TP1

Traitement basique d’images

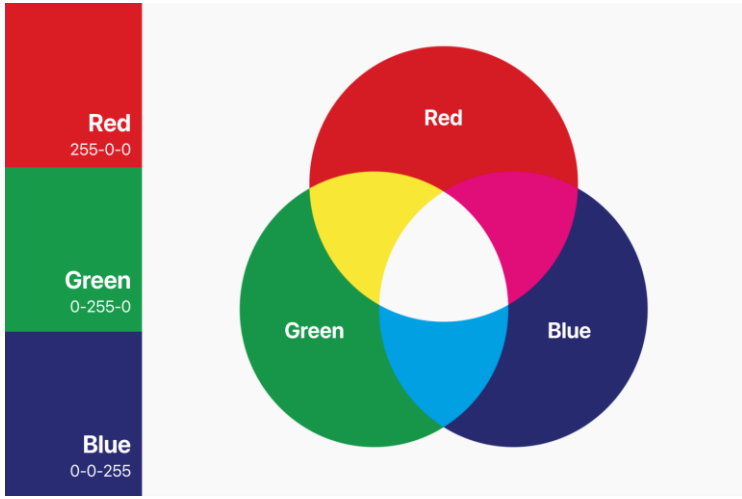
Rappel

Image RGB

Les images RGB sont des images en couleur définies par trois composantes de couleur : Rouge (Red), Vert (Green), et Bleu (Blue). Chaque pixel d'une image RGB est un triplet de valeurs correspondant à l'intensité de ces trois couleurs de base. En combinant ces couleurs de différentes manières, on peut représenter une large gamme de couleurs.

Exemple :

- Un pixel avec des valeurs (255, 0, 0) sera rouge pur.
- Un pixel avec des valeurs (0, 255, 0) sera vert pur.
- Un pixel avec des valeurs (0, 0, 255) sera bleu pur.
- Un pixel avec des valeurs (255, 255, 255) sera blanc (toutes les couleurs à leur intensité maximale).



Rappel

Image en Niveaux de Gris

Les images en niveaux de gris sont des images où chaque pixel est représenté par une seule valeur correspondant à une nuance de gris. Cette valeur va généralement de 0 (noir) à 255 (blanc), avec des valeurs intermédiaires représentant différentes nuances de gris.

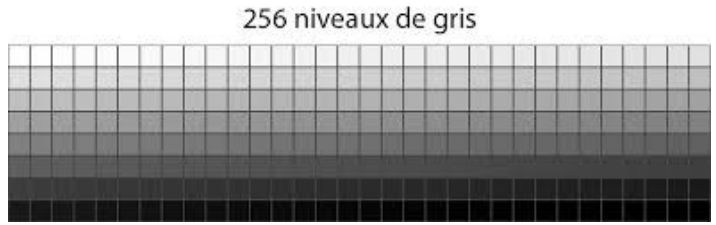
Exemple :

Une image en niveaux de gris peut être vue comme une image en noir et blanc où les variations d'intensité représentent les différents niveaux de gris.

Un pixel avec une valeur de 0 est complètement noir.

Un pixel avec une valeur de 255 est complètement blanc.

Un pixel avec une valeur de 128 sera un gris moyen.



Rappel

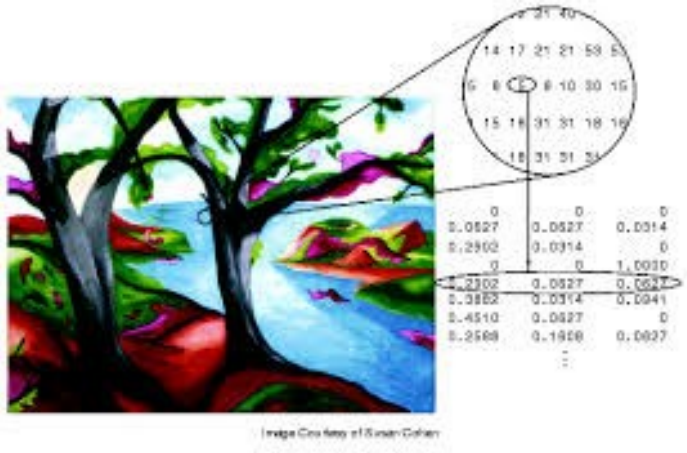
Image Indexée

Les images indexées sont un type d'images où chaque pixel n'est pas stocké avec une valeur de couleur directe, mais plutôt avec un indice dans une table de couleurs (palette). La palette contient un nombre limité de couleurs, et chaque indice dans l'image pointe vers une couleur spécifique dans cette palette.

Exemple :

Supposons une palette de 256 couleurs. Chaque pixel de l'image indexée serait une valeur entre 0 et 255, correspondant à une couleur spécifique dans la palette.

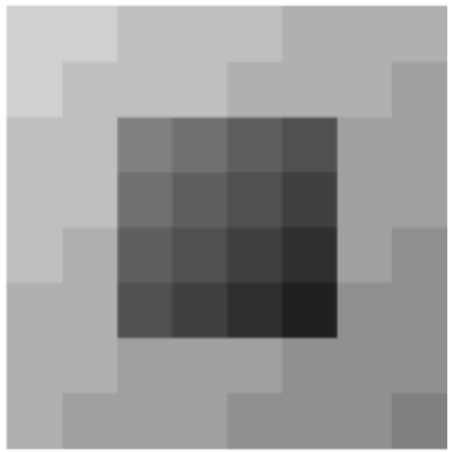
Cela permet de réduire la taille de l'image en stockant les indices plutôt que les valeurs de couleur complètes.



Définition d'un Histogramme d'une Image

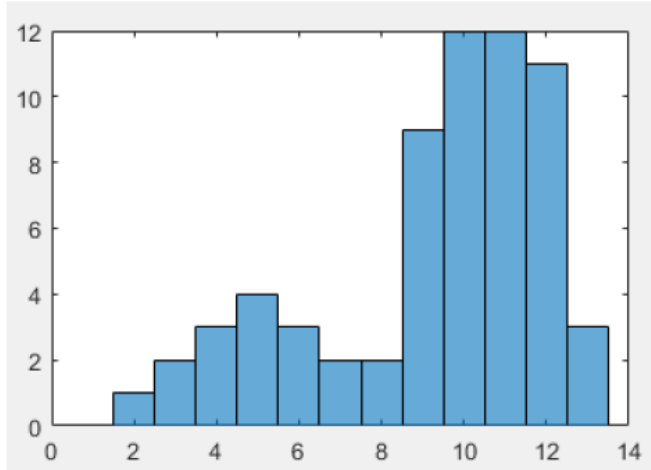
- Un histogramme d'une image est une représentation graphique de la distribution des valeurs des pixels dans cette image. C'est un outil essentiel en traitement d'images pour analyser les caractéristiques de luminosité et de contraste.

Histogramme d'images: Exemple



Les valeurs des pixels

13	13	12	12	12	11	11	11
13	12	12	12	11	11	11	10
12	12	8	7	6	5	10	10
12	12	7	6	5	4	10	10
12	11	6	5	4	3	10	9
11	11	5	4	3	2	9	9
11	11	10	10	10	9	9	9
11	10	10	10	9	9	9	8



Pixel x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
h	0	0	1	2	3	4	3	2	2	9	12	12	11	3

Utilité de l'Histogramme :

L'histogramme est un outil précieux pour :

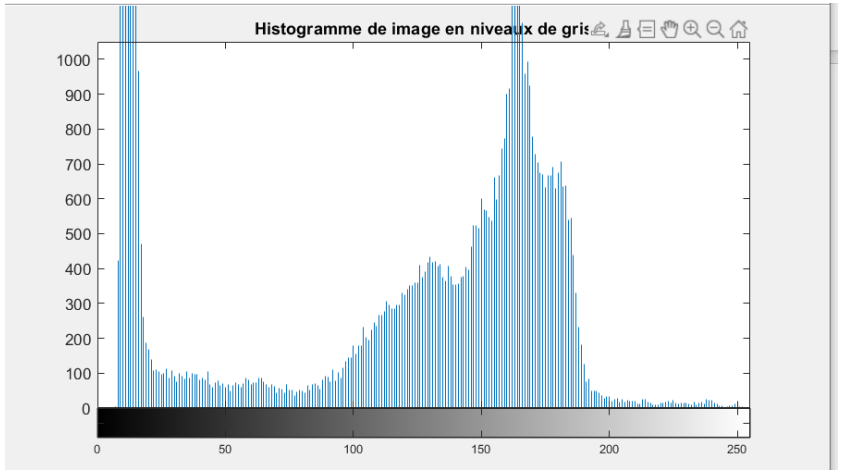
- Analyser le Contraste : Une image avec un faible contraste aura un histogramme concentré dans une petite gamme de valeurs, tandis qu'une image avec un bon contraste aura un histogramme étendu sur toute la plage des valeurs.
- Détecter la Luminosité : Un histogramme décalé vers la gauche indique une image sombre, tandis qu'un histogramme décalé vers la droite indique une image lumineuse.
- Équilibrer les Couleurs : Pour les images en couleur, analyser les histogrammes des différents canaux permet de voir si une couleur est dominante ou si les couleurs sont équilibrées.

Manipulation sur Matlab

- Pour calculer et afficher l'histogramme d'une image sur matlab , on utilise la fonction **imhist**

Exemple 1 : Histogramme d'une Image en Niveaux de Gris

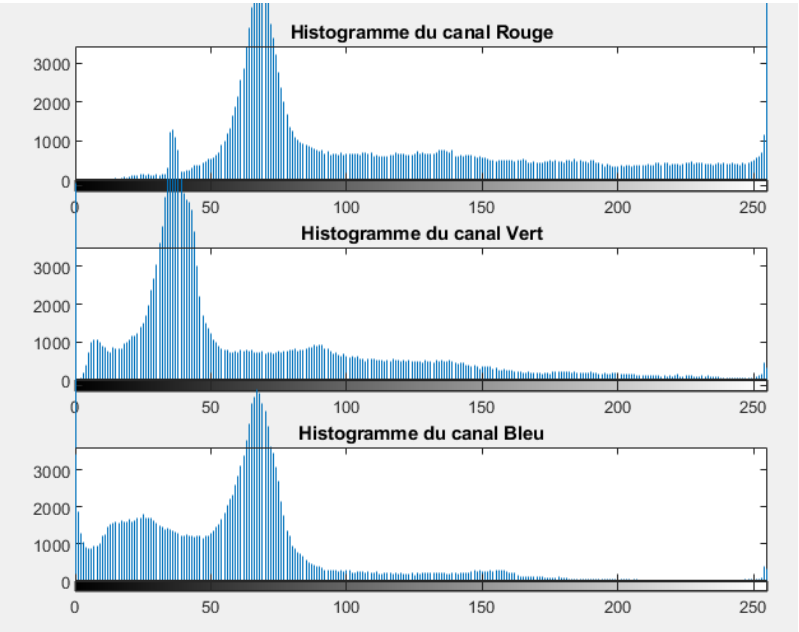
```
I = imread('cameraman.tif');  % Lire une image en niveaux de gris
figure;
imhist(I);  % Afficher l'histogramme
title('Histogramme de image en niveaux de gris');
```



- **Exemple 2 : Histogramme des Canaux d'une Image en Couleur**

```
RGB = imread('peppers.png');  % Lire une image en couleur
R = RGB(:,:,1);  % Extraire le canal rouge
G = RGB(:,:,2);  % Extraire le canal vert
B = RGB(:,:,3);  % Extraire le canal bleu

figure;
subplot(3,1,1); imhist(R); title('Histogramme du canal Rouge');
subplot(3,1,2); imhist(G); title('Histogramme du canal Vert');
subplot(3,1,3); imhist(B); title('Histogramme du canal Bleu');
```



TP1 : Traitements basiques d’images

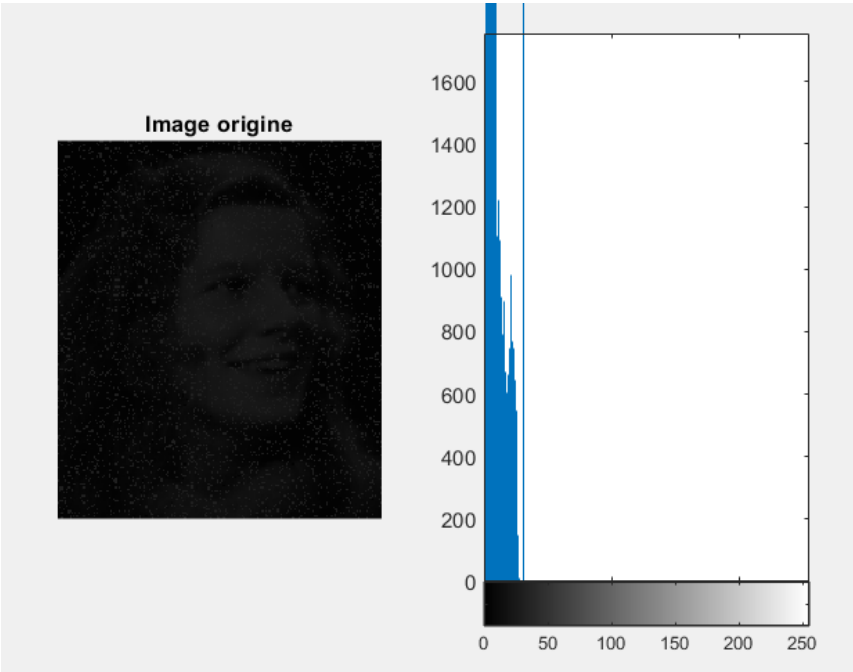
1) Méthodes d’amélioration d’images en niveaux de gris

- On dispose de deux images prises dans des mauvaises conditions d’éclairage: « sosie.png » et « scene.png ».

sosie.png

scene.png

```
img = imread('sosie.png');
figure, subplot(2,2,1), imshow(img), title('Image origine');
subplot(2,2,2), imhist(img);
```



Commentez le résultat !

Expansion dynamique / recadrage dynamique

- L'expansion dynamique est une technique de traitement d'image utilisée pour améliorer le contraste d'une image. Elle consiste à étendre la plage des niveaux de gris ou des valeurs de couleur d'une image afin d'utiliser toute l'étendue disponible des intensités (de 0 à 255 pour des images 8 bits). Cette méthode permet de rendre les détails de l'image plus visibles, surtout si l'image initiale a une plage de niveaux de gris limitée ou concentrée dans une partie étroite du spectre.

Exemple :
Si une image en niveaux de gris a des valeurs de pixel comprises entre 50 et 180, l'expansion dynamique va étirer ces valeurs pour qu'elles couvrent la plage entière de 0 à 255.

Principe de l'Expansion Dynamique:

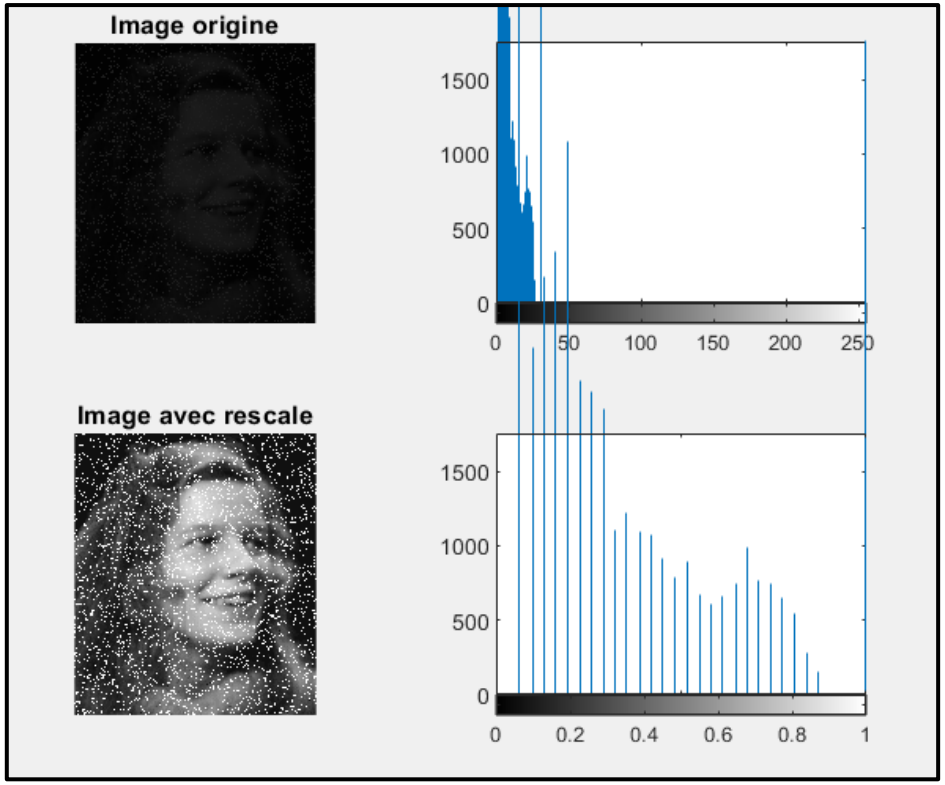
- Le principe de l'expansion dynamique est de prendre les valeurs minimales et maximales de l'intensité des pixels dans une image et de les étirer pour qu'elles couvrent toute la plage possible des niveaux de gris. Cette opération augmente le contraste de l'image, en particulier dans les régions où les différences d'intensité étaient auparavant difficiles à distinguer.

Rescale

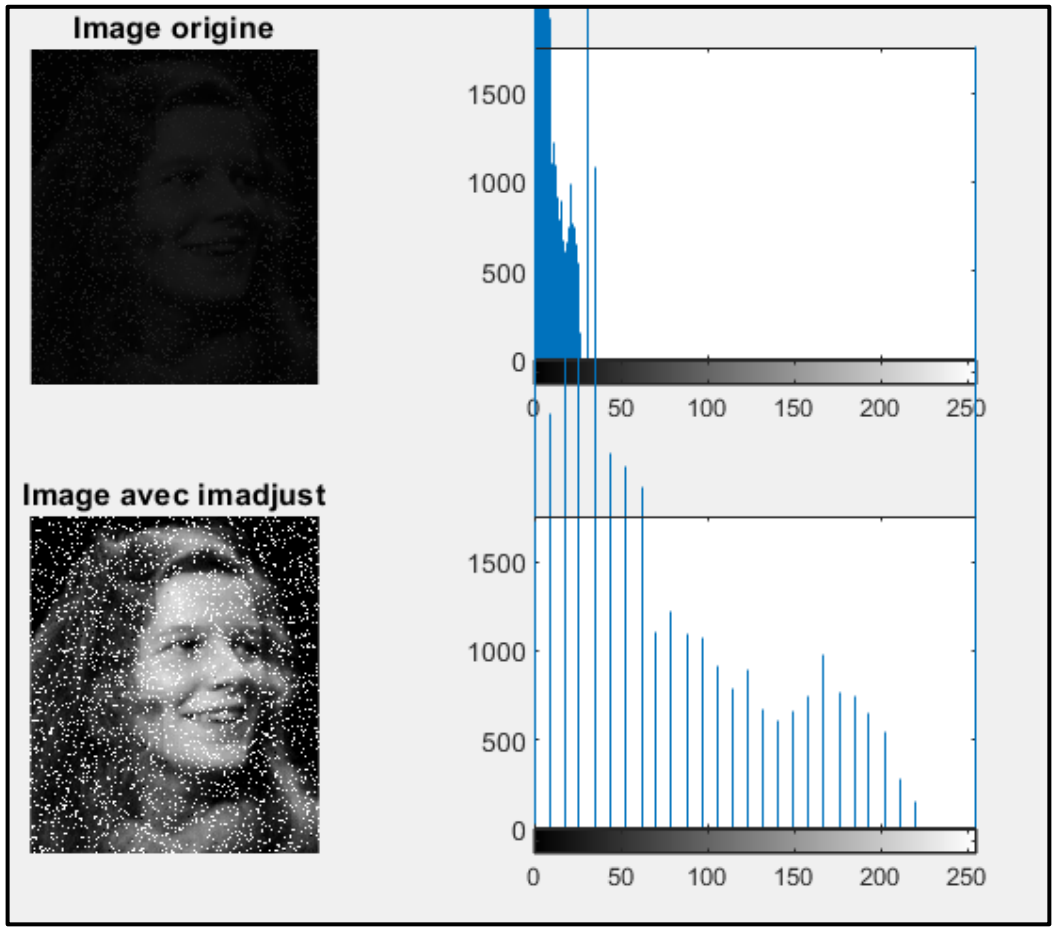
La fonction **rescale** redimensionne les valeurs d'une matrice ou d'un tableau pour qu'elles se situent entre une valeur minimale et une valeur maximale spécifiées. Par défaut, elle redimensionne les valeurs pour qu'elles se situent entre 0 et 1.

```
%Expansion dynamique, recadrage dynamique

imgRescale = rescale(img);
subplot(2,2,3), imshow(imgRescale), title('Image avec rescale');
subplot(2,2,4), imhist(imgRescale);
```

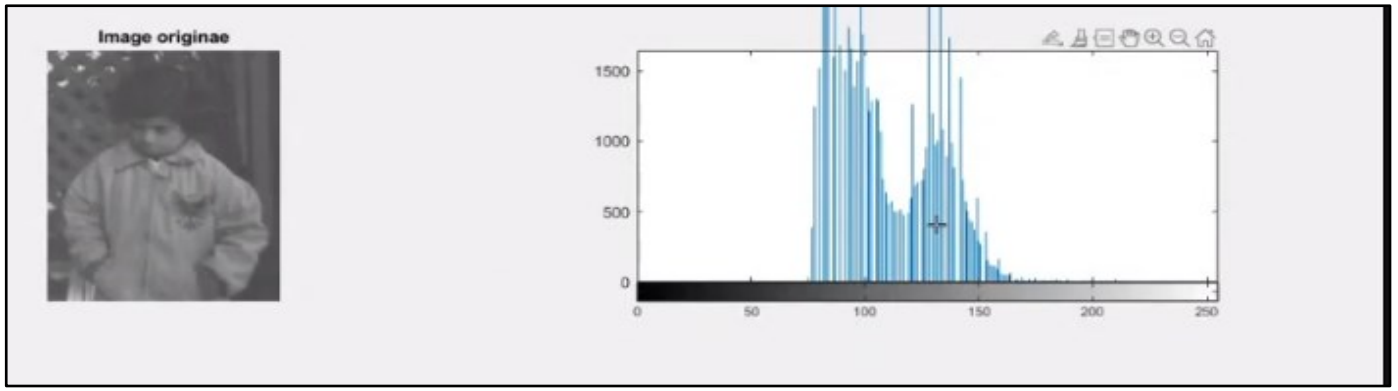


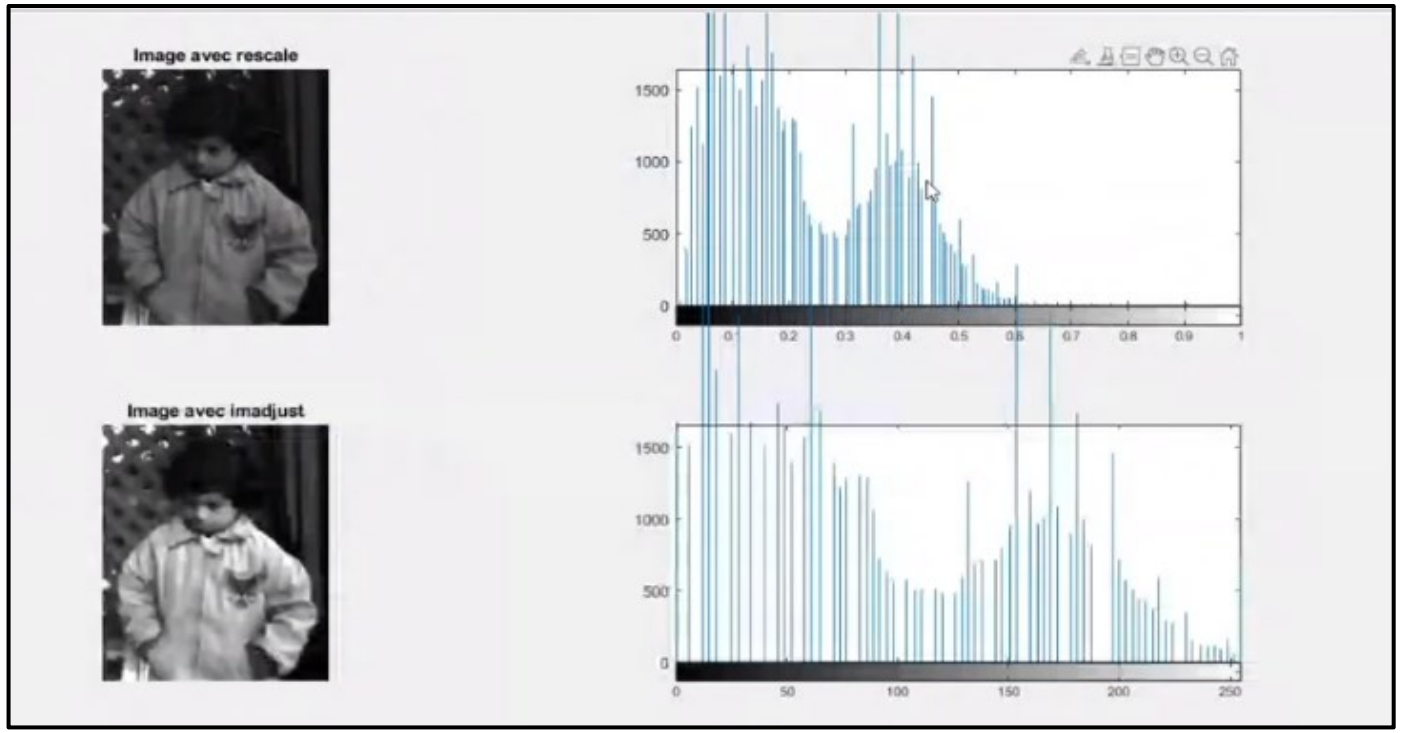
Avec la fonction **imadjust**



La fonction **imadjust** en MATLAB est utilisée pour ajuster les niveaux d'intensité d'une image, permettant ainsi d'améliorer son contraste. Cette fonction est particulièrement utile pour manipuler la gamme de valeurs des pixels de manière à obtenir une meilleure visibilité des détails dans une image.

Un autre exemple





imadjust est souvent utilisé pour des améliorations spécifiques du contraste, des ajustements de la gamme dynamique et des corrections de luminosité dans le traitement d'images.

rescale est généralement utilisé pour normaliser les valeurs d'intensité des pixels dans une plage spécifique avant d'effectuer d'autres opérations sur l'image.

1.2 Egalisation d'histogramme

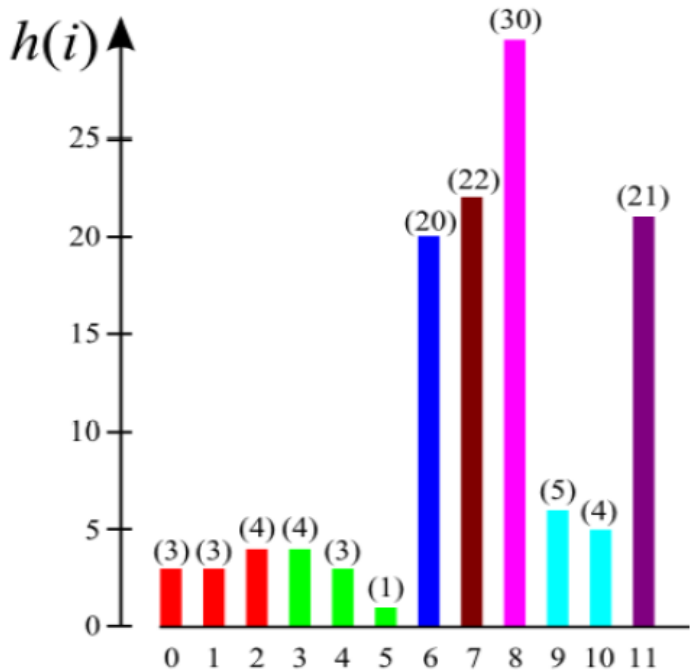
On désire appliquer sur les mêmes images, la méthode d'égalisation d'histogramme en utilisant la fonction ***histeq.*** Il est demandé aussi de visualiser les histogrammes avant et après traitement. Commentez l'effet de l'opération d'égalisation.
Comparez les résultats des deux méthodes sur chacune des images.

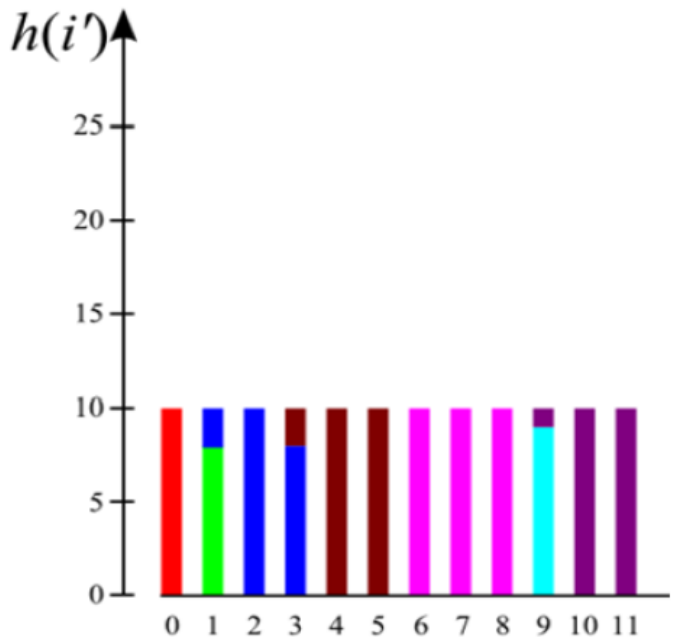
L'égalisation d'histogramme est une technique de traitement d'image utilisée pour améliorer le contraste des images en redistribuant les intensités des pixels de manière à égaliser l'histogramme de l'image. L'objectif est de rendre l'histogramme de l'image aussi uniforme que possible sur toute la plage dynamique, ce qui permet d'obtenir une meilleure représentation visuelle des données.

Egalisation d'histogramme

On cherche une fonction de transformation $t : i \mapsto i'$

- croissante (i.e. préservant l'ordre des niveaux de gris)
- qui génère (autant que possible) un histogramme h(I') «plat», c'est-à-dire une distribution uniforme des niveaux de gris





```
% Charger les images
image1 = imread('sosie.png');
image2 = imread('scene.png');

% Calculer les histogrammes avant l'égalisation
hist_orig_1 = imhist(image1);
hist_orig_2 = imhist(image2);

% Appliquer l'égalisation d'histogramme
eq_image1 = histeq(image1);
eq_image2 = histeq(image2);

% Calculer les histogrammes après l'égalisation
hist_eq_1 = imhist(eq_image1);
hist_eq_2 = imhist(eq_image2);

% Afficher les résultats
figure;
subplot(2, 3, 1), imshow(image1), title('Image 1 Originale');
subplot(2, 3, 2), plot(hist_orig_1), title('Histogramme Image 1 Avant');
subplot(2, 3, 3), imshow(eq_image1), title('Image 1 après Égalisation');
subplot(2, 3, 4), imshow(image2), title('Image 2 Originale');
subplot(2, 3, 5), plot(hist_orig_2), title('Histogramme Image 2 Avant');
subplot(2, 3, 6), imshow(eq_image2), title('Image 2 après Égalisation');

% Afficher les histogrammes après égalisation
figure;
subplot(2, 2, 1), plot(hist_eq_1), title('Histogramme Image 1 Après');
subplot(2, 2, 2), plot(hist_eq_2), title('Histogramme Image 2 Après');
```

Image 1 Originale

Histogramme Image 1 Avant

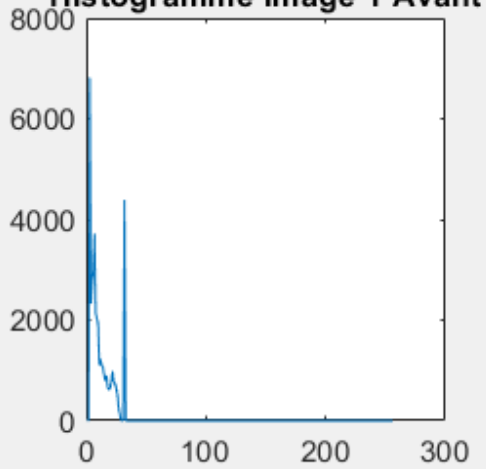


Image 1 après Égalisation

Image 2 Originale

Histogramme Image 2 Avant

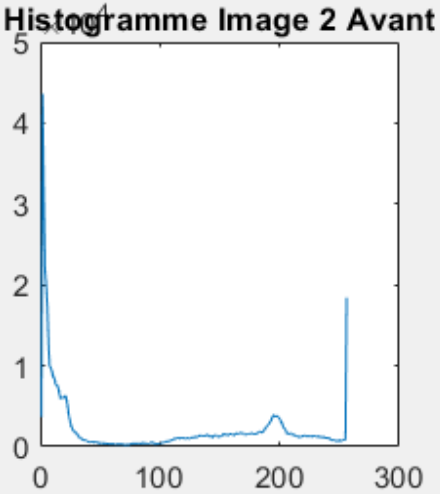


Image 2 après Égalisation


Histogramme Image 1 Après
8000
6000
4000
2000
0
0
100
200
300
Histogramme Image 2 Après
×10^4
4
3
2
1
0
0
100
200
300

- Qu'est-ce que **imbinarize** ?
- Imaginez que vous avez une photo en noir et blanc, comme une page de livre. Certains pixels de cette photo sont plus clairs (gris clair ou blanc) et d'autres sont plus sombres (gris foncé ou noir). La fonction imbinarize prend cette photo et la simplifie en transformant chaque pixel en noir ou en blanc. Cela peut aider à faire ressortir des choses importantes comme du texte ou des objets.