Chapitre 1. Perception de la couleur (2 Semaines)

- Colorimétrie. Lumière et couleur dans la perception humaine
- Systèmes de représentation de la couleur : RGB, XYZ, YUV, HSV, YIQ
- Formats couleur et stratégies de traitement de l'image couleur

Ce cours de Master 1 se situe à l'intersection de la physique, de la biologie et de l'informatique. Comprendre la couleur est crucial, car contrairement à la forme, la couleur est une **perception subjective** reconstruite par le cerveau.

Chapitre 1 : Perception et Traitement de la Couleur

1. Colorimétrie : Lumière et perception humaine

La perception de la couleur repose sur trois piliers : la source lumineuse, l'objet (réflectance) et l'observateur (l'œil).

- **La vision trichromatique :** La rétine humaine possède deux types de photorécepteurs. Les **bâtonnets** (vision nocturne) et les **cônes** (vision diurne). Il existe trois types de cônes, sensibles à des longueurs d'onde différentes :
  - **L (Long) :** Sensibles au Rouge.
  - **M (Medium) :** Sensibles au Vert.
  - **S (Short) :** Sensibles au Bleu.
- **Synthèse Additive vs Soustractive :**
  - **Additive (RVB) :** On mélange de la lumière (écrans). La somme des trois donne du blanc.
  - **Soustractive (CMJN) :** On mélange des pigments (imprimerie). La somme donne du noir.

2. Systèmes de représentation de la couleur (Espaces colorimétriques)

Un espace couleur est un modèle mathématique permettant de décrire les couleurs par des nombres.

A. Le système XYZ (CIE 1931)

C'est l'espace de référence standard. Il a été créé pour définir mathématiquement toutes les couleurs visibles par l'œil humain moyen. Le diagramme de chromaticité qui en découle montre l'ensemble des couleurs perçues.

B. Le système RGB (Red, Green, Blue)

C'est le modèle le plus utilisé en informatique. Chaque pixel est codé par l'intensité de ces trois primaires.

- **Limite :** Il est dépendant du matériel (le "rouge" d'un écran n'est pas le même que sur un autre).

C. Les systèmes YUV / YIQ

Ces modèles séparent l'information de luminosité (**Luminance Y**) de l'information de couleur (**Chrominance U, V** ou **I, Q**).

- **Pourquoi ?** L'œil humain est beaucoup plus sensible aux variations de luminosité qu'aux variations de couleur. Ces formats permettent de compresser l'image en dégradant la couleur sans que l'œil ne le remarque (utilisé en télévision et JPEG).

D. Le système HSV (Hue, Saturation, Value)

Plus intuitif pour l'humain que le RGB :

- **Hue (Teinte) :** La "nature" de la couleur (angle sur un cercle de 0 à 360°).
- **Saturation :** La pureté de la couleur (vif ou gris).
- **Value (Valeur) :** La brillance.

3. Formats couleur et stratégies de traitement de l'image

Traiter une image couleur n'est pas simplement appliquer trois fois un algorithme de niveaux de gris. Il existe deux stratégies majeures :

A. Traitement Marginal (Composante par composante)

On applique un filtre (ex: flou, détection de contours) sur les canaux R, G et B séparément.

- **Risque :** Cela peut créer des couleurs "artefacts" qui n'existaient pas dans l'image originale, car les corrélations entre canaux ne sont pas respectées.

B. Traitement Vectoriel

L'image est traitée comme un champ de vecteurs $C(x,y) = [R, G, B]^T$. On utilise des outils mathématiques (Jacobiennes, gradients vectoriels) pour traiter la couleur comme un tout cohérent.

C. Stratégie par Découplage (Luminance/Chrominance)

C'est la stratégie la plus efficace :

1. Convertir l'image de **RGB** vers **Lab** ou **HSV**.
2. Traiter uniquement la composante de brillance (L ou V) pour améliorer le contraste ou la netteté.
3. Reconvertir vers RGB.

- **Avantage :** On préserve l'aspect chromatique original tout en améliorant la qualité visuelle.

Exercice d'application : Conversion en Python (OpenCV)

Dans cet exercice, nous allons extraire la composante de "Teinte" (Hue) pour isoler un objet de couleur spécifique, ce qui est impossible en RGB pur.

Python

```
import cv2
import numpy as np

# Charger l'image couleur
image = cv2.imread('image.jpg')

# 1. Conversion de BGR (format par défaut d'OpenCV) vers HSV
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# 2. Définir une plage de couleur (ex: isoler le bleu)
lower_blue = np.array([100, 150, 50])
upper_blue = np.array([140, 255, 255])

# 3. Créer un masque qui ne garde que les pixels bleus
mask = cv2.inRange(hsv, lower_blue, upper_blue)

# Visualisation
cv2.imshow('Image Originale', image)
cv2.imshow('Masque Bleu (Espace HSV)', mask)
```

```
cv2.waitKey(0)
```

Pourquoi ce chapitre est important ?

En Master 1, vous devez comprendre que le choix de l'espace couleur dépend de l'application :

- **Stockage/Transmission :** YUV (compression).
- **Segmentation d'objets :** HSV (robustesse aux ombres).
- **Affichage :** RGB.
- **Métrologie/Mesure précise :** XYZ ou Lab.

Souhaitez-vous que je développe la partie sur la "Compression JPEG" pour expliquer comment on utilise concrètement le format YUV pour réduire la taille des fichiers ?

Voici la transcription de l'exercice précédent en **MATLAB**, qui est l'outil de référence pour le traitement d'images en milieu universitaire et ingénierie.

MATLAB traite les images nativement comme des matrices multidimensionnelles, ce qui rend les calculs sur les espaces couleurs très efficaces.

Exercice : Conversion et Segmentation (Espace HSV)

L'objectif est d'isoler une couleur spécifique en utilisant le découplage Luminance/Chrominance.

Matlab

```
% 1. Lecture de l'image
img = imread('peppers.png'); % Image d'exemple incluse dans MATLAB

% 2. Conversion de l'espace RGB vers HSV
% En MATLAB, les valeurs HSV sont normalisées entre 0 et 1
img_hsv = rgb2hsv(img);

% Extraction des composantes pour visualisation
H = img_hsv(:,:,1); % Teinte (Hue)
S = img_hsv(:,:,2); % Saturation
V = img_hsv(:,:,3); % Valeur (Luminance)

% 3. Définition des seuils pour isoler une couleur (ex: le Rouge)
% Le rouge est proche de 0 ou 1 en Teinte (Hue)
% On définit un masque : Teinte < 0.05 OU Teinte > 0.95
% On ajoute une condition sur la Saturation pour éviter le blanc/gris
masque = (H < 0.05 | H > 0.95) & (S > 0.4);

% 4. Application du masque sur l'image originale
% On multiplie chaque canal R, G, B par le masque binaire
img_segmentee = img;
img_segmentee(repmat(~masque, [1 1 3])) = 0;

% 5. Affichage des résultats
figure;
subplot(2,2,1); imshow(img); title('Image Originale (RGB)');
subplot(2,2,2); imshow(H); title('Composante Teinte (H)');
subplot(2,2,3); imshow(masque); title('Masque Binaire');
subplot(2,2,4); imshow(img_segmentee); title('Objet Segmenté');

% Bonus : Histogramme de la luminance
figure;
imhist(V); title('Distribution de la Luminance (Value)');
```

Explications des fonctions MATLAB clés :

- **`rgb2hsv()`** : Transforme les triplets [R, G, B] en [H, S, V]. C'est indispensable car en RGB, si la luminosité change, les trois valeurs R, G et B changent proportionnellement, rendant la détection de couleur instable. En HSV, seule la composante V change.
- **`repmat()`** : Utilisée ici pour transformer notre masque 2D (une seule couche) en un masque 3D (trois couches) afin de l'appliquer aux canaux Rouge, Vert et Bleu simultanément.
- **`imhist()`** : Permet d'analyser la répartition lumineuse. En Master 1, c'est l'outil de base pour décider si une image nécessite une **égalisation d'histogramme** (pour améliorer le contraste).

Stratégie de traitement vectoriel en MATLAB

Si vous souhaitez effectuer un filtrage (ex: un filtre de moyenne) sans créer d'artefacts colorés, privilégiez le traitement sur la composante V uniquement :

Matlab
```
% Filtrage de la luminance seule pour préserver les couleurs
V_floute = imfilter(V, fspecial('gaussian', [5 5], 2));
img_hsv_traitee = cat(3, H, S, V_floute);
img_rgb_finale = hsv2rgb(img_hsv_traitee);

imshow(img_rgb_finale);
```

Souhaitez-vous que je vous explique comment calculer la distance colorimétrique entre deux pixels en utilisant la norme Euclidienne dans l'espace Lab (plus précis que le RGB) ?