

Réponse Q01 a

Pour convertir une couleur définie en **RGB** (avec R, G, B normalisés entre 0 et 1) en **HSI**, on suit ces étapes :

a) Intensité (I)
$$I = \frac{R + G + B}{3}$$

b) Saturation (S)
$$S = 1 - \frac{\min(R, G, B)}{I}, \quad \text{si } I \neq 0, \text{ sinon } S = 0$$

c) Teinte (H)

- Calculer l'angle en radians à partir des composantes RGB :

$$\theta = \arccos \left(\frac{1}{2} \cdot \frac{(R - G) + (R - B)}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right)$$

- Déterminer H en fonction de la position des couleurs :

- Si $B \leq G$: $H = \theta$

- Sinon : $H = 2\pi - \theta$

- Convertir H en degrés : $H_{\text{degrés}} = H \times \frac{180}{\pi}$

a) Intensité
$$I = \frac{R + G + B}{3} = \frac{0.5 + 0.3 + 0.2}{3} = 0.333$$

b) Saturation
$$\min(R, G, B) = \min(0.5, 0.3, 0.2) = 0.2$$

c) Teinte
$$S = 1 - \frac{\min(R, G, B)}{I} = 1 - \frac{0.2}{0.333} = 1 - 0.6 = 0.4$$

- Calcul intermédiaire :

$$\theta = \arccos \left(\frac{1}{2} \cdot \frac{(0.5 - 0.3) + (0.5 - 0.2)}{\sqrt{(0.5 - 0.3)^2 + (0.5 - 0.2)(0.3 - 0.2)}} \right)$$

$$\theta = \arccos \left(\frac{1}{2} \cdot \frac{0.2 + 0.3}{\sqrt{(0.2)^2 + (0.3)(0.1)}} \right) = \arccos \left(\frac{0.25}{\sqrt{0.04 + 0.03}} \right)$$

$$\theta = \arccos \left(\frac{0.25}{0.2646} \right) = \arccos(0.944)$$

$$\theta \approx 19.89^\circ$$

- $B < G$, donc : $H = \theta = 19.89^\circ$

3. Résultat final

Les composantes HSI de la couleur sont :

- $H \approx 19.89^\circ$,
- $S = 0.4$,
- $I = 0.333$.

Réponse Q01 b

Pour convertir H , S , et I en R , G , et B , nous utilisons différentes formules selon la valeur de la teinte H .

Cas 1 : $0^\circ \leq H < 120^\circ$

$$R = I \cdot \left(1 + \frac{S \cdot \cos H}{\cos(60^\circ - H)}\right)$$

$$G = I \cdot (1 - S)$$

$$B = 3I - (R + G)$$

Cas 2 : $120^\circ \leq H < 240^\circ$

Dans ce cas, $H' = H - 120^\circ$ et les formules sont ajustées :

$$G = I \cdot \left(1 + \frac{S \cdot \cos H'}{\cos(60^\circ - H')}\right)$$

$$B = I \cdot (1 - S)$$

$$R = 3I - (G + B)$$

Cas 3 : $240^\circ \leq H < 360^\circ$

Dans ce cas, $H' = H - 240^\circ$:

$$B = I \cdot \left(1 + \frac{S \cdot \cos H'}{\cos(60^\circ - H')}\right)$$

$$R = I \cdot (1 - S)$$

$$G = 3I - (B + R)$$

Dans notre cas

- $H = 30^\circ$,
- $S = 0.6$,
- $I = 0.5$.

Cas : $0^\circ \leq H < 120^\circ$

1. Calcul de R :

$$R = I \cdot \left(1 + \frac{S \cdot \cos H}{\cos(60^\circ - H)}\right)$$

$$R = 0.5 \cdot \left(1 + \frac{0.6 \cdot \cos 30^\circ}{\cos(60^\circ - 30^\circ)}\right)$$

$$R = 0.5 \cdot \left(1 + \frac{0.6 \cdot 0.866}{0.866}\right) = 0.5 \cdot (1 + 0.6) = 0.5 \cdot 1.6 = 0.8$$

2. Calcul de G :

$$G = I \cdot (1 - S) = 0.5 \cdot (1 - 0.6) = 0.5 \cdot 0.4 = 0.2$$

3. Calcul de B :

$$B = 3I - (R + G) = 3 \cdot 0.5 - (0.8 + 0.2) = 1.5 - 1 = 0.5$$

3. Résultat final

Les composantes RGB correspondantes sont :

- $R = 0.8$,
- $G = 0.2$,
- $B = 0.5$.

Réponse Q02 a

```
import cv2
import numpy as np

# Charger l'image
image = cv2.imread('HSI.png')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Conversion en espace HSI
image_hsi = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2HSV) # Utilisation de HSV comme approximation

# Définir les plages pour la couleur rouge
lower_red1 = np.array([0, 120, 70]) # Plage 1 (rouge clair)
upper_red1 = np.array([10, 255, 255])

lower_red2 = np.array([170, 120, 70]) # Plage 2 (rouge foncé)
upper_red2 = np.array([180, 255, 255])

# Création des masques
mask1 = cv2.inRange(image_hsi, lower_red1, upper_red1)
mask2 = cv2.inRange(image_hsi, lower_red2, upper_red2)
mask = mask1 + mask2
print(mask)
# Application du masque
segmented_image = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)

# Affichage des résultats
cv2.imshow('Original Image', image)
cv2.imshow('Segmented Image', segmented_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

#=====

# Charger l'image
image = cv2.imread('D:\Enseignements\Maamar\TP-05\HSI.png')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Étape 1 : Détection des contours sur chaque canal RGB
edges_r = cv2.Canny(image_rgb[:, :, 0], 100, 200) # Canal Rouge
edges_g = cv2.Canny(image_rgb[:, :, 1], 100, 200) # Canal Vert
edges_b = cv2.Canny(image_rgb[:, :, 2], 100, 200) # Canal Bleu

# Étape 2 : Fusionner les contours
edges_combined = cv2.bitwise_or(edges_r, edges_g)
edges_combined = cv2.bitwise_or(edges_combined, edges_b)

# Étape 3 : Afficher les résultats
cv2.imshow('Contours Rouge', edges_r)
cv2.imshow('Contours Vert', edges_g)
cv2.imshow('Contours Bleu', edges_b)
cv2.imshow('Contours Fusionnés', edges_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()

#=====

# Conversion en espace HSI (HSV)
image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Détection des contours sur chaque canal (H, S, V)
edges_h = cv2.Canny(image_hsv[:, :, 0], 100, 200) # Teinte
edges_s = cv2.Canny(image_hsv[:, :, 1], 100, 200) # Saturation
edges_v = cv2.Canny(image_hsv[:, :, 2], 100, 200) # Intensité

# Fusionner les contours
edges_combined_hsv = cv2.bitwise_or(edges_h, edges_s)
edges_combined_hsv = cv2.bitwise_or(edges_combined_hsv, edges_v)

# Afficher les résultats
cv2.imshow('Contours (HSI fusionnés)', edges_combined_hsv)
cv2.waitKey(0)
cv2.destroyAllWindows()
```