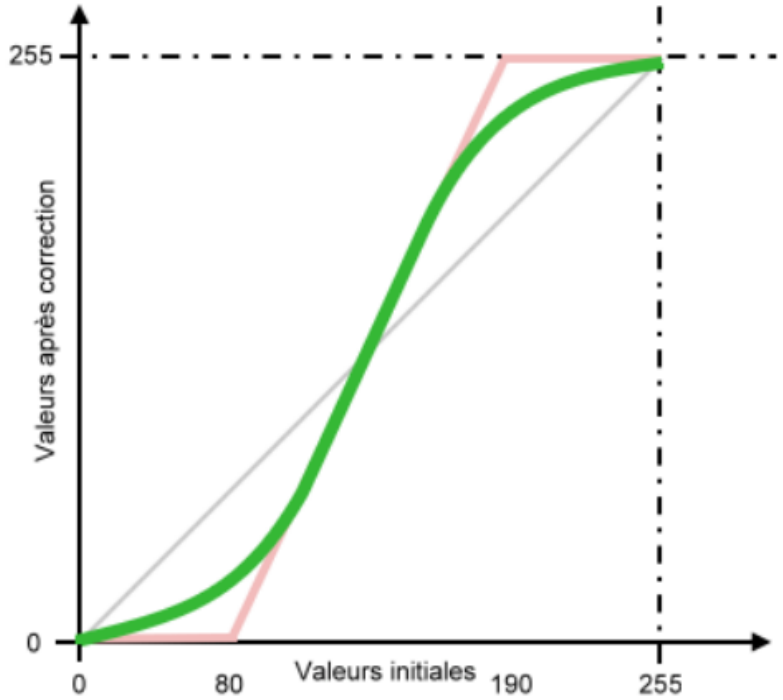


Figure: Courbe Tonale d'une image

Augmenter le contraste

Pour rendre une image plus contrastée, il faut assombrir les points foncés et éclaircir les points clairs, par exemple comme dans les figures ci-dessous :

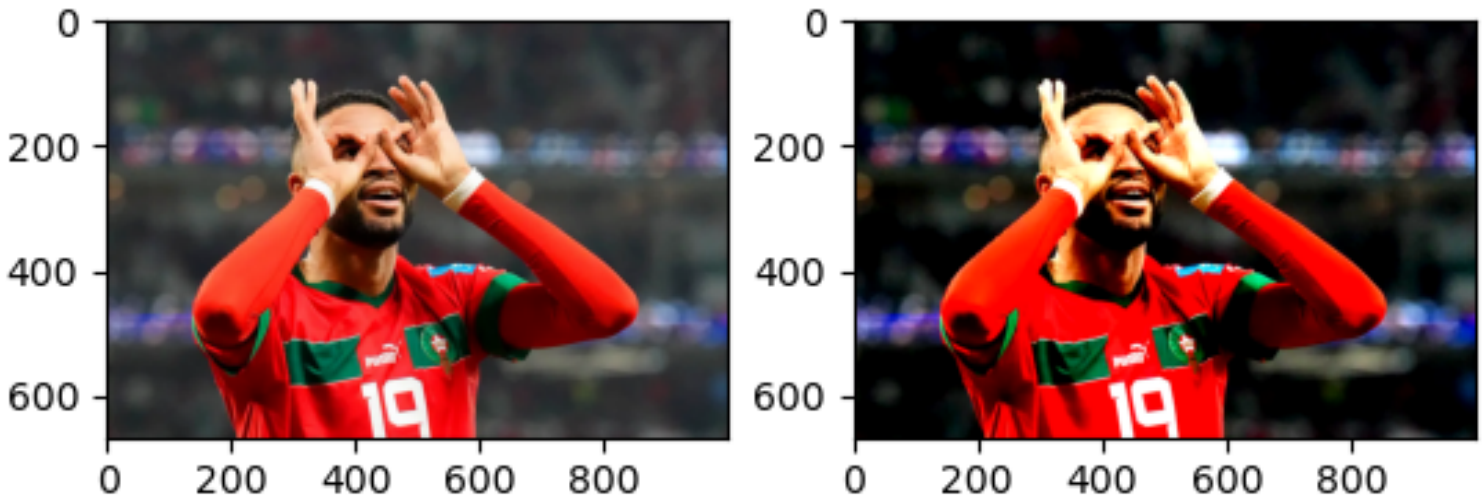




Augmenter le contraste

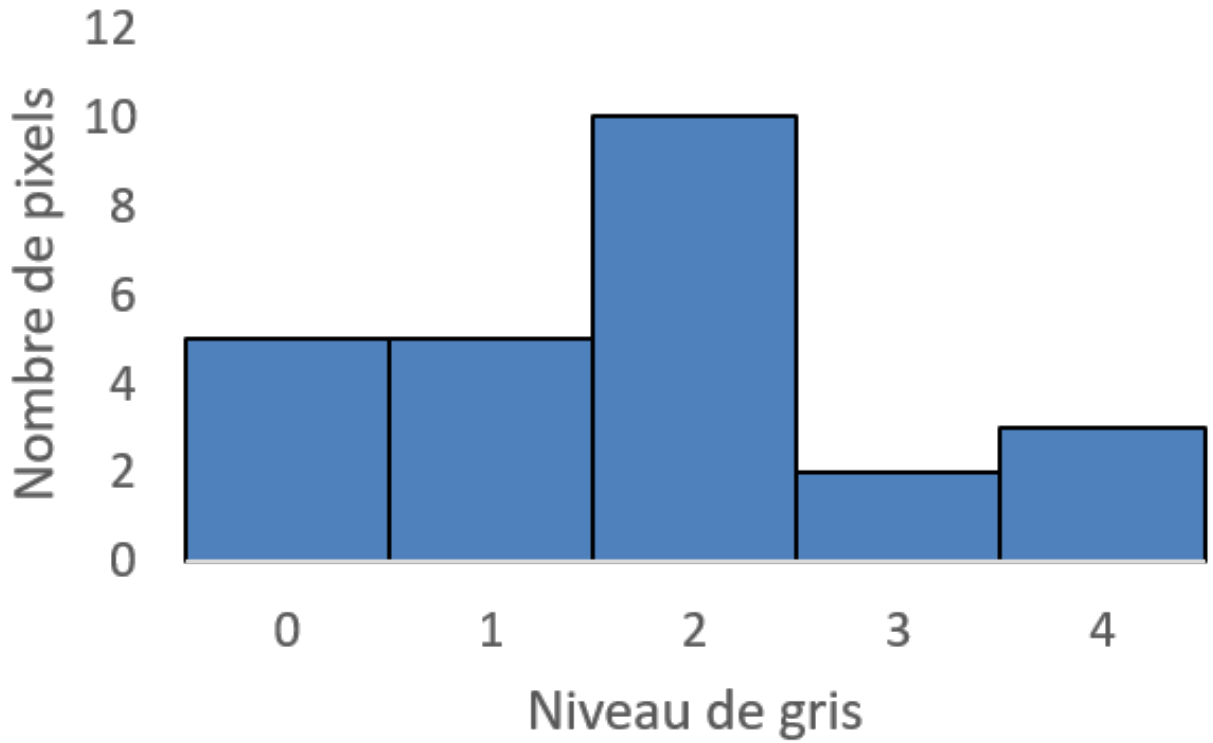
Corriger le contraste d'une image consiste à modifier la répartition des valeurs entre 0.0 et 1.0 dans chacun des canaux. Cela consiste à appliquer, pour chaque canal de chaque pixel, une fonction définie de la sorte :

- si la valeur v est inférieure à vmin, on la remplace par 0 ;
- si la valeur v est supérieure à vmax, on la remplace par 255 ;
- sinon, on remplace la valeur v par $\frac{v-vmin}{vmax-vmin}$.



Histogramme d'une image

L'histogramme d'une image est le graphique qui représente le nombre de pixels existant pour chaque valeur.
Calculer l'histogramme de l'image en niveaux de gris, c'est en d'autres termes compter combien il y a de pixels pour chaque nuance de gris.



Exerice

Ecrire Une fonctione **Histograme(image-grise)** qui permet de calculer l'histograme d'une image grise passée en paramètre, et qui retourne une liste des couples (niveu de gris,nombre de pixels).

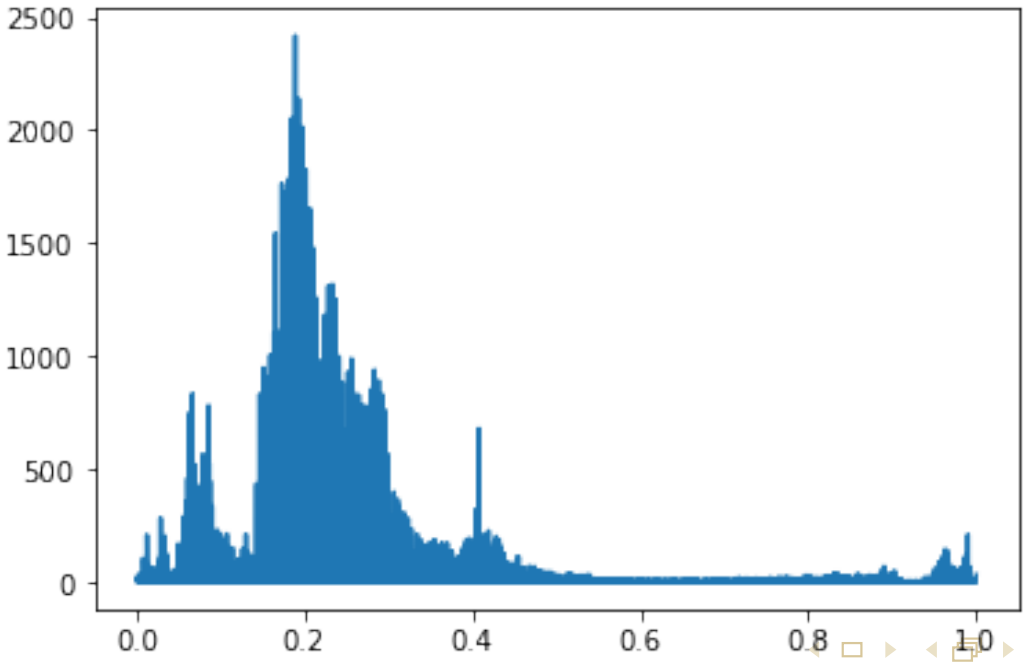
Réponse

```
1 def histogram(image_grise):
2     histo=dict()
3     for l in image_grise:
4         for c in l:
5             if c not in histo:
6                 histo[c]=1
7             else:
8                 histo[c]+=1
9     return histo.items()
```

Histogramme d'une image

Réponse

```
1 image_grise2=Monochrome(img)
2 histo2=sorted(histogram(image_grise2))
3 pixels,occurences=np.array(histo2).transpose
    ()
4 plt.plot(pixels,occurences)
```



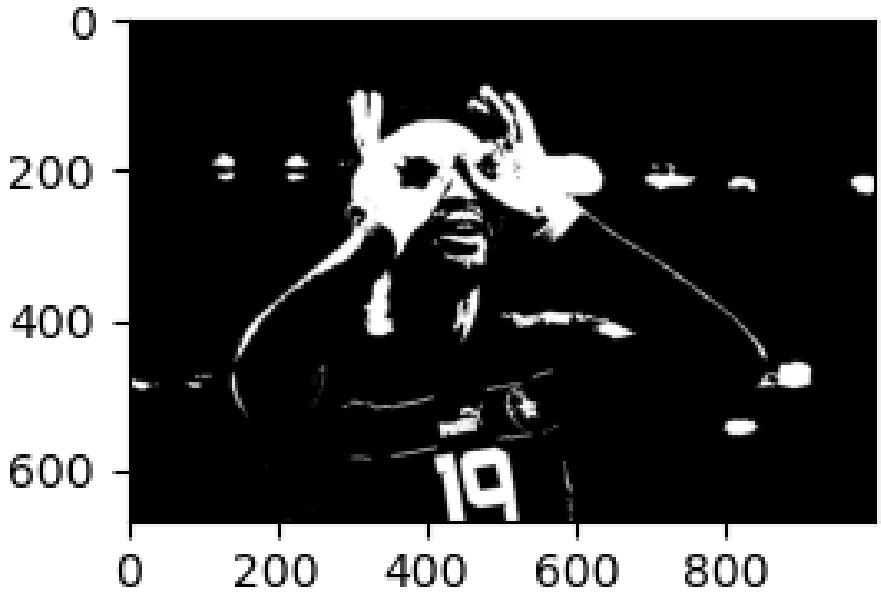
Seuillage fixe

Le seuillage est un traitement qui permet de selectionner les informations significatives dans une image, ce traitement nécissite le réglage d'un seuil **S**,

- Si la valeur d'un pixel Im[i,j] est inférieur au seuil S, alors la valeur de ce pixel est rémplacé par 0
- Sinon la valeur de ce pixel est remplacé par 1

Ainsi , on obtient une image en noir et blanc sans niveau de gris

Seuillage fixe

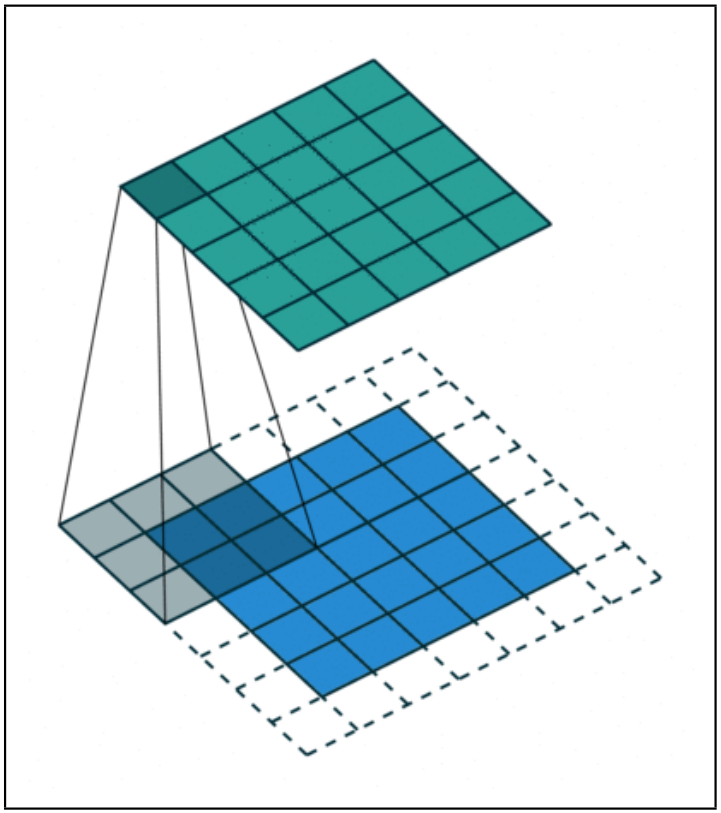


Seuillage Fixe

```
1 def seuillage(image,S=0.5):
2     for i in range(len(image)):
3         for j in range(len(image[i])):
4             image[i,j]=0 if image[i,j]<S
    else 1
5     return image
```

Application de filtres

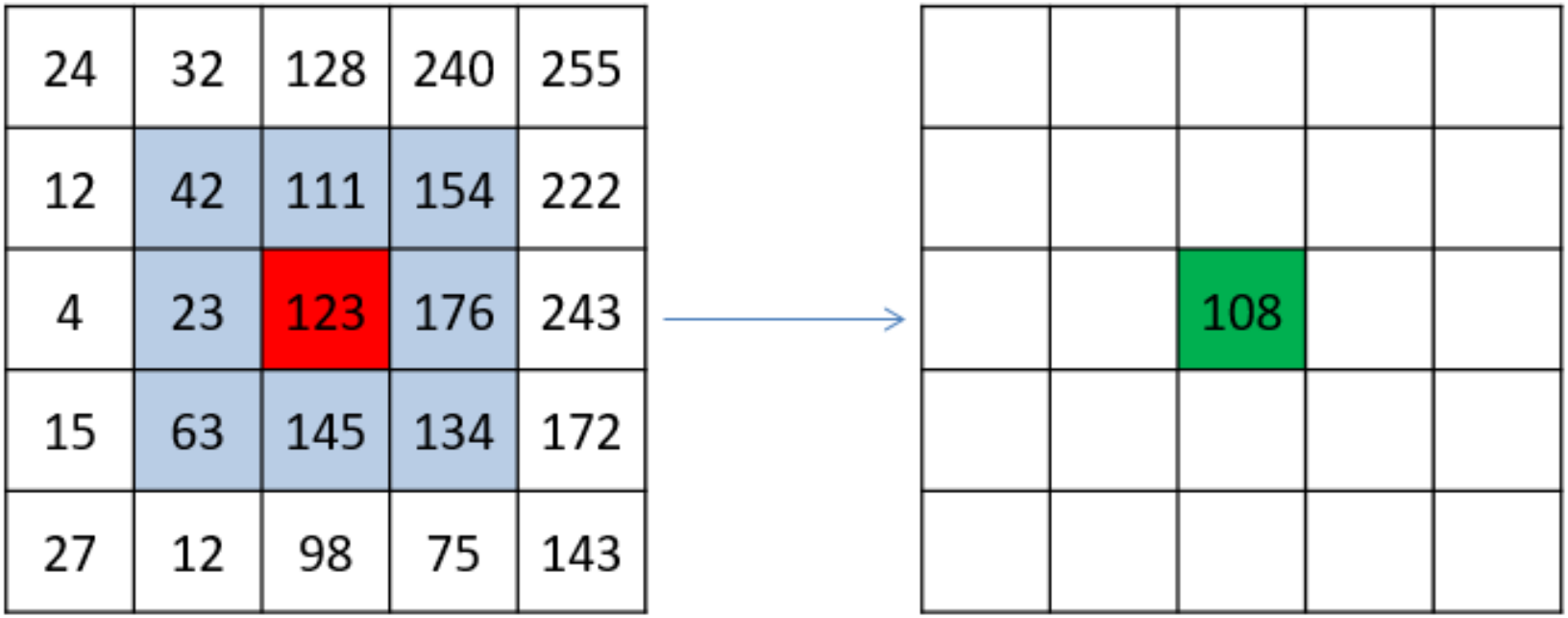
Un filtre est une transformation mathématique permettant de modifier la valeur d'un pixel en fonction des valeurs des pixels avoisinants, affectées de coefficients.



Le filtre est représenté par un tableau (une matrice), caractérisé par ses dimensions et ses coefficients, dont le centre correspond au pixel concerné.

Filtre moyenneur: Image Flou

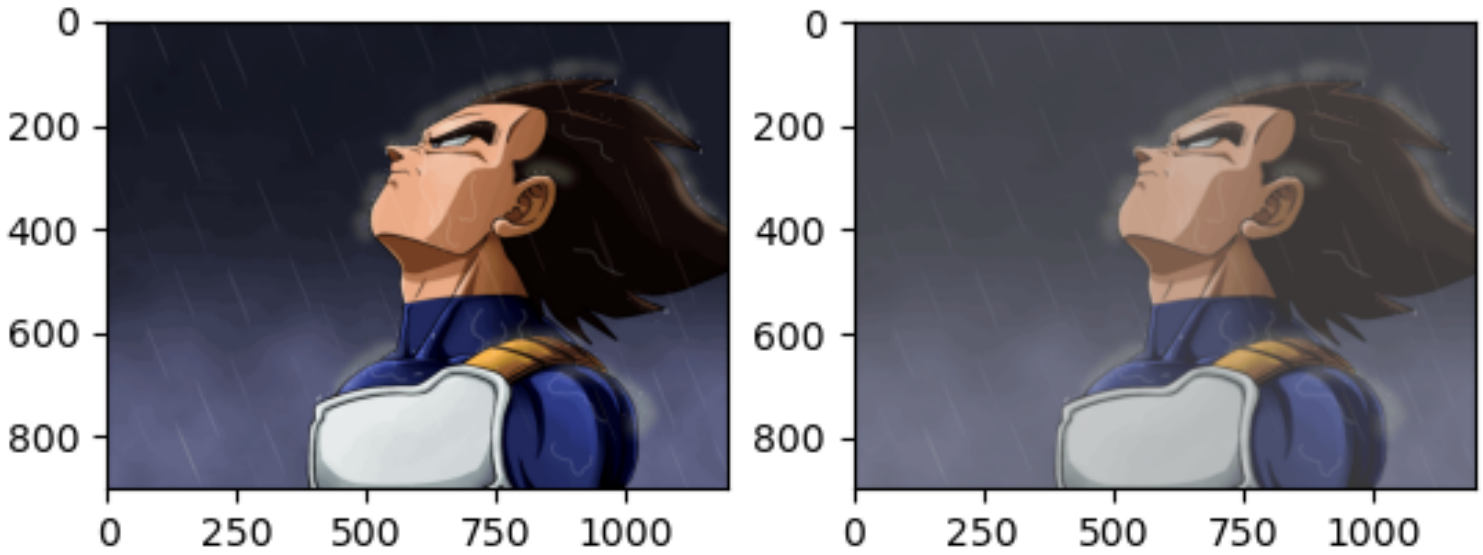
Pour appliquer un flou à l'image, on remplace chaque couleur de chaque pixel par une moyenne des couleurs des neufs pixels entourant le pixel considéré.



Filtre moyenneur: Image Flou

Exercice

Écrire une fonction Floute(img) qui prend en entrée une image img et retourne une nouvelle image correspondant à un flou de l'image originale, et vérifier son bon fonctionnement.



Filtre moyenneur: Image Flou

Filtre moyenneur: calcul vectoriel

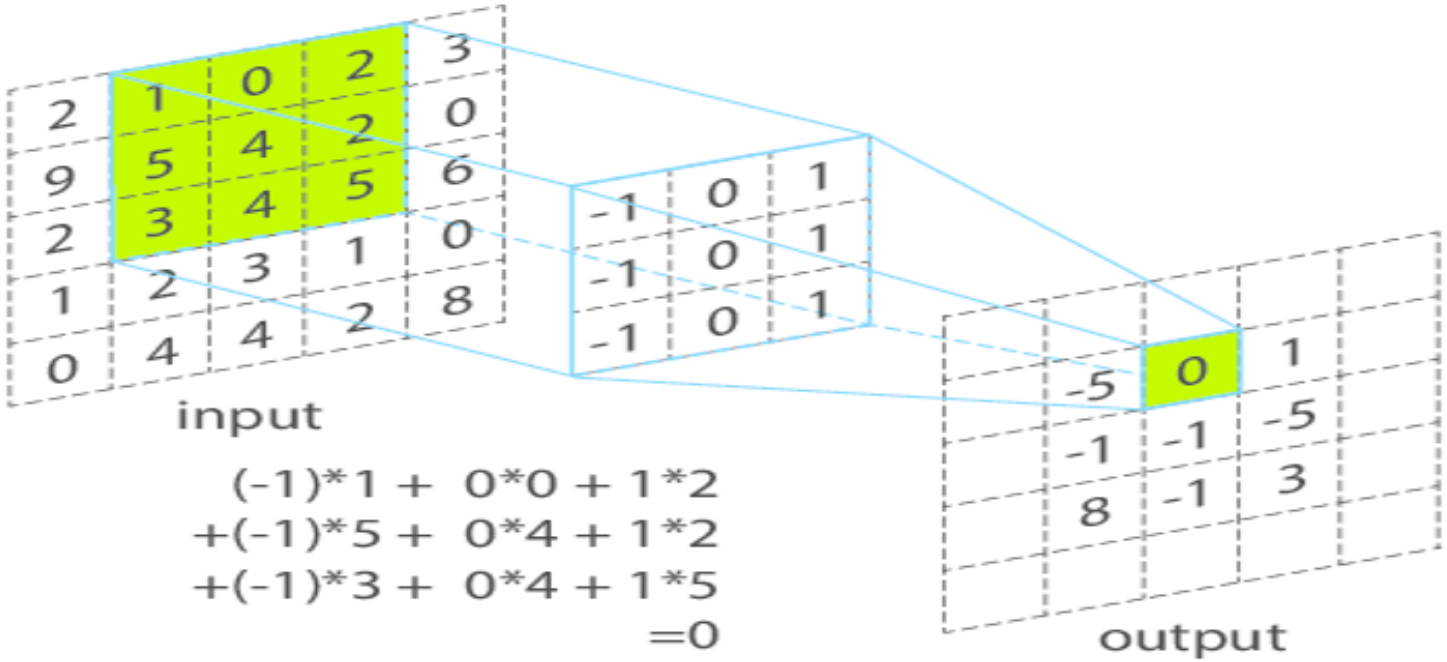
```
1 def flou(image):
2     return (image[2:,2:]+image[2:,1:-1]+
    image[2:,:-2]+\
3     image[1:-1,2:]+image[1:-1,:-2]+\
4     image[:-2,2:]+image[:-2,1:-1]+image
    [:-2,:-2])/9
```

Filtre moyenneur: calcul itératif

```
1 for i in range(1,len(image)-1):
2     for j in range(1,len(image[0])-1):
3         newimage[i,j]=(image[i-1,j-1]+image[
    i-1,j]+image[i-1,j+1]+image[i,j-1]+image[
    i,j+1]+image[i+1,j-1]+image[i+1,j]+image[
    i+1,j+1])/9
```

Application de filtres=Produit de Convolution

La convolution, ou produit de convolution, est une généralisation du filtre moyenneur où l'on considère cette fois une moyenne pondérée. La fenêtre glissante est alors elle même une image qui contient les coefficients de pondération. On l'appelle généralement noyau de convolution ou masque de convolution (kernel ou mask en anglais) :



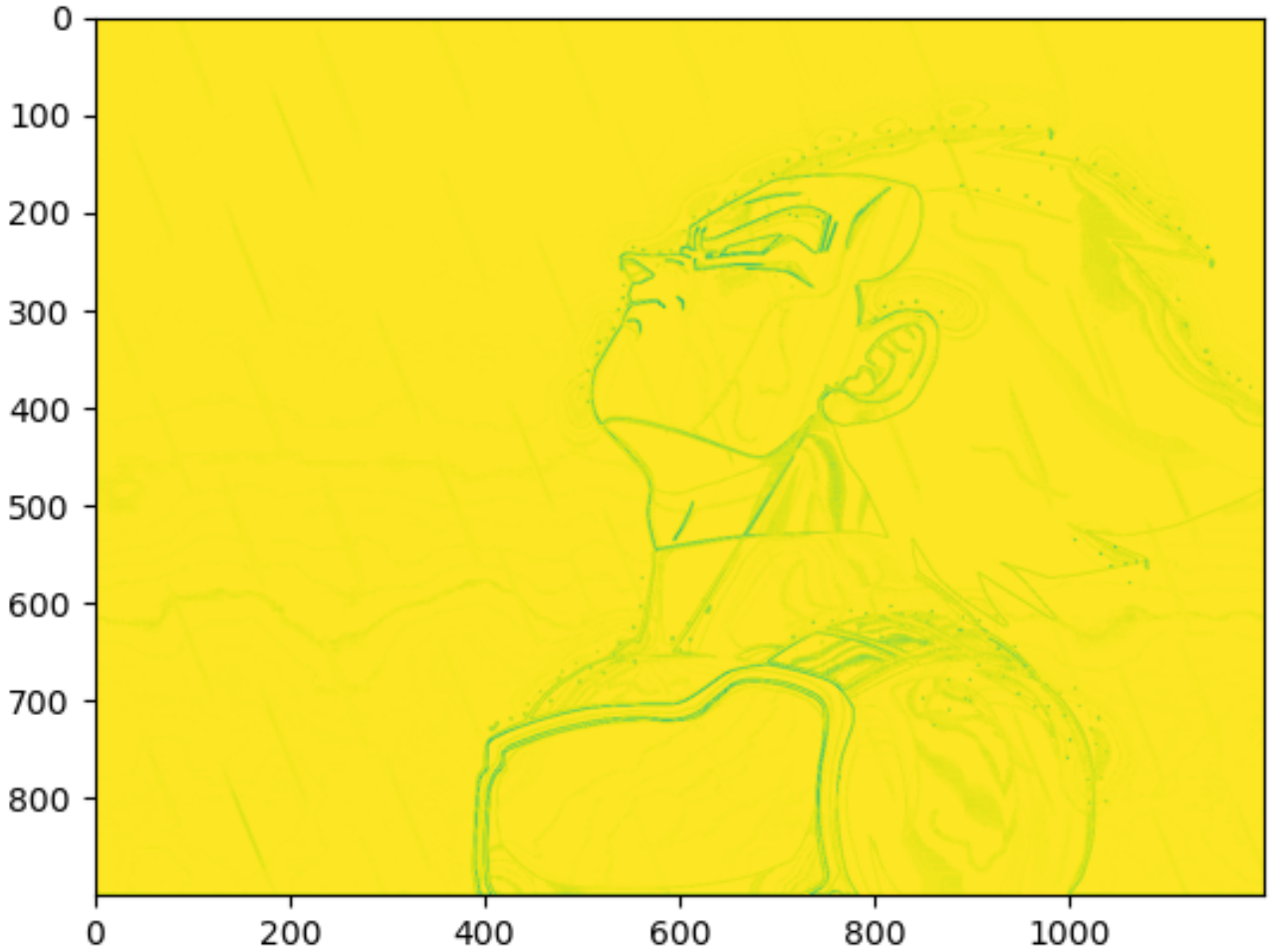
Produit de Convolution

Produit de convolution

```
1 def convolution(image,noyeau):
2     l,c=image.shape
3     image_augmentee=np.zeros((l+2,c+2))
4     image_augmentee[1:-1,1:-1]=image
5     newimage=np.zeros_like(image)
6     for i in range(l):
7         for j in range(c):
8             newimage[i,j]=abs(sum(sum(noyeau
    *image_augmentee[i:i+3,j:j+3])))
9     return newimage
```

Détection de contours

La détection de contours consiste à chercher les courbes continues le long des zones de fortes variations dans l'image.



Détection de contours

La détection de zones de variation des niveaux de gris de l'image correspond à l'opération de dérivation. Comme une image numérique n'est pas une fonction continue, la notion de dérivée n'est pas formellement définie et on utilisera un analogue appelé gradient.
Comme une image a 2 dimensions, le gradient de l'image f, notée ∇f, est une image vectorielle,
donnée par les deux dérivées partielles: