

Plan

- Vision artificielle
- Introduction à l'imagerie numérique.
- Les types d'images.
- **Prétraitement des Images**
- Caractérisation de Texture.
- Extraction Avancée de Caractéristiques.
- Détection et Reconnaissance d'Objets.
- L'Apprentissage Automatique pour la Classification des Images
 - Classification avec les algorithmes d'apprentissage (linéaires et non-linéaire).
 - Introduction aux réseaux de neurones convolutifs (CNN).
- Présentation des Projets et Révisions

1. Le contraste

Le contraste d'une image est une mesure de la différence d'intensité entre les pixels les plus clairs et les plus sombres de l'image.

Valeur proche de 0 :

- Description : Lorsque le contraste est proche de 0, cela signifie que la différence entre l'intensité maximale et minimale des pixels est faible par rapport à leur somme.
- Implication Visuelle : L'image apparaîtra plutôt plate et uniforme, sans variations significatives entre les zones sombres et claires. En d'autres termes, elle manque de dynamisme et de distinction entre les objets.

Valeur proche de 1 :

- Description : Lorsque le contraste est proche de 1, cela signifie que la différence entre l'intensité maximale et minimale des pixels est presque égale à leur somme.
- Implication Visuelle : L'image aura des variations significatives entre les zones sombres et claires. Les objets dans l'image seront bien définis, avec une bonne distinction entre les différents éléments.

$$C = \frac{\max[f(x, y)] - \min[f(x, y)]}{\max[f(x, y)] + \min[f(x, y)]}$$

Calculer le contraste

```
min_intensity = np.min(image)
```

```
max_intensity = np.max(image)
```

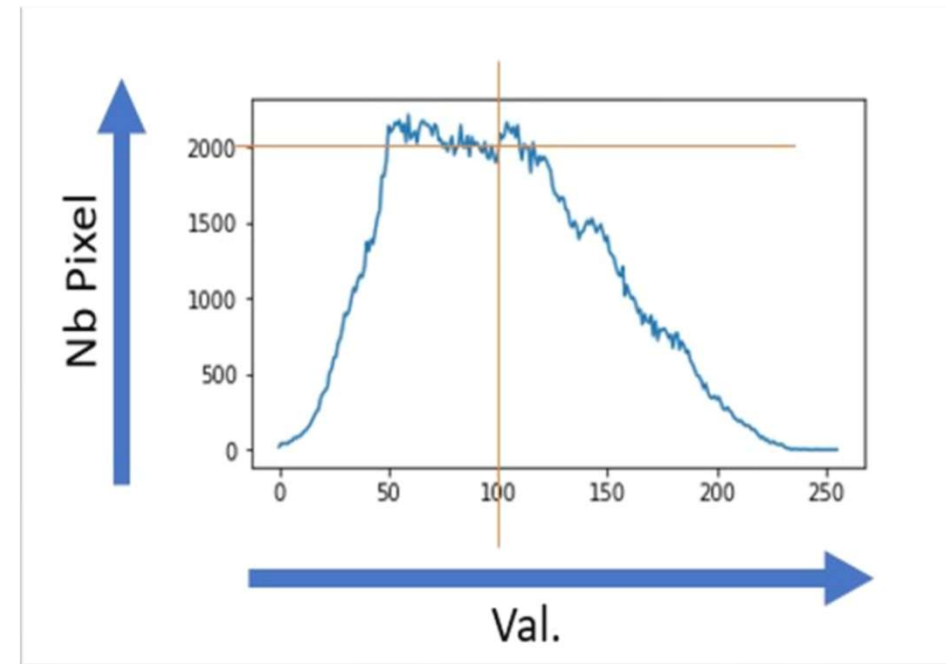
```
contrast = (max_intensity - min_intensity) / (max_intensity + min_intensity)
```

```
print(f'Contraste : {contrast}')
```

2. Les Histogrammes

Un histogramme d'image est un graphique qui affiche :

- En abscisse (Val. dans le graphe ci-dessous) les différentes valeurs de canaux/pixel
 - En ordonnée (Nb Pixel) le nombre de canaux/pixel qui possèdent cette valeur
- Visualisation de la Distribution des Intensités
 - Amélioration du Contraste
 - Détection de Problèmes d'Exposition (beaucoup de pixels sombres ou clairs)
 - Segmentation d'Image (binariser l'image pour isoler des objets ou des régions d'intérêt)
 - Normalisation et Étalonnage
 - Compression d'Images
 - Analyse de Texture pour la reconnaissance et la classification d'image
 - Mesure de la Similarité (recherche d'images ou de reconnaissance.)
 -



2. Les Histogrammes

Analyse de l'Image

1. Distribution des Intensités :

1. L'histogramme montre une concentration d'intensités dans une certaine plage. Cela indique que l'image contient principalement des niveaux de gris situés dans cette plage.
2. Une large concentration autour de valeurs moyennes (ex. 50-150) peut indiquer une bonne exposition de l'image.

2. Détection des Problèmes d'Exposition :

1. Si l'histogramme est trop concentré à gauche (vers les valeurs faibles), l'image peut être sous-exposée.
2. Si l'histogramme est trop concentré à droite (vers les valeurs élevées), l'image peut être surexposée.

3. Contraste :

1. Un histogramme étalé sur toute la plage de valeurs (0-255) indique un bon contraste.
2. Un histogramme concentré dans une plage étroite indique un faible contraste.

2. Les Histogrammes

Conclusions à partir de l'Analyse :

1.Exposition de l'Image :

1. Si l'histogramme est bien réparti, l'image est probablement **bien exposée**.
2. S'il est concentré à une extrémité, des **ajustements** de l'exposition peuvent être nécessaires.

2.Contraste de l'Image :

1. Un histogramme étalé suggère un bon contraste, rendant **les détails de l'image plus visibles**.
2. Un histogramme **étroit** suggère **un faible contraste**, et l'égalisation d'histogramme pourrait être appliquée pour améliorer le contraste.

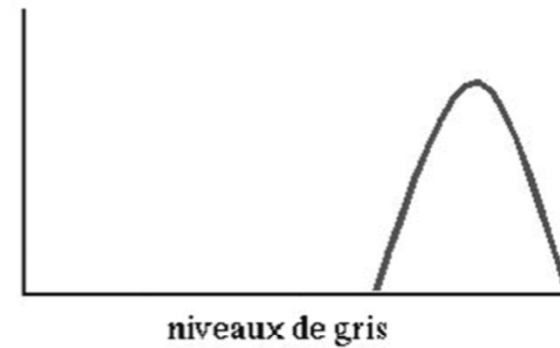
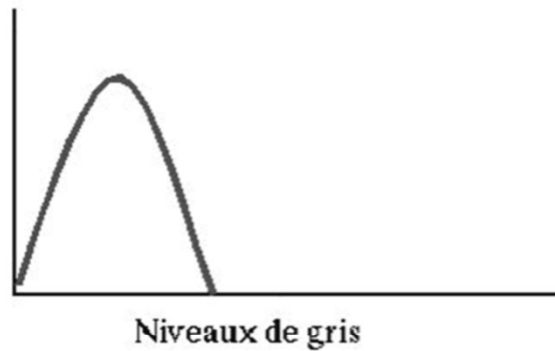
1.Segmentation et Analyse de Contenu :

1. Les **pics** dans l'histogramme peuvent représenter des **objets** ou des **régions** distinctes dans l'image.
2. La segmentation basée sur l'histogramme peut être utilisée pour **isoler** ces objets.

2. Les Histogrammes

Que peut-on dire des images dont les histogrammes sont représentés ci-dessous ?

Luminance



Contraste



2. Les Histogrammes

SCÈNE DE FAIBLE LUMINANCE



Niveaux de gris

SCÈNE DE FORTE LUMINANCE

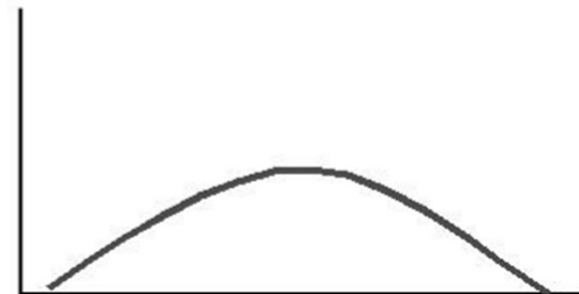


niveaux de gris

SCÈNE DE BAS CONTRASTE



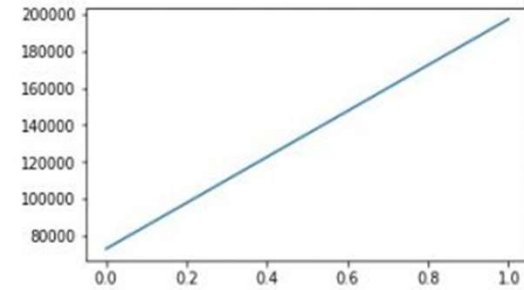
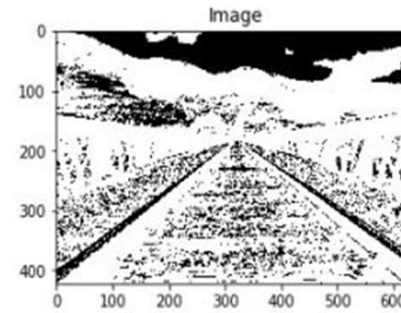
SCÈNE DE HAUT CONTRASTE



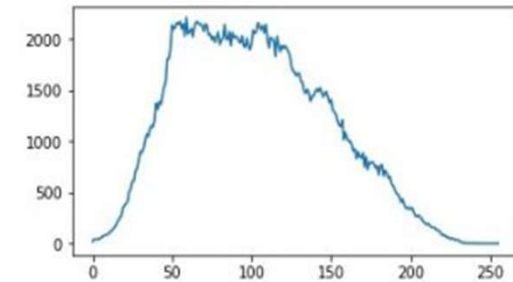
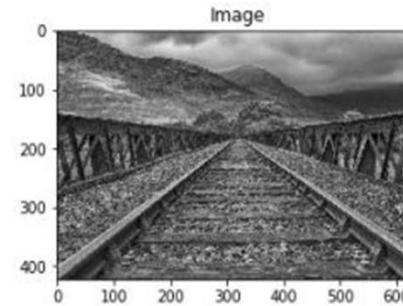
Prétraitement des Images

2. Les Histogrammes

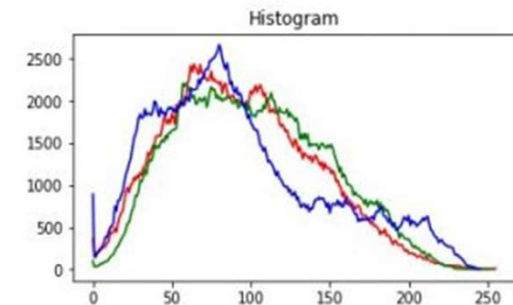
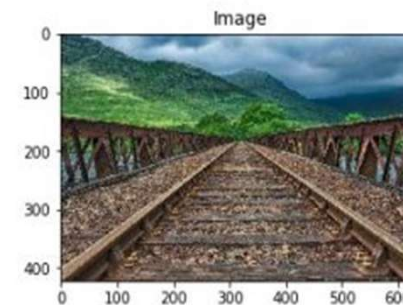
Une image en noir et Blanc a un histogramme très basique (binaire) qui ne comporte que des valeurs 0 ou 1 (pas de nuances sur 24 bits par exemple) :



Une image en niveau de gris ne possède qu'une seule courbe (celle des nuances de gris):



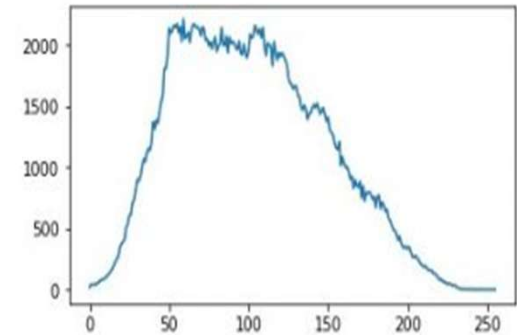
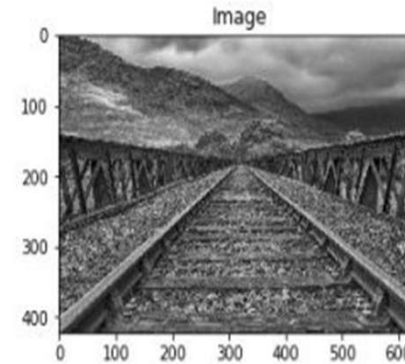
Si on a une image en couleur il nous faut maintenant avoir les nuances sur les trois canaux (Rouge, Vert et Bleu). On a donc 3 courbes :



2. Les Histogrammes

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

image = cv2.imread('1.jpg', cv2.IMREAD_GRAYSCALE)
histogram = cv2.calcHist([image], [0], None, [256], [0, 256])
plt.figure(figsize=(12, 6))
plt.subplot(121)
plt.imshow(image, cmap='gray')
plt.title('Image en niveaux de gris')
plt.subplot(122)
plt.plot(histogram)
plt.title('Histogramme')
plt.xlabel('Intensité des pixels')
plt.ylabel('Nombre de pixels')
plt.show()
```



2. Les Histogrammes

```
image = cv2.imread('-----')

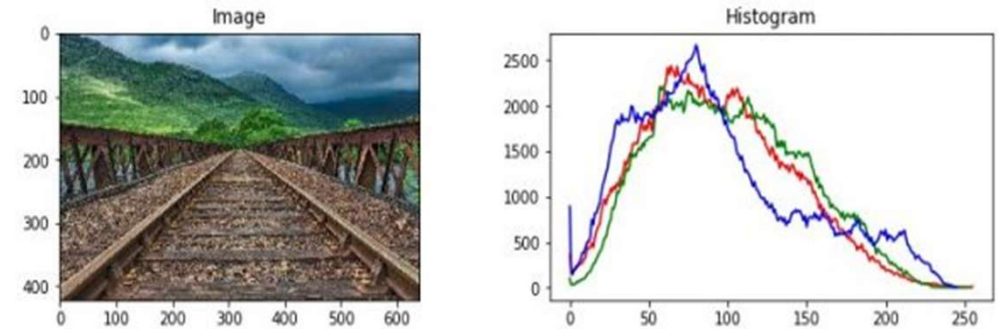
# Séparer les canaux de couleur
colors = ('r', 'g', 'b')

# Afficher l'image et les histogrammes pour chaque canal
plt.figure(figsize=(12, 6))

plt.subplot(131)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Image en couleur')

plt.subplot(132)
for i, color in enumerate(colors):
    histogram = cv2.calcHist([image], [i], None, [256], [0, 256])
    plt.plot(histogram, color=color)
    plt.xlim([0, 256])

plt.title('Histogramme')
plt.xlabel('Intensité des pixels')
plt.ylabel('Nombre de pixels')
```



```
histogram_red = cv2.calcHist([image], [0], None, [256], [0, 256])
plt.plot(histogram_red, color='r')
plt.xlim([100, 150])
plt.title('Histogramme Rouge')
plt.xlabel('Intensité des pixels')
plt.ylabel('Nombre de pixels')
```

3. Techniques de Lissage

Le filtrage: Le principe du filtrage est de **modifier** la valeur des **pixels** d'une image, généralement dans le but **d'améliorer** son aspect. En pratique, il s'agit de créer une nouvelle image en se servant des valeurs des pixels de l'image d'origine.

Le lissage d'images vise à réduire le bruit et les variations rapides dans une image. Il est souvent utilisé pour :

- Réduire le bruit
- Atténuer les détails fins
- Préparer l'image pour une analyse ultérieure

Techniques de lissage :

- **Filtre moyenneur** : Chaque pixel de l'image est remplacé par la moyenne des pixels voisins. Cela a pour effet de réduire les variations brusques de l'image.
- **Filtre gaussien** : Utilise une fonction gaussienne pour effectuer une convolution sur l'image. Ce type de filtre est également utilisé pour flouter l'image de manière plus naturelle par rapport au filtre moyenneur.

3. Techniques de Lissage

Filtre moyennneur : Chaque pixel de l'image est remplacé par la moyenne des pixels voisins. Cela a pour effet de réduire les variations brusques de l'image.

Ce filtre permet de :


- Lisser l'image (effet de flou)
- Réduire le bruit
- Réduire les détails

Pour un filtre de taille $n = 3$, le noyau de convolution est donné par :

$$h = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

D'une manière générale, si on a un filtre de taille n , tous les coefficients du filtre ont comme valeur $w_i = \frac{1}{n^2}$. Plus n est grand, plus le lissage sera important, et plus l'image filtrée perd les détails de l'image origi

24	32	128	240	255
12	42	111	154	222
4	23	123	176	243
15	63	145	134	172
27	12	98	75	143



		108		

3. Techniques de Lissage

Filtrage Gaussien (Gaussian Filtering):

Le filtrage gaussien de l'image résulte de la convolution de cette fonction avec des gaussiennes en chaque point de l'image.

Un inconvénient évident du filtrage gaussien est qu'il ne **lisse** pas **uniquement** le **bruit**, mais il **gomme** aussi les **contours**, les rendant difficilement identifiables.

Le filtre gaussien donnera un meilleur lissage et une meilleure réduction du bruit que le filtre moyenne.

Le noyau gaussien centré et d'écart-type σ est défini par :

$$h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

La largeur du filtre est donnée par son écart-type σ :

- Largeur du filtre de part et d'autre du point central : $\text{Ent}+(3 \sigma)$ ($\text{Ent}+(.)$ est l'entier supérieur)
- Largeur totale du filtre : $2\text{Ent}+(3 \sigma) + 1$
- Si σ est plus petit qu'un pixel le lissage n'a presque pas d'effet
- Plus σ est grand, plus on réduit le bruit, mais plus l'image filtrée est floue
- Si σ est choisi trop grand, tous les détails de l'image sont perdus

On doit trouver un compromis entre la quantité de bruit à enlever et la qualité de l'image en sortie disparaissent.

Exemple pour $\sigma = 1.41$

Largeur du filtre de part et d'autre du point central : $\text{Ent}+(3 \sigma) = 5$

Largeur totale du filtre : $2\text{Ent}+(3 \sigma) + 1 = 11$

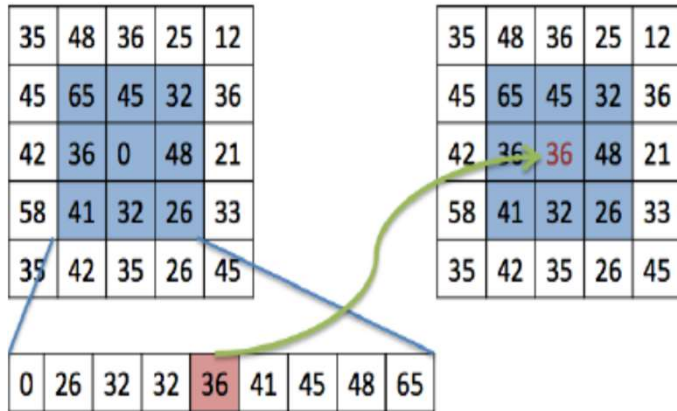
On obtient le filtre suivant :

$$h = \frac{1}{864} \begin{pmatrix} 11 & 23 & 29 & 23 & 11 \\ 23 & 48 & 62 & 48 & 23 \\ 29 & 62 & 82 & 62 & 29 \\ 23 & 48 & 62 & 48 & 23 \\ 11 & 23 & 29 & 23 & 11 \end{pmatrix}$$

3. Techniques de Lissage

Filtrage Médian (Median Filtering) : Remplacer le pixel central (sur lequel est positionnée la fenêtre) par la valeur médiane des pixels inclus dans la fenêtre.

- Très adapté au bruit type "poivre et sel" (faux "blanc" et "noir" dans l'image)
- Préserve les contours
- Réduit le bruit additif uniforme ou gaussien (lissage de l'image)
- Si le bruit est supérieur à la moitié de la taille du filtre, alors le filtre est inefficace



```
import numpy as np
kernel = np.array([65, 45, 32, 36, 0, 48, 41, 32, 26])
sorted_kernel = np.sort(kernel)
median_value = np.median(sorted_kernel)
print("Les valeurs du noyau triées : ", sorted_kernel)
print("La médiane est : ", median_value)
```

```
median_filtered_image = cv2.medianBlur(image, 3)
```

3. Techniques de Lissage

Filtrage Gaussien (Gaussian Filtering):

```
image = cv2.imread('votre_image.jpg', cv2.IMREAD_GRAYSCALE)

# Ajouter du bruit gaussien à l'image
row, col = image.shape
mean = 0
sigma = 25
gauss = np.random.normal(mean, sigma, (row, col))
noisy_image = image + gauss

# Appliquer le filtre gaussien
gaussian_filtered = cv2.GaussianBlur(noisy_image, (5, 5), 0)
```

4. Binarisation

L'objectif principal de la binarisation est de simplifier l'image pour faciliter son analyse et son traitement ultérieurs. En convertissant une image en une forme binaire, on peut extraire des informations importantes telles que les **contours**, les **formes**, et les **régions** d'intérêt, tout en éliminant les détails superflus et le bruit.

Applications de la binarisation

- Utilisée dans les systèmes de reconnaissance optique de caractères (OCR) pour segmenter les caractères des arrière-plans.
- Utilisée pour détecter et analyser les formes géométriques dans les images.
- Utilisée pour extraire les contours des objets dans une image, facilitant ainsi la segmentation et l'identification des objets.
- Utilisée pour séparer les objets d'intérêt du fond de l'image.

4. Binarisation

Méthodes de binarisation

- 1- Seuil fixe :
- Un seuil constant est choisi manuellement pour toute l'image.
 - Simple à implémenter mais peut ne pas fonctionner bien si l'éclairage dans l'image est non uniforme.

```
_, binary_image = cv2.threshold(image, threshold_value, 255, cv2.THRESH_BINARY)
```

- 2- Seuil adaptatif : Le seuil est calculé localement pour différentes régions de l'image.
Plus efficace pour les images avec des variations d'éclairage.

```
binary_image = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, block_size, C)
```

- 3- Méthode d'Otsu : Un seuil est automatiquement calculé en minimisant l'intra-classe de la variance.

```
_, binary_image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

4. Binarisation

Méthodes de binarisation

Image Originale



Image Binarisée



Image Originale



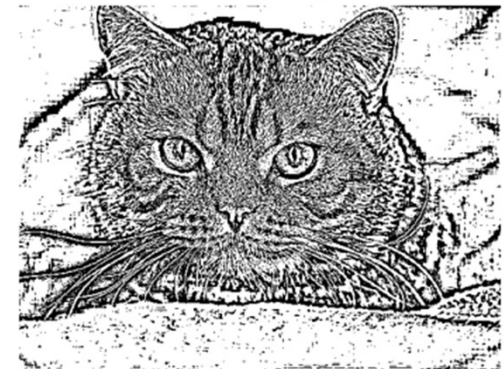
Binarisation Seuil Fixe



Binarisation Otsu (Seuil: 128)



Binarisation Adaptative



4. Binarisation

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
# Matrice de l'image plus claire
image = cv2.imread('1.jpg', cv2.IMREAD_GRAYSCALE)

# Binarisation
threshold = 128 # Choisir un seuil
_, binary_image_fixed = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY)
_, binary_image_otsu = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
binary_image_adaptive = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
plt.figure(figsize=(15, 5))
plt.subplot(131)
plt.imshow(binary_image_fixed, cmap='gray', vmin=0, vmax=255)
plt.title('Binarisation Seuil Fixe')
plt.axis('off')
plt.subplot(132)
plt.imshow(binary_image_otsu, cmap='gray', vmin=0, vmax=255)
plt.title(f'Binarisation Otsu (Seuil: {threshold})')
plt.subplot(133)
plt.imshow(binary_image_adaptive, cmap='gray', vmin=0, vmax=255)
plt.title('Binarisation Adaptative')
plt.show()
```

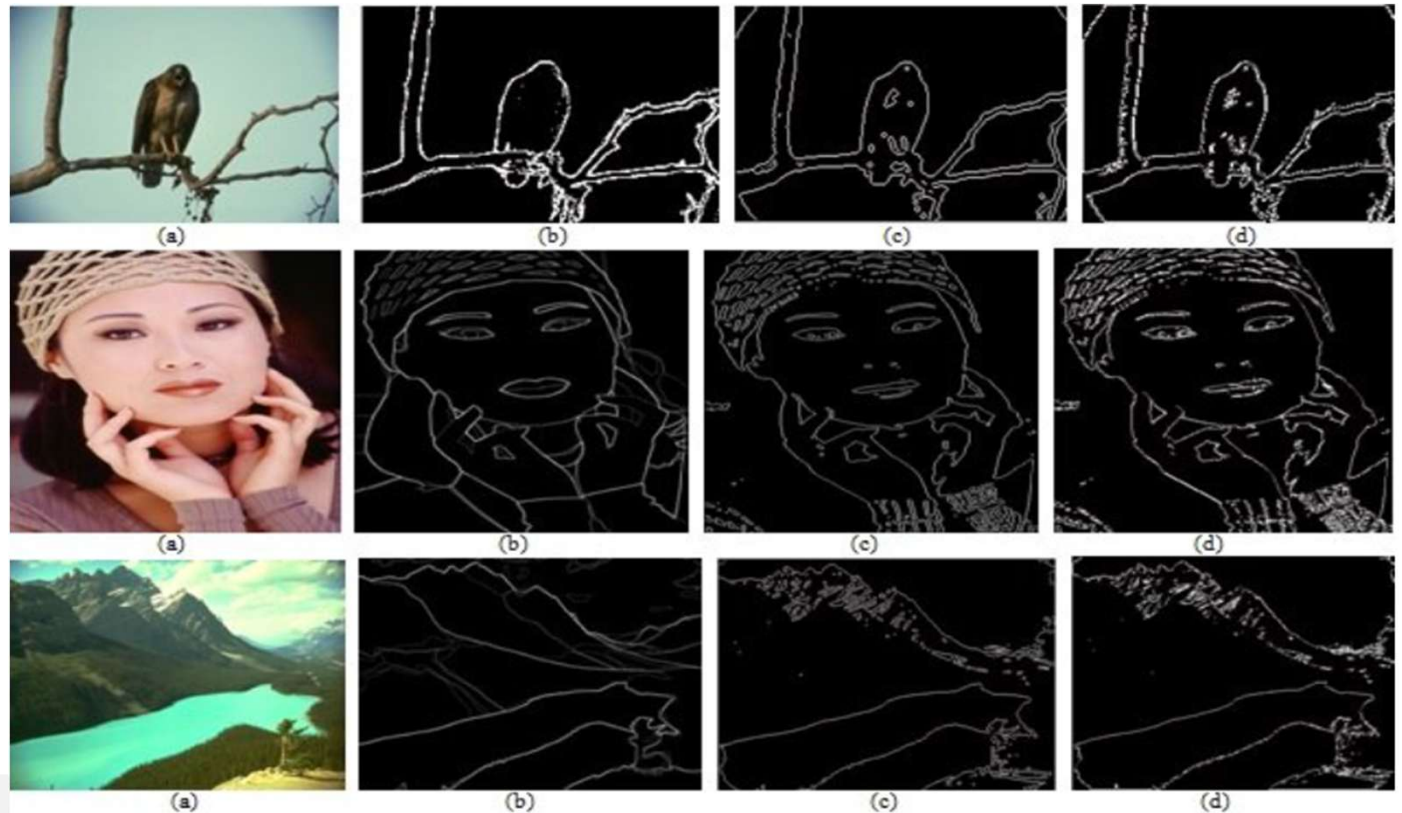
6. Les contours

On appelle détection de contours les procédés permettant de repérer les points d'une image matricielle qui correspondent à un changement brutal de l'intensité lumineuse. Ces changements de propriétés de l'image numérique indiquent en général des éléments importants de structure dans l'objet représenté.

Un **contour** dans une image représente la **frontière** ou le **bord** d'un objet ou d'une forme détectée dans l'image.

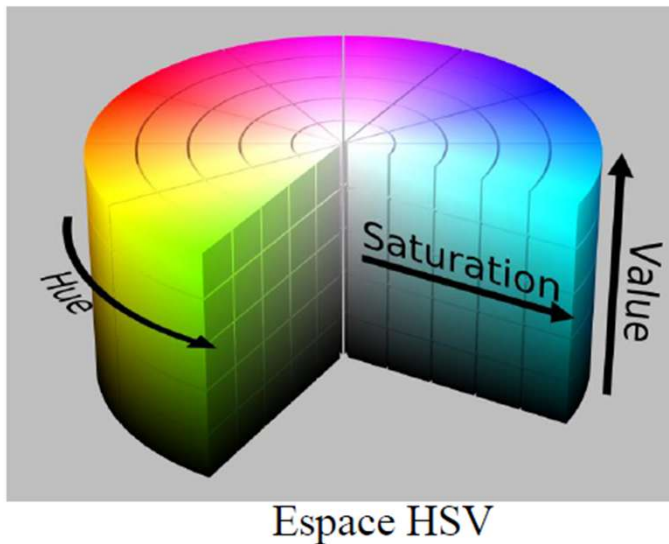
Un **contour** est une **courbe fermée** qui relie des points adjacents ayant la même intensité.

OpenCV: `cv2.findContours()`



7. HSV : Hue (H), Saturation (S), Value (V)

- Séparation de la **teinte**, de la **saturation** et de **l'intensité**
 - Plus intuitif pour identifier et spécifier les couleurs
 - Traitement des ombrages plus facile lors de la segmentation
 - Moins sensible aux variations de lumière



- l'espace HSV est essentiel pour la vision par ordinateur, car il permet une analyse plus précise et plus robuste des couleurs, facilitant ainsi de nombreuses applications dans la détection et la reconnaissance d'objets.

- Séparation claire de la teinte, de la saturation et de la luminosité
- Meilleure gestion des ombrages et des variations d'éclairage
- Segmentation plus intuitive et précise des couleurs.
- Dans l'espace HSV, les **couleurs similaires** ont des valeurs de teinte proches, ce qui rend la **segmentation** des couleurs plus simple et plus intuitive.

8. Luv : Luminosité (L), Cordonnées (u v) Teinte(H) et saturation(S)



Merci de votre attention

Benfriha Hichem



hbenfriha@Teccart.com

