Technique de base

Autor: Cristopher McIntyre Garcia

Mini-Cours

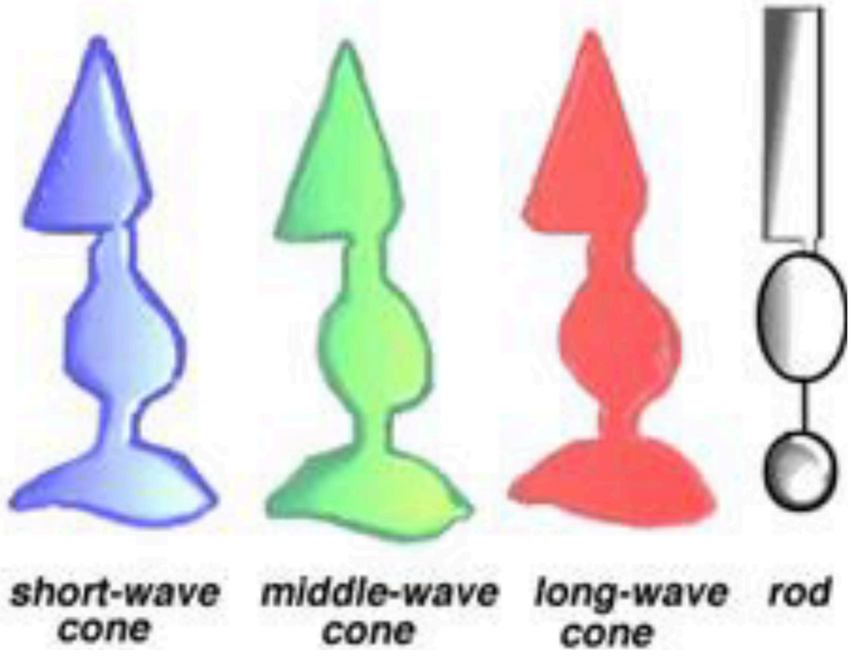
- ~~Jour 1: Introduction à l’image digital~~
- Jour 2: Techniques de base de la vision artificiel et au traitement d’image
- Jour 3: Filtre et technique traditionnel avancé
- Jour 4: Introduction a l’intelligence artificiel (ML et DL)
- Jour 5: Technique modern - Conversation de nos courent

Modèles de couleurs

- RGB (Red, Green, Blue)
- CMYK (Cyan, Magenta, Yellow, Key/Black)
- HSV (Hue, Saturation, Value)
- HSL (Hue, Saturation, Lightness)
- YUV (Luminance, couleur)

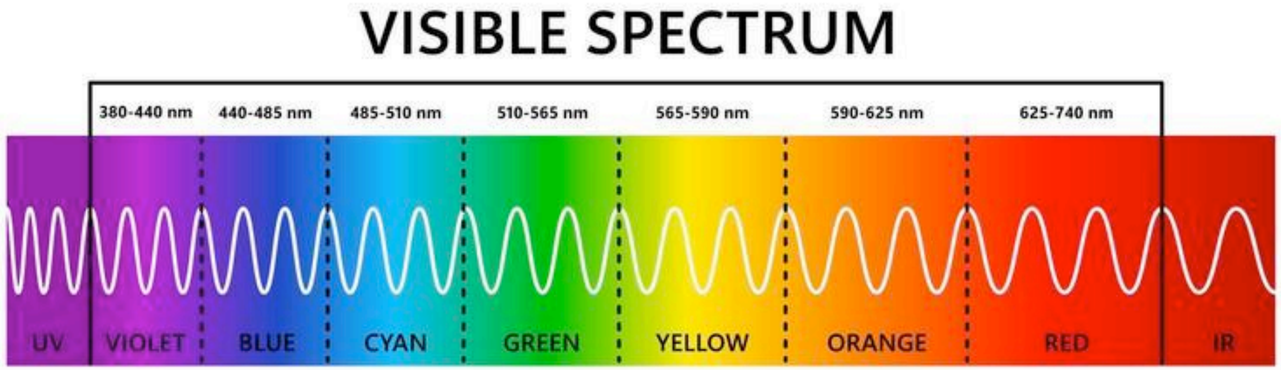
Qu'est-ce qu'une couleur ?

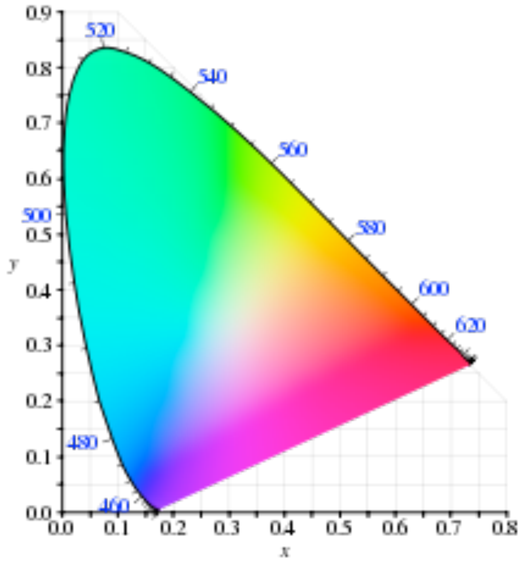
- Distribution spectrale de puissance de lumière
- Réfléchie ou transmise par un objet
- Photorécepteurs
  - Capture la lumière visible
  - Rouge (onde de 700 nm)
  - Vert (onde de 546 nm)
  - Bleue (onde de 435 nm)

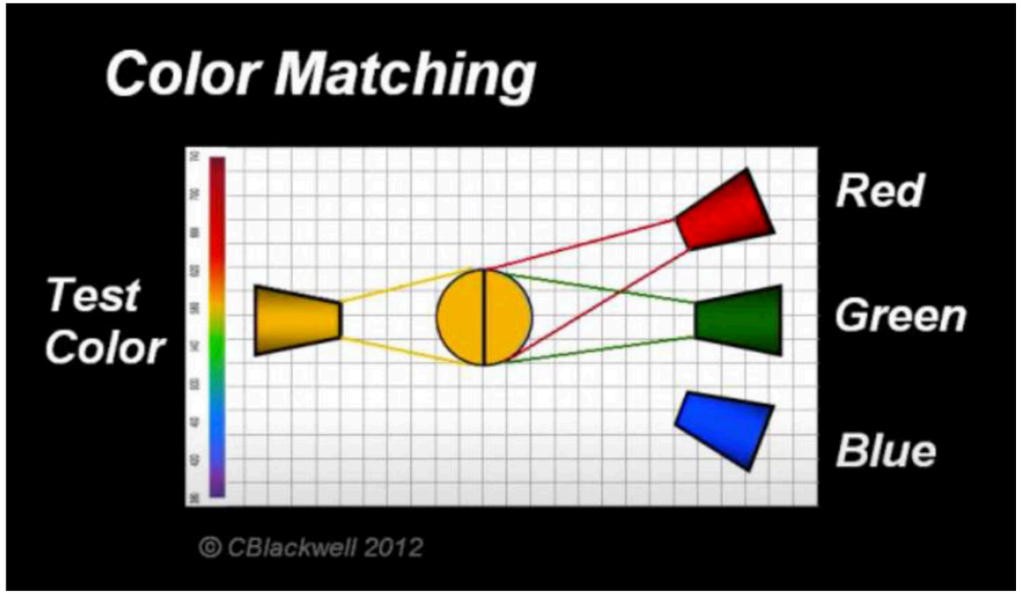


Spectre lumineux

- Couleur primaires
- Combiner les couleurs
- Diagramme de chromaticité CIE

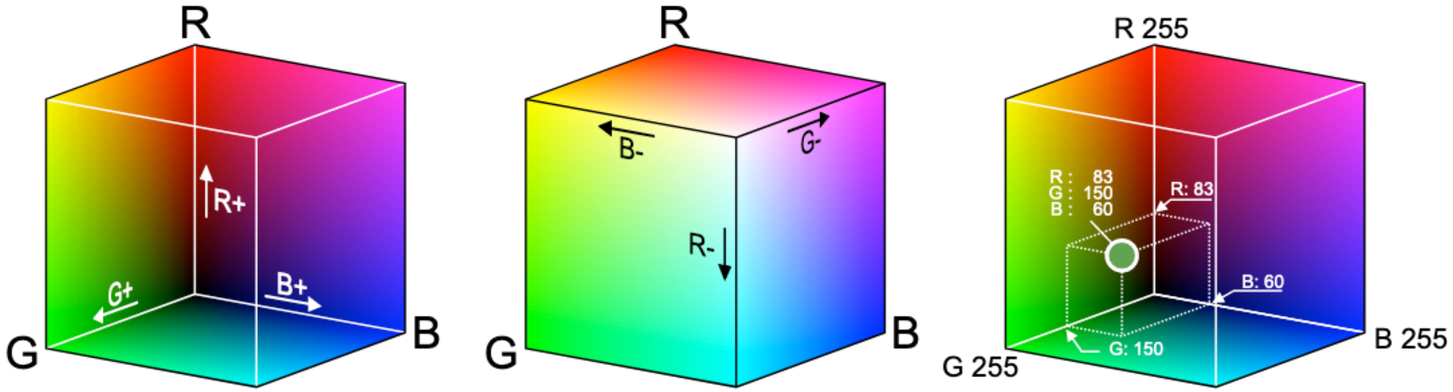






RGB

- Rouge, vert et bleue
- Cube de couleur



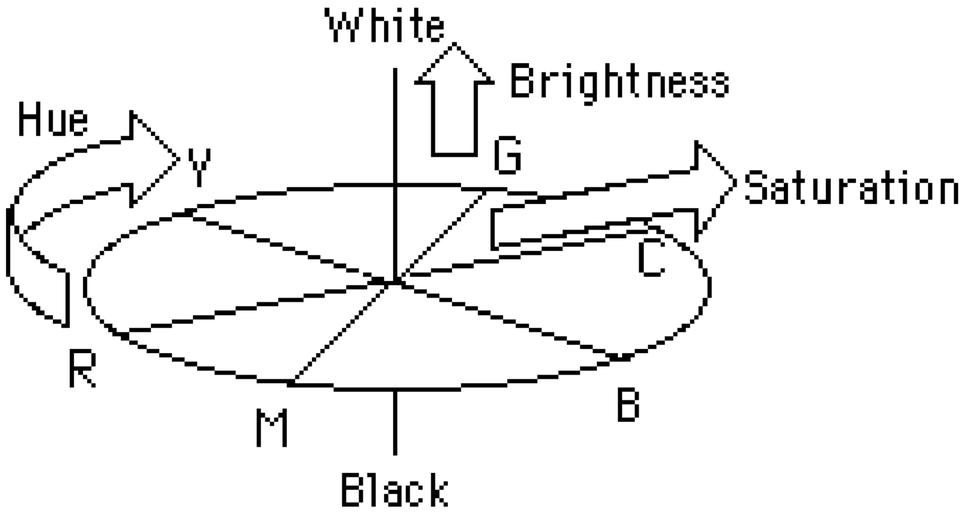
Original Image | Red Channel | Green Channel | Blue Channel

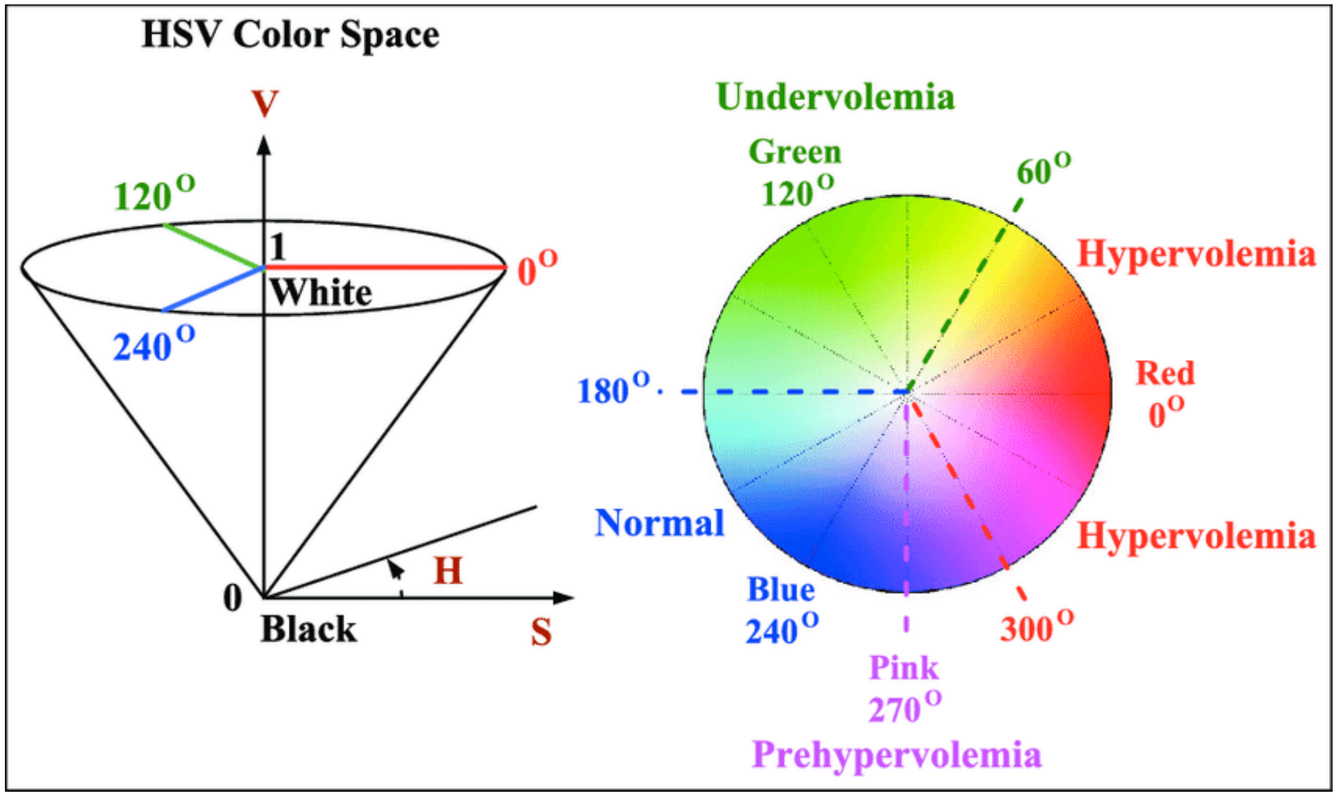
HSB et HSI

- Hue (Teinte) - définit le type de couleur
- Saturation - une mesure de la vivacité de la couleur
- Brightness (Luminosity) - une impression visuelle de l'intensité de la couleur
- Intensite
  - (R+G+B)/3
- Valeur
  - Max(R,G,B)

HSV

- Utilisé dans les graphiques
- Définies en termes d'intensités

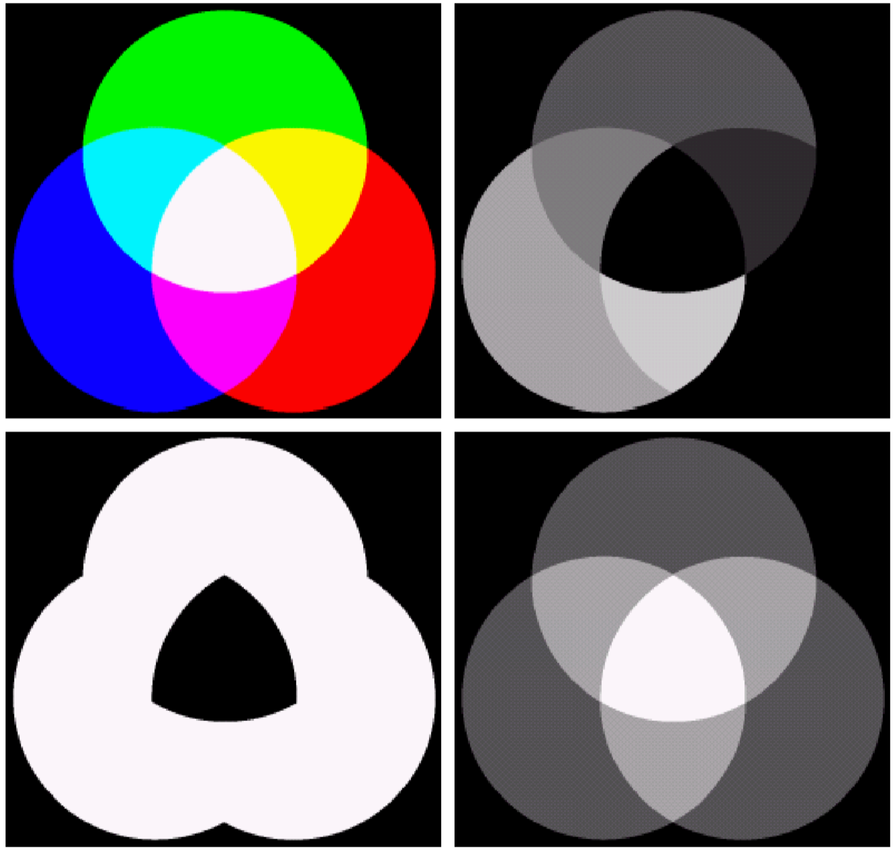


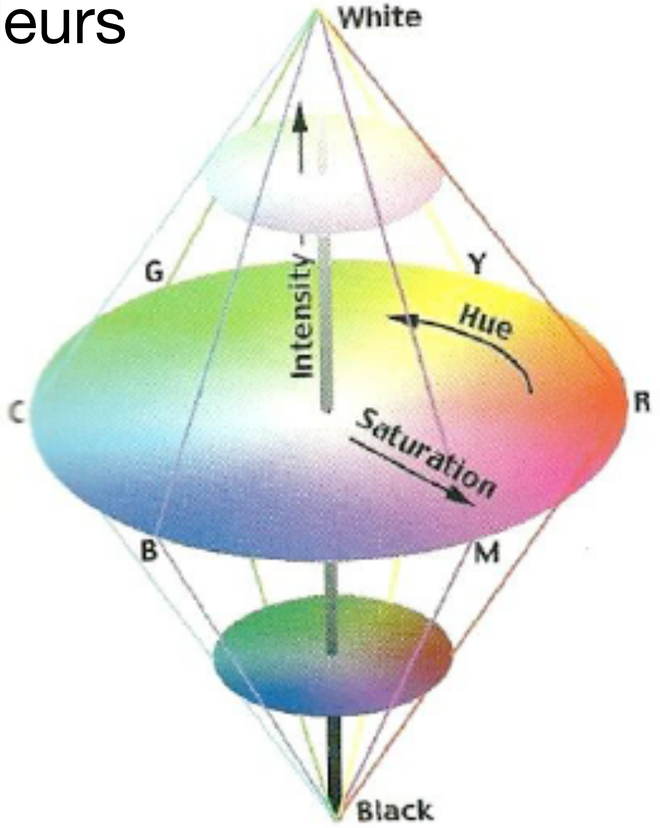


HSI

- Similaire à la façon dont l'œil humain perçoit les couleurs

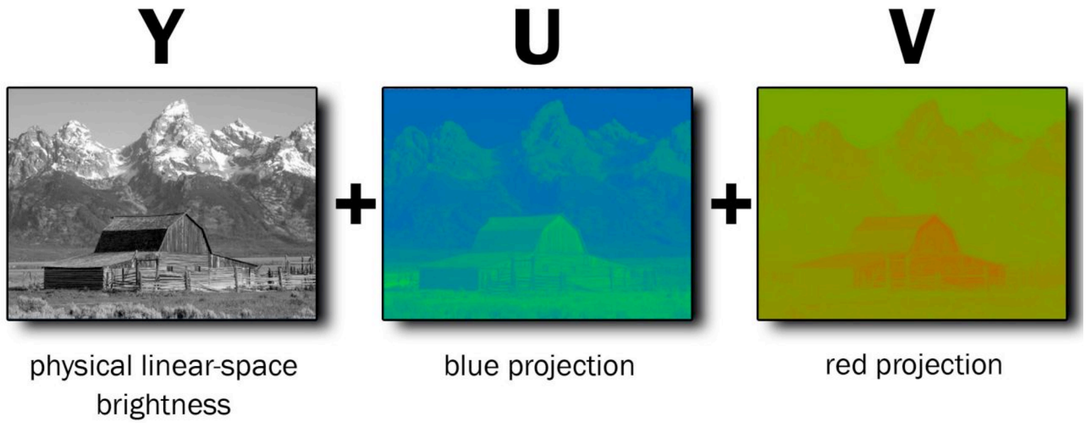
- RGB et HSI
  - Hue
  - Saturation
  - Intensity





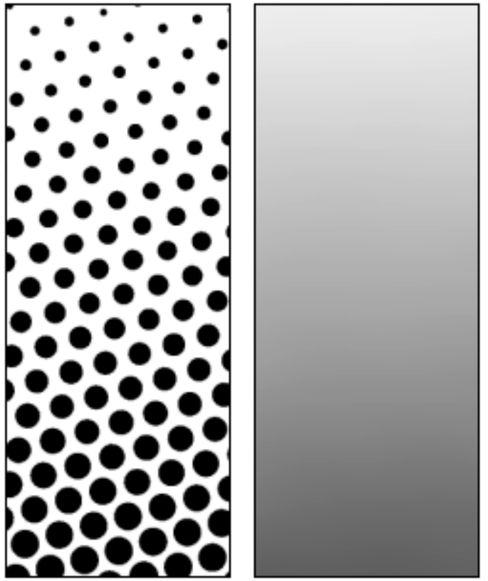
YUV

- YPbPr
  - Vidéo composante analogique
- YCbCr
  - Télévision couleur numérique



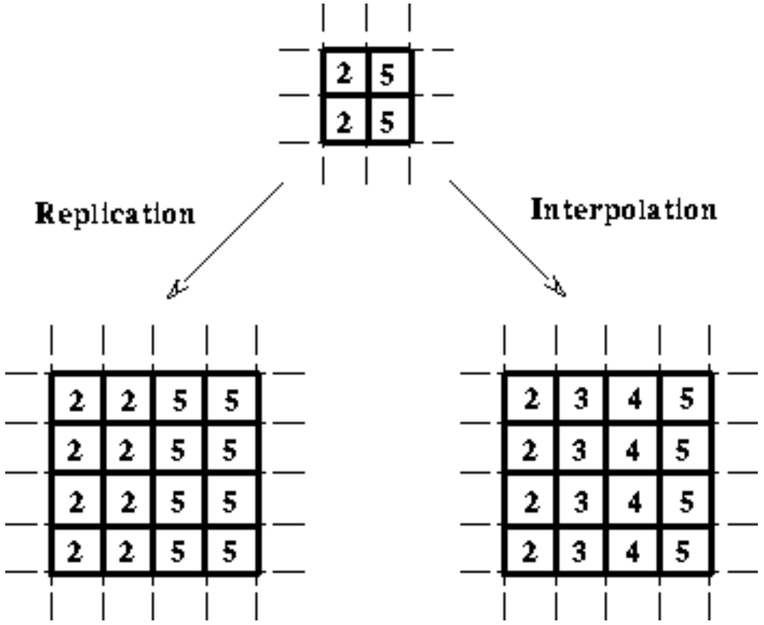
Impression de couleur

- Halftone screening
  - Criblage en demi-teinte
  - Surface des points proportionnelle à l'intensité



Transformation

- Ecaillage (Scaling)
- Rotation
- Recadrage (Crop)
- Correction de couleur
- Filtrage



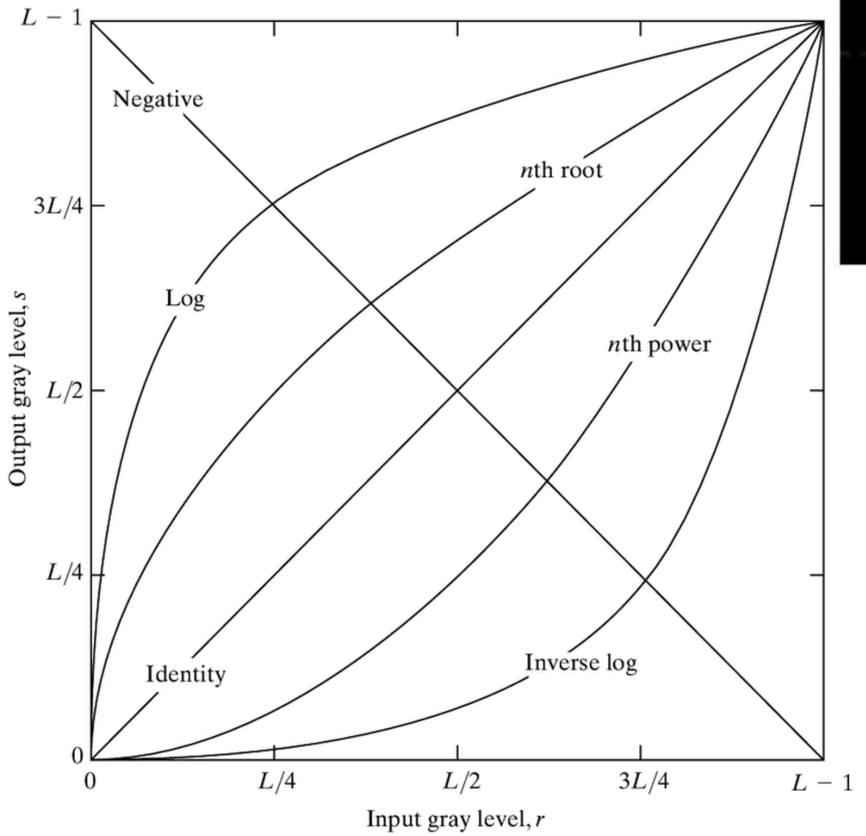
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

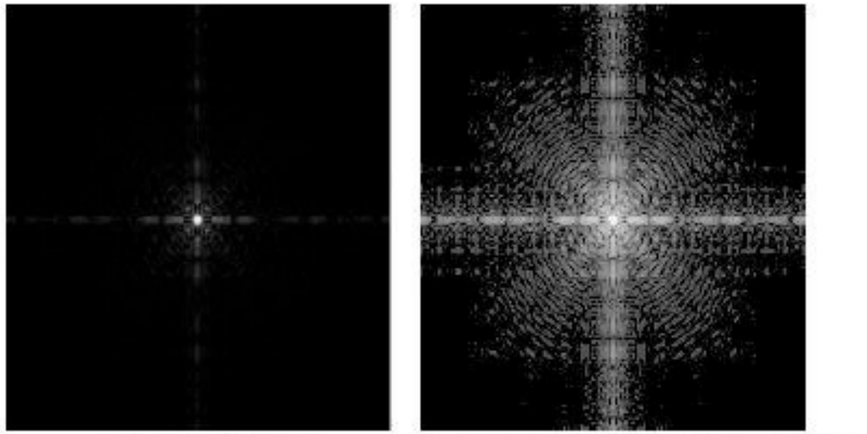
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformations logarithmiques

- Correction des couleurs

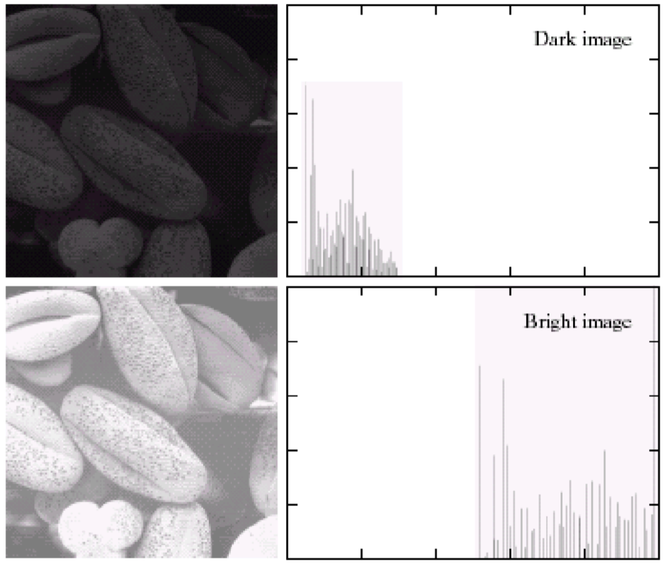


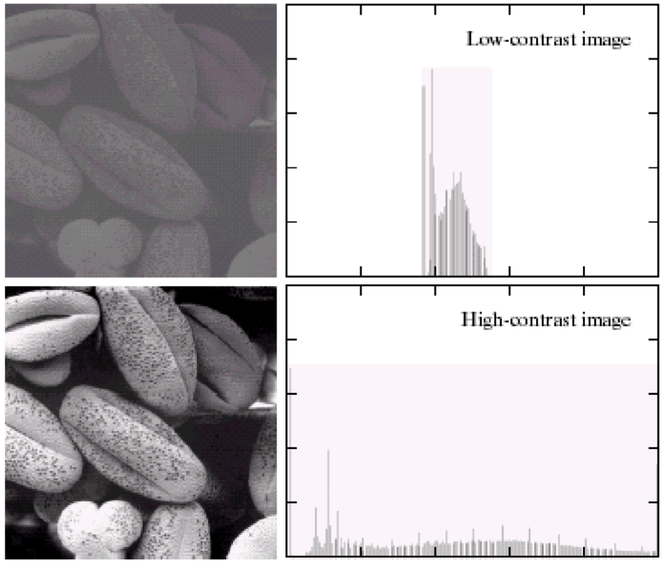


(a) Original image (b) Result of Log transform with $c = 1$

Traitement des histogrammes

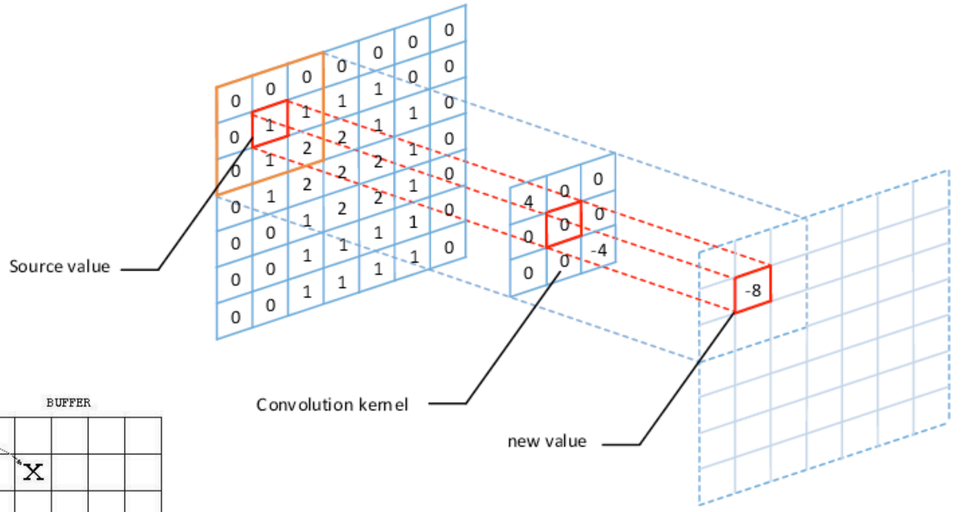
- Types d’images
  - Sombre
  - Léger
  - Faible contraste
  - Contraste élevé

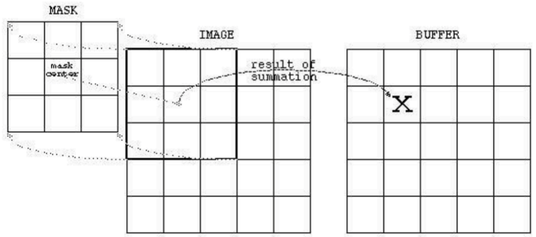




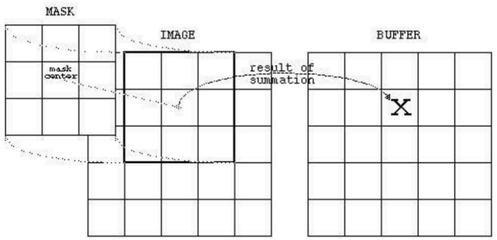
Filtrage spatial

- Operation de voisinage
- Processus de convolution





a) Overlay the convolution mask in upper left corner of the image. Multiply coincident terms, sum, put result into the image buffer at the location that corresponds to the mask's current center, which is (r,c) = (1,1).

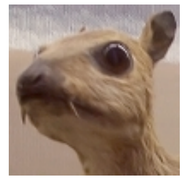


b) Move the mask one pixel to the right, multiply coincident terms, sum, and place the new result into the buffer at the location that corresponds to the new center location of the convolution mask, now at (r,c) = (1,2). Continue to the end of the row.

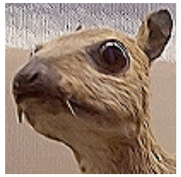
Convolution

- Flou
- Detection de bord
- Lissage

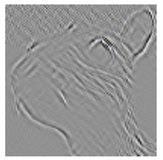
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- https://en.wikipedia.org/wiki/Kernel_(image_processing)

Python + OpenCV

- GitHub
  - https://github.com/cmcin019/mini-cours
- Google Colab

Lire une image

- Lire avec imread()
- Voir l’image avec plt.imshow
- Propriétés des images
  - Size
  - Shape
  - Type

Nous pouvons maintenant ouvrir une image :

```
[ ]  input_image=cv2.imread('noidea.jpg')
```

Nous pouvons découvrir plusieurs choses sur cette image

```
print(f"Size: {input_image.size}")
print(f"Shape: {input_image.shape}")
print(f"Type: {input_image.dtype}")
```

```
Size: 776250
Shape: (414, 625, 3)
Type: uint8
```

Et nous pouvons visualiser l'image:

```
[ ]  plt.imshow(input_image)
```

Couleurs avec OpenCV

- Séparer les couleurs principales
- Convertir en échelle de gris
- Convertir en RGB

```
# split channels
b,g,r=cv2.split(input_image)
# show one of the channels (this is
plt.imshow(r, cmap='gray')
--NORMAL--
```

```
merged=cv2.merge([r,g,b])
# merge takes an array of single channel matrices
plt.imshow(merged)
```

```
opencv_merged=cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB
plt.imshow(opencv_merged)
```

Recadrer l'image

- Tranche d’image

```
dogface = input_image[60:250, 70:350]
plt.imshow(dogface)
```

```
fresh_image=cv2.imread('noidea.jpg')

fresh_image[200:200+dogface.shape[0], 200:200+dogface.shape[1]]=dogface
print(dogface.shape[0])
print(dogface.shape[1])
plt.imshow(fresh_image)
```

Flou et accentuation

```
d=3
img_blur3 = cv2.GaussianBlur(input_image, (2*d+1, 2*d+1), -1)[d:-d,d:-d
plt.imshow(cv2.cvtColor(img_blur3, cv2.COLOR_BGR2RGB))
```

```
# Define the sharpening kernel. The values might need to be adjusted depending on the image.
sharpening_kernel = np.array([
    [-1, -1, -1],
    [-1,  9, -1],
    [-1, -1, -1]
])

sharpened_image = cv2.filter2D(img_blur3, -1, sharpening_kernel)

plt.imshow(cv2.cvtColor(sharpened_image, cv2.COLOR_BGR2RGB))
```

Fin

- Merci