# II. Analyse élémentaire d'images

**1- HISTOGRAMME**

- **Densité de probabilité des niveaux de gris d'une image = histogramme NORMALISÉ**

- **En abscisses : le nombre de niveaux de quantification de l'image**

- **En ordonnées : le nombre de pixels de l'image correspondant à chaque niveau de quantification**

- **Sous Matlab/Octave : instruction IMHIST**
  **(faire « *>> help imhist* » pour lire les caractéristiques de la fonction (ou aussi « *>> doc imhist* » sous Matlab))**
  **(sous Octave : il faut au préalable avoir chargé le package 'image' à l'aide de « *>> pkg load image* »)**

# II. Analyse élémentaire d'images

```
-- Function File: imhist (I)
-- Function File: imhist (I, N)
-- Function File: imhist (X, CMAP)
-- Function File: [COUNTS, X] = imhist (...)
    Produce histogram counts of image I.

    The second argument can either be N, a scalar that specifies the
    number of bins; or CMAP, a colormap in which case X is expected to
    be an indexed image.  If not specified, N defaults to 2 for binary
    images, and 256 for grayscale images.

    If output is requested, COUNTS is the number of counts for each bin
    and X is a range for the bins so that 'stem (X, COUNTS)' will show
    the histogram.

    _Note:_ specially high peaks that may prevent an overview of the
    histogram may not be displayed.  To avoid this, use 'axis "auto y"'
    after the call to 'imhist'.

    See also: hist, histc, histeq.

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at https://www.octave.org and via the help@octave.org
mailing list.
```

# II. Analyse élémentaire d'images

- **Exemple :**
  ***>> I = [0.5 0.4 0.3 ; 0.4 0.3 0.4 ; 0.3 0.4 0.5];***
  ***>> [n, x] = imhist(I, 11)***
  **rend :**
  ***n = [0  0  0  3  4  2  0  0  0  0  0 ]***
  ***x = [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]***

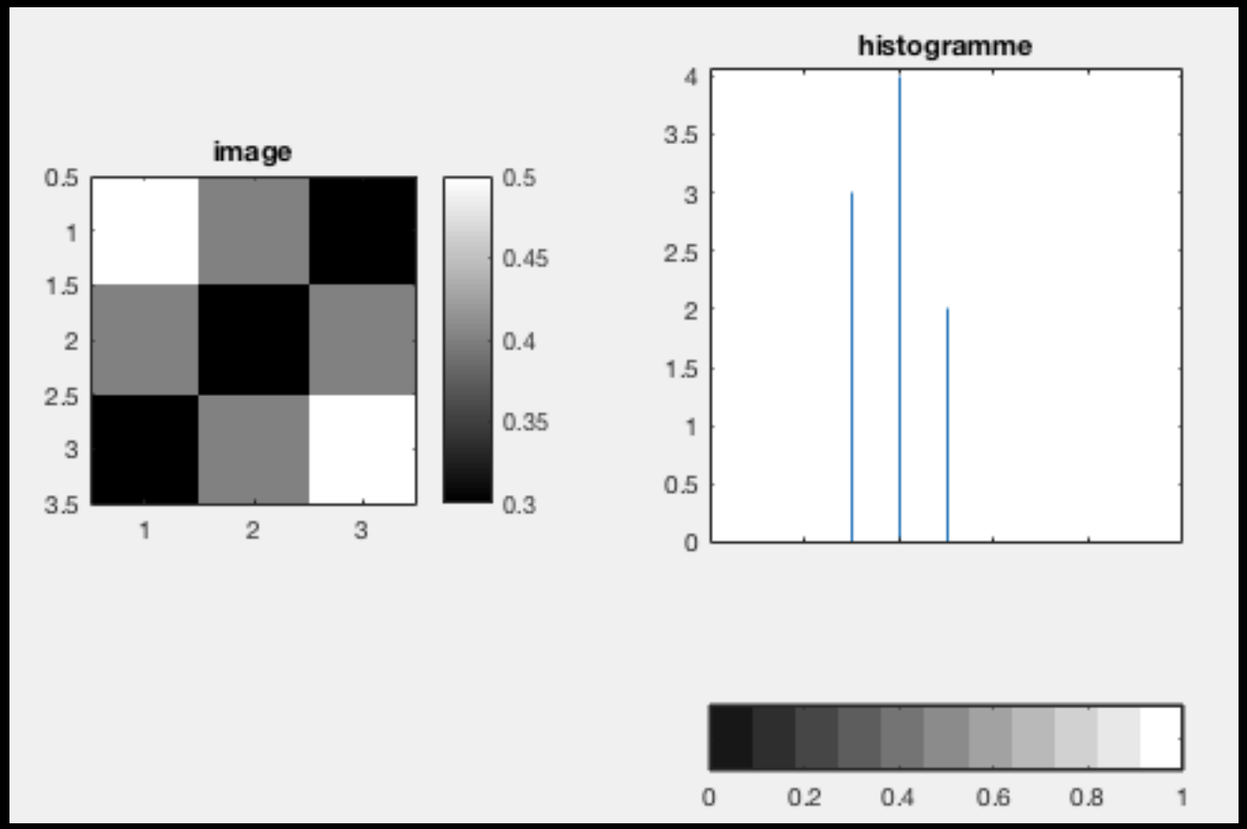
  ***>> figure, imhist(I, 11)***
  **rend : la représentation de cet histogramme**

- **Représenter à la fois l'image et son histogramme :**

  ***>> figure, subplot(1, 2, 1), colormap(gray)***
  ***>> imagesc(I), colorbar, axis('square')***
  **(ou :** ***>> imshow(I, 'InitialMagnification','fit'), colorbar*****)**
  ***>> subplot(1, 2, 2), imhist(I, 11)***

# II. Analyse élémentaire d'images

```
clear
close all

I=[.5 .4 .3; .4 .3 .4; .3 .4 .5]
[n, x] = imhist(I, 11);
'niveaux de quantification dans l"image = ', x
'nb de px dans chq niveaux = ', n

figure
subplot(1,2,1)
imagesc(I), colormap(gray), colorbar, axis('square'), title('image')
subplot(1,2,2)
imhist(I, 11), axis('square'), title('histogramme')
```

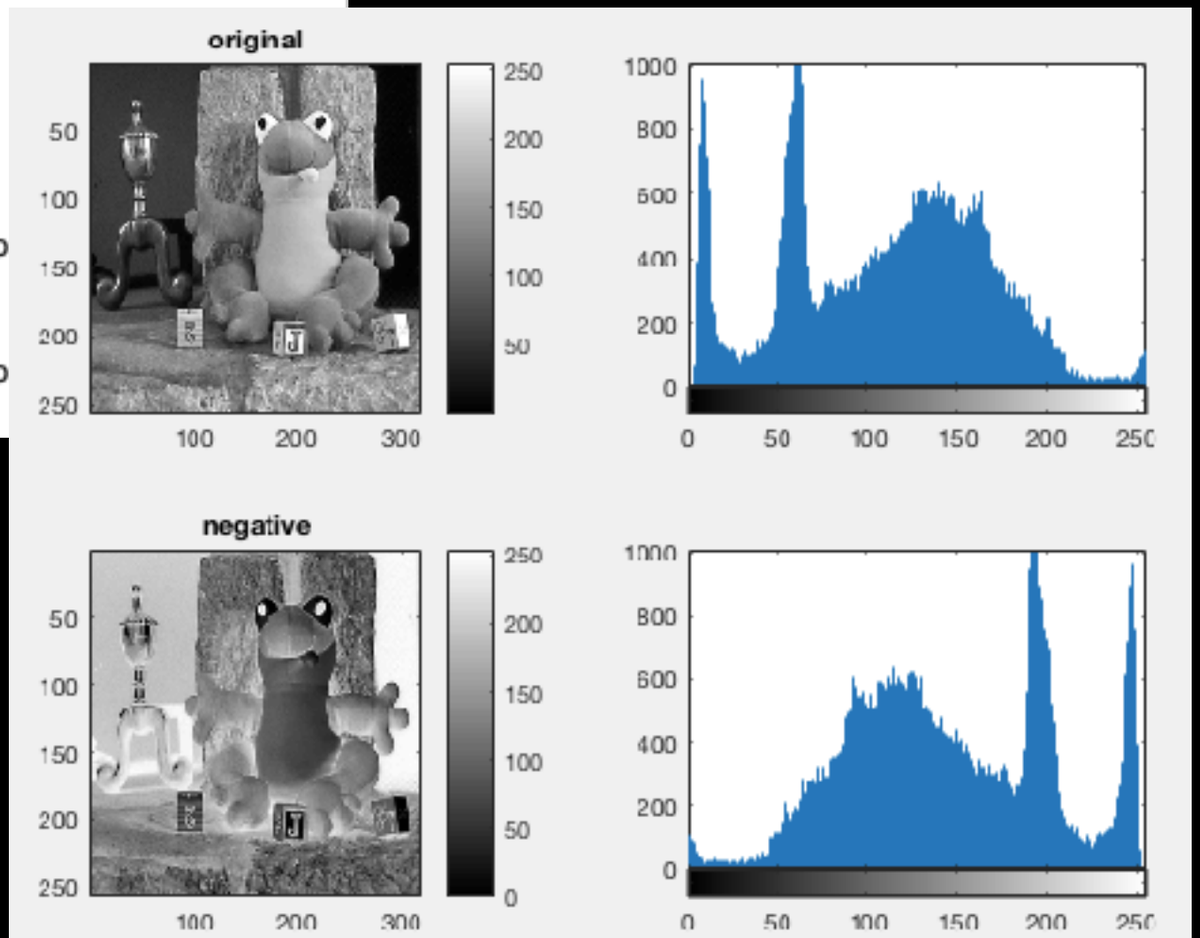


# II. Analyse élémentaire d’images

- **EXERCICE 1 : Lire une image plus compliquée (p.ex. frog.jpg) en couleur. La convertir en image d’intensité puis tracer son histogramme ainsi que celui de son négatif, sur 256 niveaux. Comparer.**

# II. Analyse élémentaire d'images

```
clear
close all

frog=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg');
'type(frog) :', whos frog

frog_int=rgb2gray(frog);
'type(frog_int) :', whos frog_int
'min(frog_int) :', min(min(frog_int))
'max(frog_int) :', max(max(frog_int))

frog_neg=255-frog_int;
'min(frog_neg) :', min(min(frog_neg))
'max(frog_neg) :', max(max(frog_neg))

figure
subplot(2,2,1)
imagesc(frog_int), colormap(gray), title('original'), colo
subplot(2,2,2), imhist(frog_int, 256)
subplot(2,2,3)
imagesc(frog_neg), colormap(gray), title('negative'), colo
subplot(2,2,4), imhist(frog_neg, 256)
```



# II. Analyse élémentaire d'images

## 2- ÉGALISATION D'HISTOGRAMME

- **Peut améliorer des images de mauvaise qualité (contraste : trop sombre/ trop claire, mauvaise répartition des niveaux d'intensité).**

- **But : rendre l'histogramme d'une image aussi plat que possible afin que chaque niveau d'intensité soit équitablement représenté (et ainsi profiter de toute la dynamique possible).**

- **Moyen : une transformation *f* des niveaux d'intensité qui pour l'intensité *i* d'un pixel donné dans l'image originale va rendre *f(i)* dans l'image que l'on va appeler « égalisée ».**

- **En général, on choisit une fonction *f* en escalier, en déterminant largeur et hauteur de chaque marche de manière à aplanir au mieux l'histogramme.**

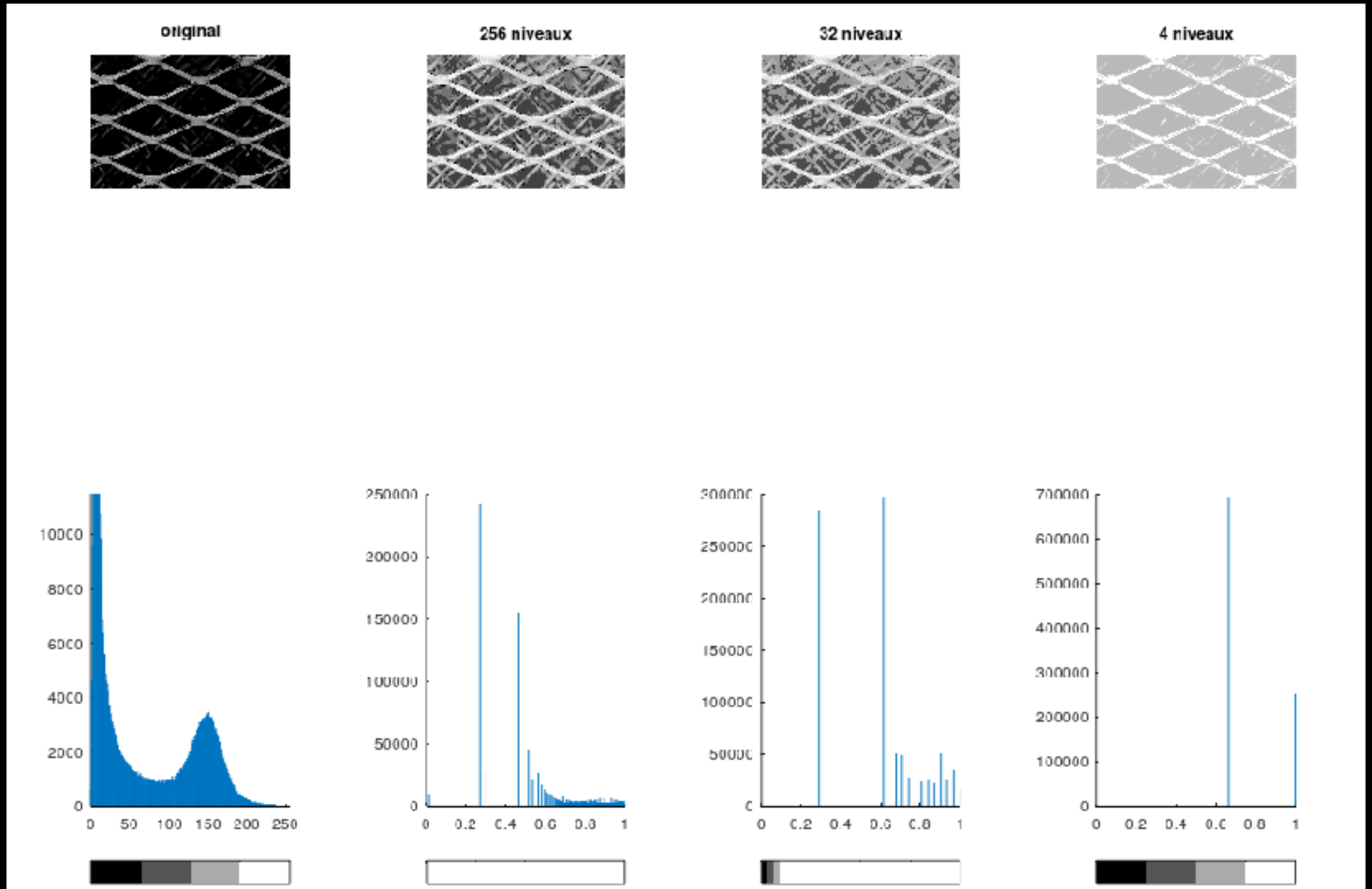
- **Sous Matlab/Octave : instruction HISTEQ**
  ***>> frog_eq = histeq(frog_int, n) ;***
  **avec : *n* = nombre de niveaux d'intensité dans l'image égalisée**

# II. Analyse élémentaire d'images

- **EXERCICE 2 : Égaliser l'histogramme d'une image de votre choix (p.ex. le grillage rouillé en niveaux de gris). Comparer avant et après (images et histogrammes). Égaliser avec moins de niveaux, que se passe-t-il ?…**

```
clear
close all

I=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/grillage.jpeg');
Iint=rgb2gray(I);
figure, subplot(2,4,1), imshow(Iint), title('original')
subplot(2,4,5), imhist(Iint, 256)

n = 256;
Ieq = histeq(Iint, n);
subplot(2,4,2), imshow(Ieq), title('256 niveaux')
subplot(2,4,6), imhist(Ieq, n)

n = 32;
Ieq = histeq(Iint, n);
subplot(2,4,3), imshow(Ieq), title('32 niveaux')
subplot(2,4,7), imhist(Ieq, n)

n = 4;
Ieq = histeq(Iint, n);
subplot(2,4,4), imshow(Ieq), title('4 niveaux')
subplot(2,4,8), imhist(Ieq, n)
```

# II. Analyse élémentaire d'images



# II. Analyse élémentaire d'images

**3- REMARQUE**

- **L'histogramme représente une analyse en un seul point !**

  **=> négligence de toute corrélation entre les points (voisins ou pas)**

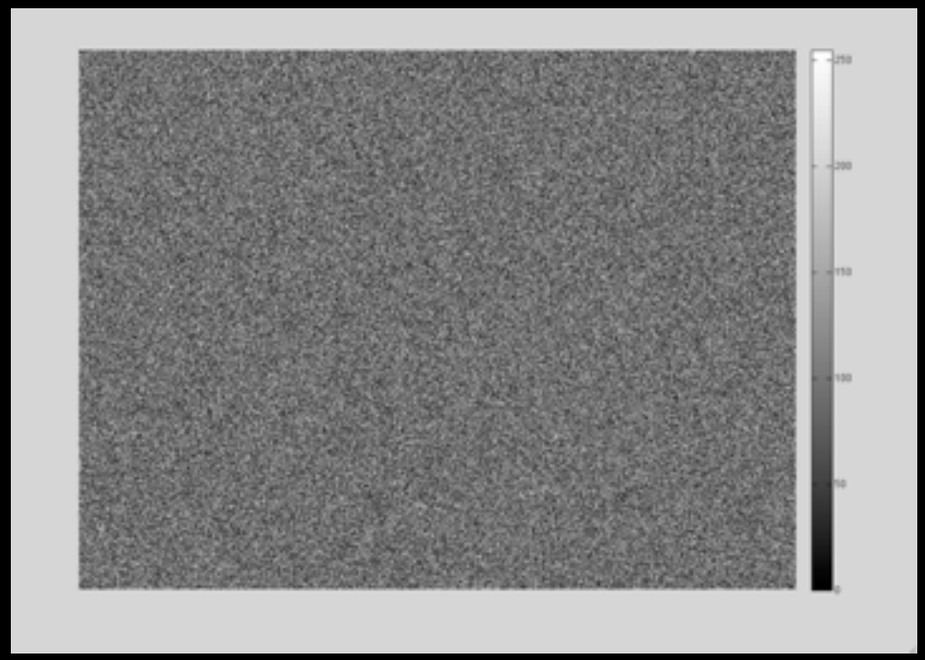
  **=> pas de notion de forme, de détails, de fréquence spatiale**

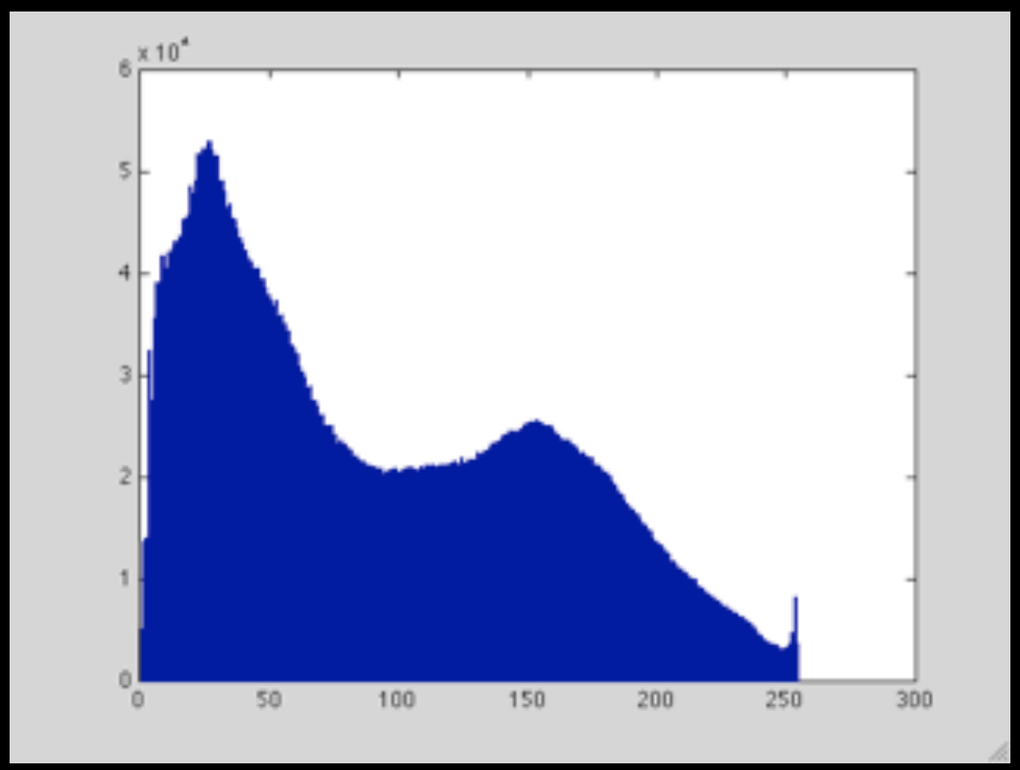
  **=> on peut redistribuer tous les pixels d'une image, la rendant méconnaissable, complètement différente ou même sans forme notable, sans texture aucune, elle aura toujours le même histogramme.**

- **Pour avoir une idée de la texture : analyse en 2 points nécessaire**

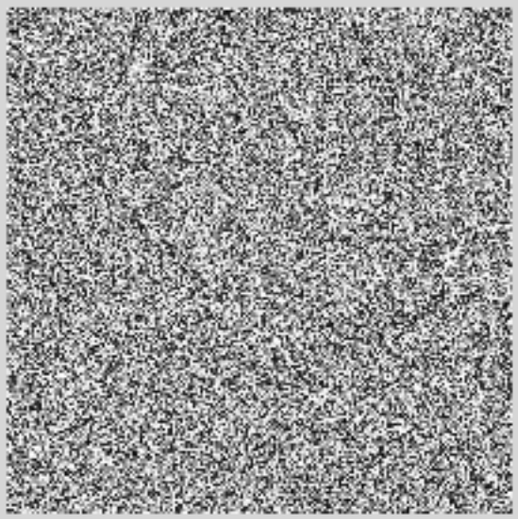
  **=> densité de probabilité bi-dimensionnelle, i.e. co-occurrence de valeurs en deux points d'une image…**

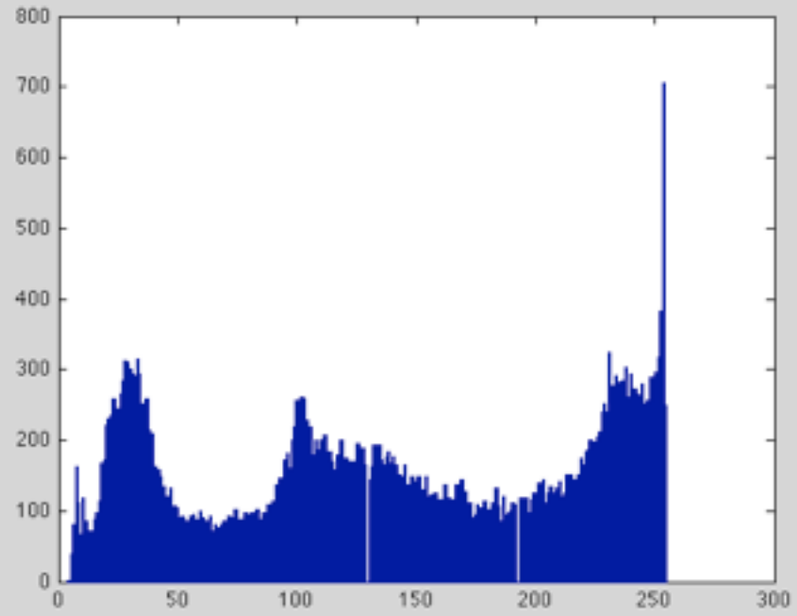
# II. Analyse élémentaire d’images

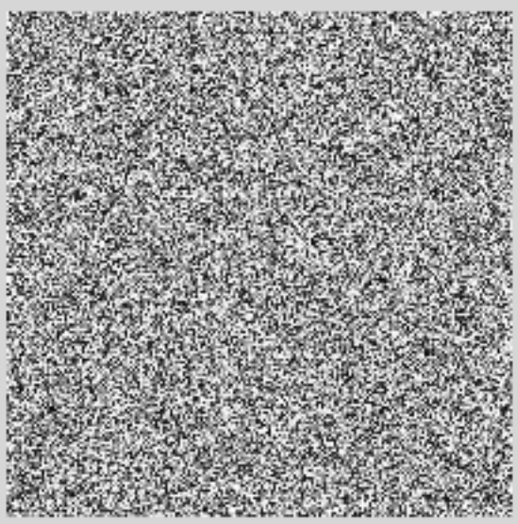




# II. Analyse élémentaire d'images



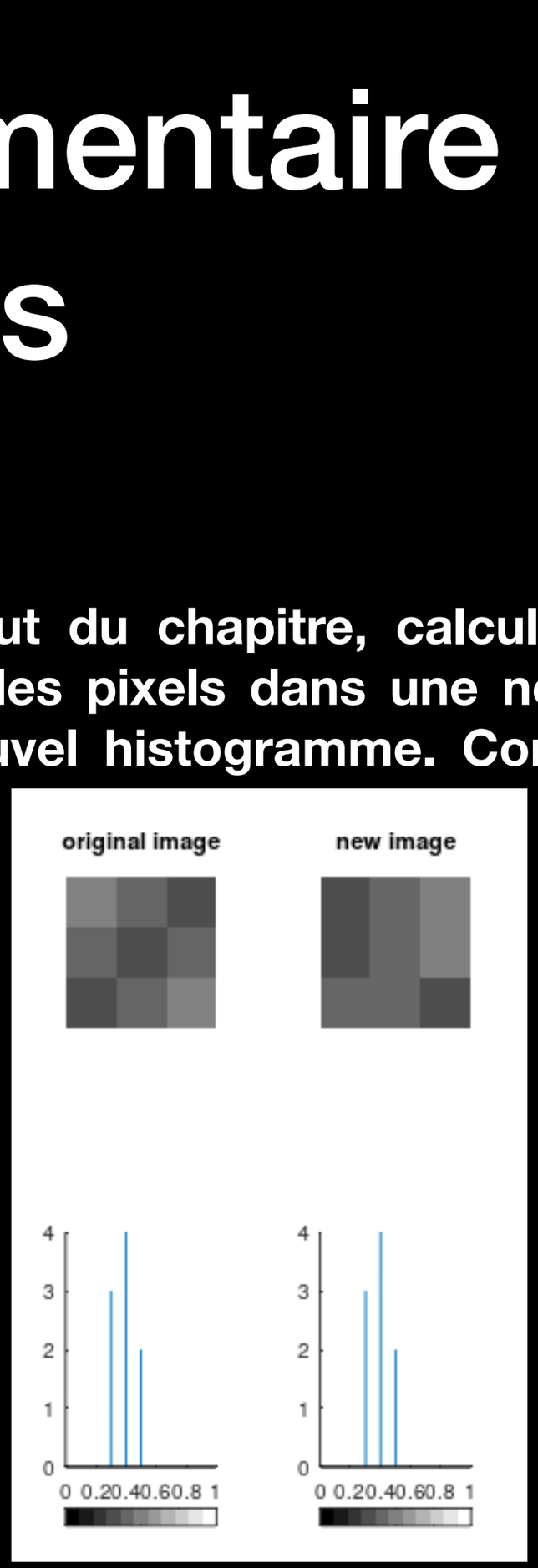


# II. Analyse élémentaire d'images

- **EXERCICE 3 : Reprendre l'image *I* du début du chapitre, calculer son histogramme, puis redistribuer les valeurs des pixels dans une nouvelle image (ou même plusieurs). Calculer le nouvel histogramme. Comparer images et histogrammes.**

```
clear
close all

I=[.5 .4 .3 ; .4 .3 .4 ; .3 .4 .5];
figure
subplot(2,4,1), imshow(I), title('original image')
subplot(2,4,5), imhist(I,11)

I2=[.3 .4 .5 ; .3 .4 .5 ; .4 .4 .3];
subplot(2,4,2), imshow(I2), title('new image')
subplot(2,4,6), imhist(I2,11)
```



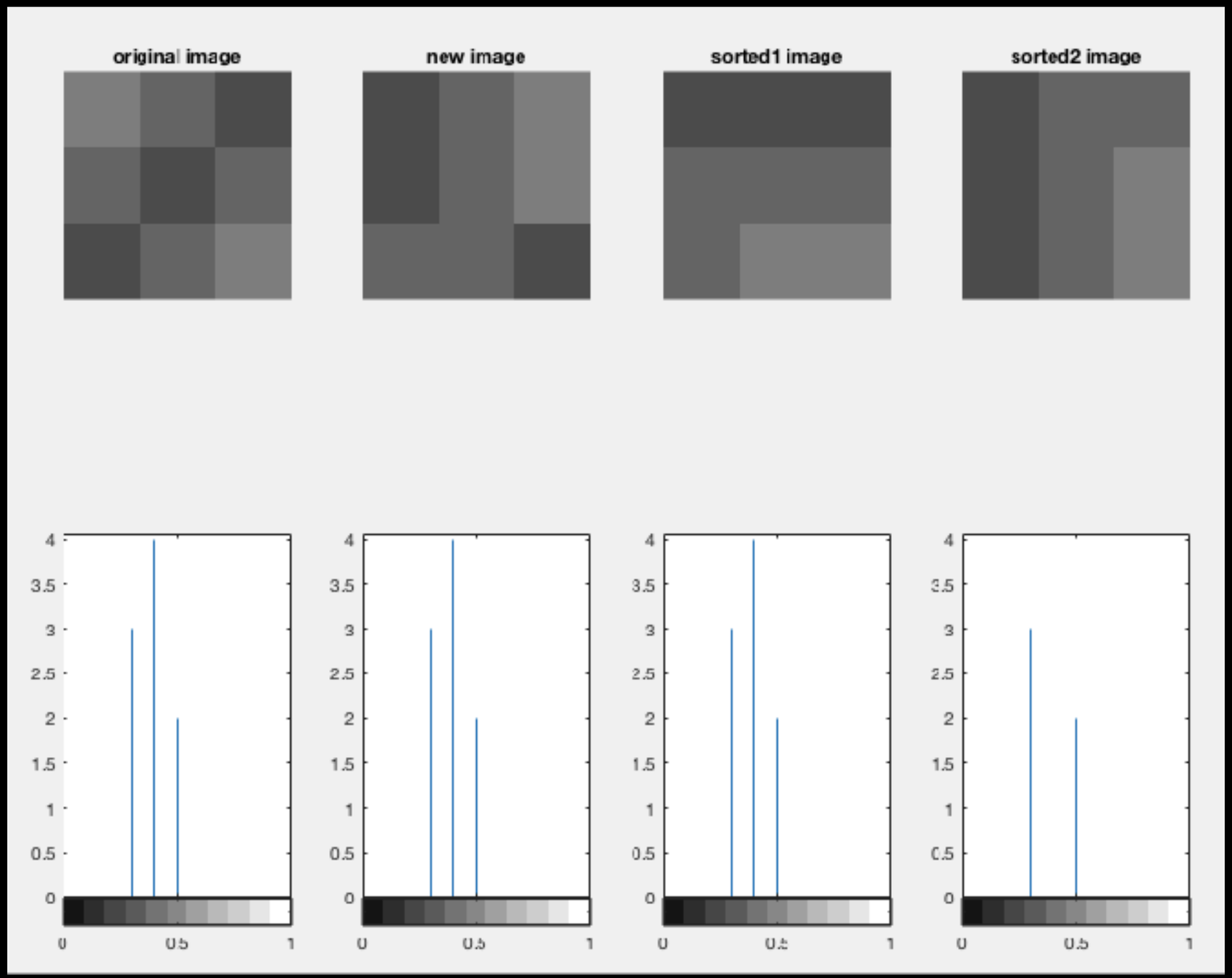
# II. Analyse élémentaire d'images

- **EXERCICE 3 (suite) : Trouver une fonction pouvant transcoder l'image et vérifier que l'on a toujours le même histogramme quelque soit la distribution spatiale des pixels.**

**plusieurs solutions possibles : sort(sort()), fliplr/flipud, rot90, imtranslate, shuffle (randperm), etc.**

```
clear
close all

I=[.5 .4 .3 ; .4 .3 .4 ; .3 .4 .5];
figure
subplot(2,4,1), imshow(I), title('original image')
subplot(2,4,5), imhist(I,11)

I2=[.3 .4 .5 ; .3 .4 .5 ; .4 .4 .3];
subplot(2,4,2), imshow(I2), title('new image')
subplot(2,4,6), imhist(I2,11)

I3=sort(sort(I),2)
subplot(2,4,3), imshow(I3), title('sorted1 image')
subplot(2,4,7), imhist(I3,11)

I4=sort(sort(I,2))
subplot(2,4,4), imshow(I4), title('sorted2 image')
subplot(2,4,8), imhist(I4,11)
```

# II. Analyse élémentaire d'images



# II. Analyse élémentaire d’images

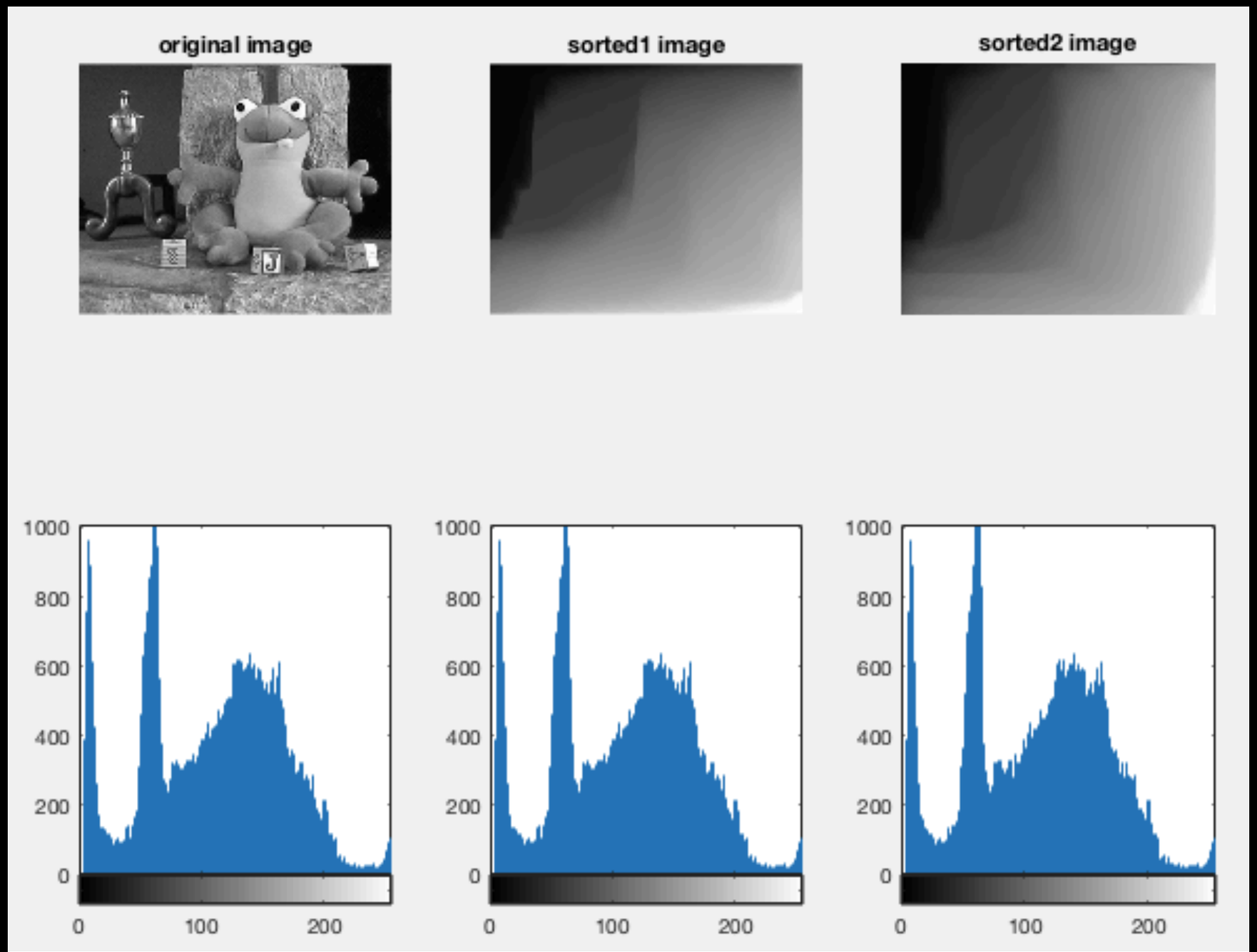
- **EXERCICE 3bis : Appliquer à une image plus complexes (p.ex. la grenouille en niveaux de gris)…**

  **pour rappel, plusieurs solutions possibles : sort(sort()), fliplr/flipud, rot90, imtranslate, shuffle (randperm), etc.**

# II. Analyse élémentaire d'images

```
clear
close all

frog=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg');
frog=rgb2gray(frog);

whos frog

figure
subplot(2,3,1), imshow(frog), title('original image')
subplot(2,3,4), imhist(frog,256)

frog1=sort(sort(frog),2);
subplot(2,3,2), imshow(frog1), title('sorted1 image')
subplot(2,3,5), imhist(frog1,256)

frog2=sort(sort(frog,2));
subplot(2,3,3), imshow(frog2), title('sorted2 image')
subplot(2,3,6), imhist(frog2,256)
```

# II. Analyse élémentaire d'images



# II. Analyse élémentaire d'images

**4- COUPE D'UNE IMAGE**

- **Afficher une image, lancer *IMPROFILE*, un click pour le 1er point, deux clicks pour le 2me.**
  **(pas encore implémenté sous Octave…)**

- **En alternative : *PLOT* ( p.ex. : >> *plot(frog_int(:,20))* —> plot de la col. n. 20)**

# II. Analyse élémentaire d'images

**5- AMÉLIORATION DU CONTRASTE**

- ***IMADJUST*** **: sature 1% des basses et hautes intensités et fait en sorte d'utiliser toute la dynamique disponible (en redistribuant les intensités pour utiliser toute la gamme de représentation permise (par uint8 p.ex.)).**
  ***>> I_adj = imadjust(I)***

- **On peut aussi choisir la gamme de valeurs d'intensité sur laquelle agir :**
  ***>> I_adj2 = imadjust(I, [0, 0.5], [])***
  **(« *[0, 0.5]* » correspondant à [0, moitié des niveaux possibles] - p.ex. !)**

# II. Analyse élémentaire d'images

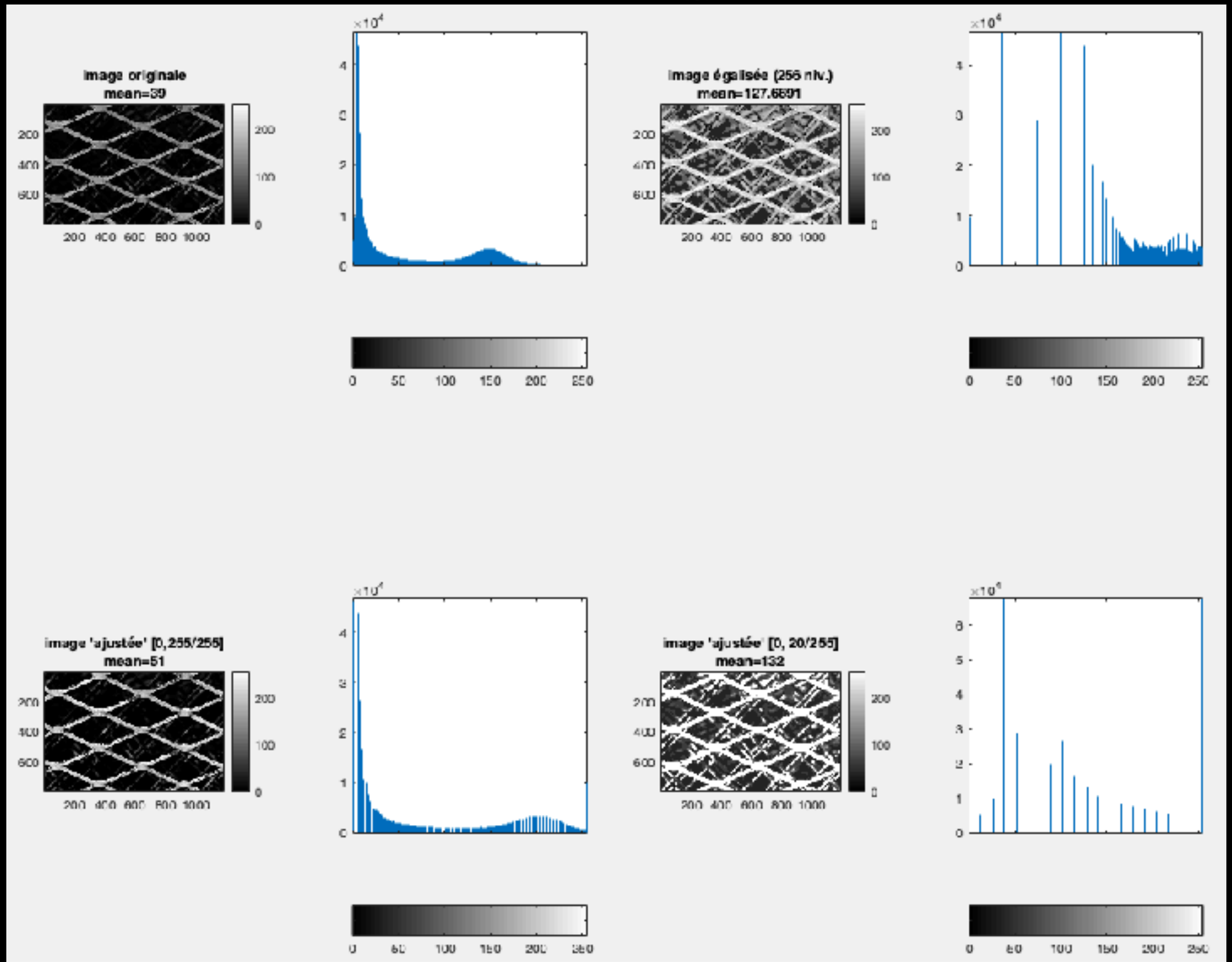
- **EXERCICE 4 : Reprendre l'image du grillages et comparer le résultat de l'égalisation d'histogramme (sur 256 niveaux) à celui d'IMADJUST. Calculer et afficher la valeur moyenne de chaque image et comparer image, histogramme et valeur moyenne de l'image.**

- **EXERCICE 4bis : Toujours en utilisant IMADJUST, adapter la gamme à la première bosse de l'histogramme (celle correspondant aux valeurs de l'intensité entre 0 et 20/255). Que se passe-t-il ?**

# II. Analyse élémentaire d'images

```
clear
close all

% original image
I=imread('/Users/marcel/Documents/MATLAB/GBM/0-images/grillage.jpeg');
I_int=rgb2gray(I);
mean_int = mean(mean(I_int));

figure(1)
subplot(2,4,1), imagesc(I_int), colormap(gray), colorbar, axis('image')
title({'image originale' ; ['mean=',int2str(mean_int)]})
subplot(2,4,2), imhist(I_int, 256), axis('square')

% equalized image (256)
I_eq = histeq(I_int, 256);
mean_eq = mean(mean(I_eq));

subplot(2,4,3), imagesc(I_eq), colormap(gray), colorbar, axis('image')
title({'image égalisée (256 niv.)' ; ['mean=',num2str(mean_eq)]})
subplot(2,4,4), imhist(I_eq, 256), axis('square')

% "adjusted" image
I_adj = imadjust(I_int);
mean_adj = mean(mean(I_adj));

subplot(2,4,5), imagesc(I_adj), colormap(gray), colorbar, axis('image')
title({'image ''ajustée'' [0,255/255]' ; ['mean=',int2str(mean_adj)]})
subplot(2,4,6), imhist(I_adj, 256), axis('square')

% "adjusted" image with selected input range to adjust
I_adj2 = imadjust(I_int, [0., 20./255], []);
mean_adj2 = mean(mean(I_adj2));

subplot(2,4,7), imagesc(I_adj2), colormap(gray), colorbar, axis('image')
title({'image ''ajustée'' [0, 20/255]' ; ['mean=',int2str(mean_adj2)]})

subplot(2,4,8), imhist(I_adj2, 256), axis('square')
```

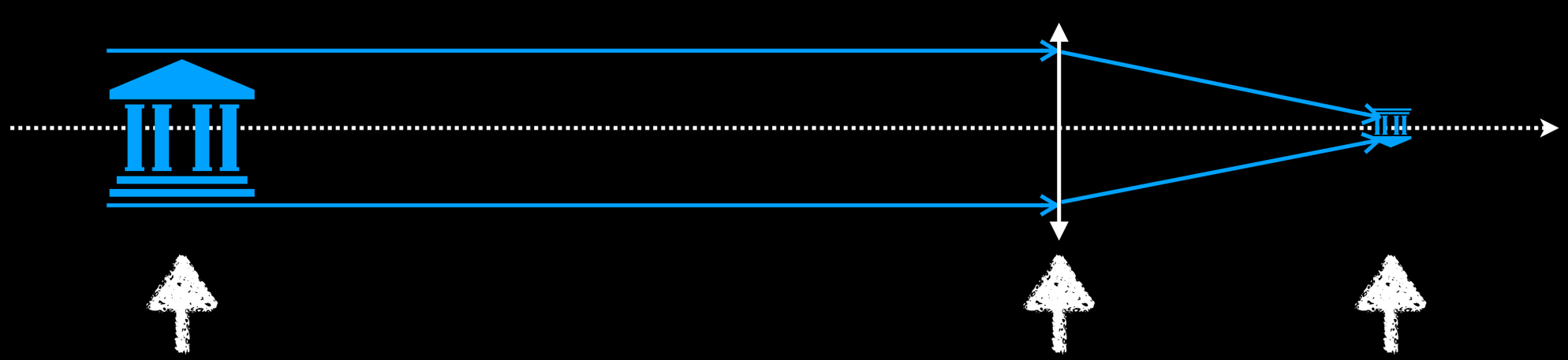
# II. Analyse élémentaire d'images



# II. Analyse élémentaire d'images

**6- AU FAIT, QU'EST-CE QU'UNE IMAGE ?…**

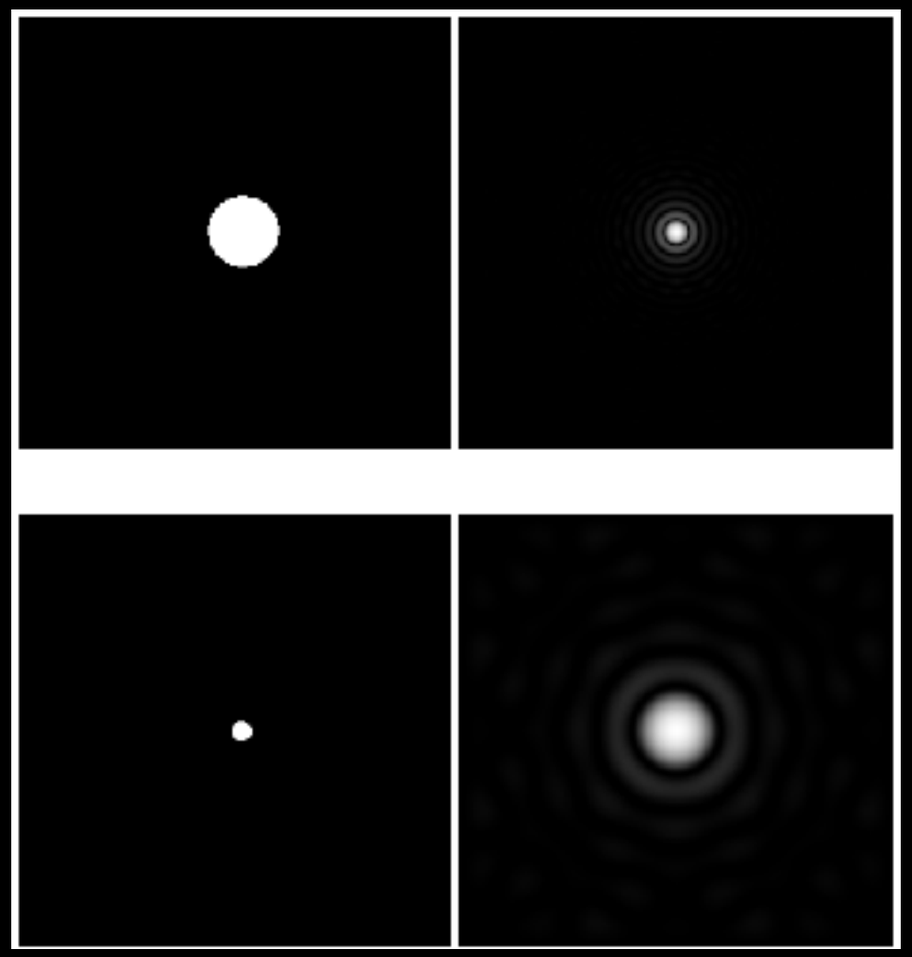
- **<u>Formation d'une image (optique)</u> :**




**L'image vue (=détectée) au foyer est l'image d'un point convoluée par l'objet.**

# II. Analyse élémentaire d'images

**Or : l'image d'un point est une « tache d'Airy », dont le cœur est de largeur à mi-hauteur ~λf/D (ou ~λ/D en unités angulaires), avec λ la longueur d'onde.**



**=> plus D est grand, plus la résolution dans l'image (inversement proportionnelle à λ/D) sera importante (pour une longueur d'onde λ donnée).**

# II. Analyse élémentaire d'images

- **<u>Détection</u> :**

  **L'image précédente est ensuite détectée par un… détecteur, par exemple la matrice CCD (ou CMOS, ou EMCCD, ou autre technologie) qui équipe votre smartphone ou n'importe quel appareil « imageur » plus sophistiqué. Et cette détection est assujettie à plusieurs bruits…**

**- Le <u>bruit de photons</u> (ou *photon noise*, ou *shot noise*) qui suit une distribution de Poisson :**

**p(n) = probabilité de détecter n photons quand N sont attendus**

$$p(n) = \frac{N^n \, e^{-N}}{n!}, \text{ avec } : \sigma^2 = N$$

**sous Matlab/Octave :**
***>> image_phot = imnoise(image_int, 'poisson') ;***

# II. Analyse élémentaire d'images

- **Le <u>bruit de lecture</u> (ou *read-out noise*, *RON*) qui est un bruit ADDITIF caractérisé par une distribution de Gauss (ou 'normale') de moyenne nulle et d'écart-type $\sigma_e$ :**

$$p(x) = \frac{1}{\sigma_e \sqrt{2\pi}} \, e^{-\frac{x^2}{2\sigma_e^2}}$$

**sous Matlab/Octave :**

***>> image_ron = imnoise(image_double, 'gaussian', 0.0, 0.01) ;***

**image en double précision entre 0 et 1**

**moyenne**

**variance relative**

# II. Analyse élémentaire d'images

- **Entre les deux (dans la même chaîne d'acquisition d'images), apparaissent d'<u>autres bruits électroniques</u> :**

  **• pixels « chauds » et « froids » sur le CCD => bruits de type « poivre et sel »**

  **sous Matlab/Octave :**
  ***>> image_snp = imnoise(image_int, 'salt & pepper', d)***
  ***[ATTENTION : 'salt and pepper' sous certaines versions d'Octave…]***

  **densité de bruit**
  **(*ratio* de pixels affectés)**

**• bruit de courant d'obscurité, bruits spécifiques à des détecteurs « exotiques » ou des applications très pointues, etc.**

**• « bruits périodiques » (par.ex. tramage dû à un scan ou une compression jpeg trop forte) => en fait plutôt un BIAIS qu'un véritable bruit (pas aléatoire).**

# II. Analyse élémentaire d’images

- **EXERCICE 5 :**

  **- Prendre l’image « frog » (p.ex.). Appliquer successivement les 3 bruits principaux (bruit de photon, bruit poivre et sel, bruit de lecture), puis représenter les histogrammes consécutifs ET leur différence d’avec le précédent (utiliser PLOT).**

  **- Calculer et afficher minimum, maximum et moyenne de chaque image.**

  **- Calculer et afficher également les distances (au sens des moindres carrés) entre images et histogrammes successifs et leurs précédents.**

  **- Données pour les bruits : *d*=5%, *variance_RON*=1%.**