Une image contenant Police, Graphique, logo, graphisme

Description générée automatiquement

**Département de génie logiciel et des T.I.**

Rapport de Laboratoire

|  |  |
| --- | --- |
| **Numéro du laboratoire** | Laboratoire 1 |
| **Nom du laboratoire** | Espaces de couleurs |
| **Étudiant(s)** | Olivier, Granger- Hotte  Gildor Makesa Mvuemba |
| **Code(s) permanent(s)** | MAKM87260201  GRAO89120006 |
| **Numéro d’équipe** | 06 |
| **Cours** | GTI 411 |
| **Session** | Hiver 2025 |
| **Groupe** | S20251-GTI411-01 |
| **Chargé(e) de laboratoire** | Lucas Mercier |
| **Date** | 30 janv. 25 |

Table des matières

[Introduction 4](#_Toc189434782)

[Outils et concepts 5](#_Toc189434783)

[Espace de couleur RGB 5](#_Toc189434784)

[Espace de couleur CMYK 5](#_Toc189434785)

[Espace de couleur HSV 6](#_Toc189434786)

[Espace de couleur LAB 6](#_Toc189434787)

[Implémentation 7](#_Toc189434788)

[Conversion entre les espaces de couleurs 7](#_Toc189434789)

[Décompositions 11](#_Toc189434790)

[Transformations 12](#_Toc189434791)

[Librairies utilisées 12](#_Toc189434792)

[Résultats et discussion 13](#_Toc189434793)

[Curseur/Interpolation CMYK 13](#_Toc189434794)

[Curseur/Interpolation HSV 14](#_Toc189434795)

[Curseur/Interpolation LAB 15](#_Toc189434796)

[Décomposition CMYK 16](#_Toc189434797)

[Décomposition HSV 17](#_Toc189434798)

[Décomposition LAB 18](#_Toc189434799)

[Transformations 19](#_Toc189434800)

[Difficultés rencontrées 20](#_Toc189434801)

[Améliorations possibles ou éventuelles 21](#_Toc189434802)

[Conclusion 21](#_Toc189434803)

[Annexe : Manuel d’utilisateur 22](#_Toc189434804)

[Installation des dépendances 22](#_Toc189434805)

[Exécution du code 22](#_Toc189434806)

# Introduction

Ce premier laboratoire nous introduit à la manipulation des couleurs, un aspect fondamental du traitement d'images. En effet, différents espaces de couleur sont utilisés en fonction des besoins spécifiques des applications, que ce soit pour la correction d’image, la segmentation d’objets ou l'amélioration de la qualité visuelle. Bien que l’espace de couleur ***RGB*** soit couramment utilisé pour l’affichage des images sur les écrans, il n’est pas toujours le plus adapté pour certaines opérations de traitement d’image. Par exemple, le modèle ***HSV*** est souvent privilégié pour la segmentation, car il sépare la teinte, la saturation et la luminosité, facilitant l’extraction d’éléments spécifiques d’une image. Ou encore, le modèle ***CMYK***, quant à lui, est essentiel pour l’impression, car il reflète la manière dont les encres se mélangent sur le papier, contrairement à l’addition des couleurs en RGB. Enfin, l’espace ***LAB*** est particulièrement utile pour les ajustements avancés de la couleur et la préservation des détails lors des modifications d’image, car il s’appuie sur un modèle perceptif proche de la vision humaine.

Par conséquent, ce laboratoire vise à approfondir la compréhension et l’utilisation des espaces de couleur comme RGB, HSV, CMYK et LAB, ainsi qu’à explorer les transformations d’image telles que la modification du contraste et de la luminosité. Les fonctionnalités qui seront abordées dans ce rapport de laboratoire trouvent des applications concrètes dans divers logiciels et domaines technologiques. Par exemple, **Adobe Photoshop** exploite plusieurs espaces de couleurs pour la retouche photo, **Adobe Illustrator** permet la conversion entre RGB et CMYK pour l’impression, ou encore **MATLAB** est largement utilisé en analyse scientifique et imagerie médicale, notamment pour l’optimisation du contraste. Ces applications illustrent l'importance des concepts abordés dans ce laboratoire, qui sont au cœur des outils professionnels, de l’intelligence artificielle et des technologies d’imagerie avancée.

Le laboratoire est structuré en trois grandes parties. Dans la première partie, nous explorerons la conversion entre différents espaces de couleur et implémenterons des méthodes pour passer d’un modèle à un autre. La deuxième partie sera consacrée à la décomposition d’une image en ses différents canaux de couleur, permettant une meilleure compréhension de la structure des espaces de couleur. Enfin, la troisième partie traitera des transformations d’image, notamment la modification du contraste et de la luminosité à l’aide de curseurs interactifs. Chacune de ces parties sera accompagnée d’exemples pratiques et d’implémentations guidées afin d’assurer une compréhension approfondie des concepts abordés.

# Outils et concepts

Présentation des espaces de couleurs

## Espace de couleur RGB

Tout d’abord, le premier espace de couleur que nous explorons est le modèle ***RGB*** (Rouge, Vert, Bleu). Ce modèle est le plus couramment utilisé en imagerie numérique et en affichage sur écran. Ce modèle repose sur l’addition des trois couleurs primaires de la lumière : en combinant le rouge, le vert et le bleu à différentes intensités, il est possible de reproduire une large gamme de couleurs, et leur addition maximale donne du blanc. L’un des principaux avantages du modèle *RGB* est sa simplicité et son intégration universelle dans les écrans d’ordinateurs, de télévisions et d’appareils mobiles. Par conséquent, il représente aussi un standard en imagerie numérique.

Toutefois, le modèle *RGB* n’est pas idéal pour l’impression. Les écrans émettent de la lumière, tandis que les imprimantes fonctionnent en mode soustractif avec des encres (*CMYK*). Par conséquent, une couleur affichée sur un écran peut apparaître différemment une fois imprimée sur papier. De plus, la gamme de couleurs du modèle *RGB* est relativement limité, ce qui peut entraîner des difficultés à représenter certaines teintes très saturées ou des couleurs précises utilisées en graphisme professionnel.

## Espace de couleur CMYK

Par la suite, le deuxième espace de couleur que nous voyons est l’espace *CMYK*. Cet espace fonctionne de manière soustractive au contraire de l’espace RGB qui est de manière additive. Plus précisément, le modèle *CMYK* absorbe certaines longueurs d’onde à l’aide de pigments ou d’encres. Ce modèle est composé des couleurs cyan (C), magenta (M), jaune (Y) et Noir (K, pour "Key"), et est principalement utilisé en impression. En effet, les écrans émettent de la lumière, tandis que les imprimantes fonctionnent en mode soustractif avec des encres. Par conséquent, l’un des principaux avantages du *CMYK* est sa fidélité aux couleurs imprimées, offrant un rendu plus réaliste sur le papier qu’en le visionnons dans l’écran. De plus, l’ajout du noir comme composante indépendante permet de renforcer les contrastes, d’améliorer la netteté et d’optimiser l’utilisation de l’encre, ce qui le rend plus économique. Il a donc une meilleure résolution en sortie et il permet d’économiser de l’encre.

Cependant, ce modèle de couleur possède quelques inconvénients. En effet, certaines couleurs très vives, par exemple les couleurs extrêmement saturées, ne peuvent pas être reproduites fidèlement en CMYK. Aussi, la conversion de *RGB* à *CMYK* n’est pas toujours exacte, comme la conversion de RGB à CMYK n’est pas toujours de 100%.

## Espace de couleur HSV

Troisièmement, l’espace de couleur que nous étions confrontés est HSV (hue, saturation, value). Ce modèle est conçu pour être plus proche de la perception humaine des couleurs. Contrairement à RGB et CMYK, qui représentent les couleurs à travers des combinaisons de trois canaux de base, HSV sépare la teinte, la saturation et la luminosité, ce qui permet une manipulation plus intuitive des couleurs. Il est donc souvent utilisé dans des logiciels de traitement de photos, comme il permet de faire des retouches précises. La teinte (le “H” ou “Hue”) est aussi plus intuitive que l’espace RGB ou le CMYK afin de visualiser les couleurs.

Son principal inconvénient est qu’il nécessite des conversions mathématiques depuis et vers RGB pour être affiché sur les écrans ou imprimé.

## Espace de couleur LAB

Le modèle *LAB* se distingue des autres espaces de couleur par son indépendance vis-à-vis des périphériques d’affichage et d’impression. Contrairement à *RGB* et *CMYK*, qui dépendent des caractéristiques des écrans et des imprimantes, *LAB* repose sur des calculs complexes qui lui permettent de représenter les couleurs de manière uniforme et absolue, en accord avec la perception humaine. Il s’agit d’un standard international largement utilisé dans des domaines nécessitant une grande précision colorimétrique. L’un de ses principaux atouts réside dans sa capacité à quantifier les différences de couleurs perçues par l’œil humain en les traduisant sous forme de distances mathématiques. Grâce à cela, *LAB* offre une gamme de couleurs étendue, ce qui en fait un choix privilégié pour des applications avancées comme la correction colorimétrique, la reconnaissance d’objets et l’amélioration de la qualité des images. Il est notamment intégré dans des logiciels professionnels tels qu’Adobe Photoshop, où il permet des ajustements de couleur bien plus précis que dans *RGB* ou *CMYK*.

Il a comme inconvénient d’être beaucoup plus complexe que ses paires, et nécessite la conversion en format intermédiaire XYZ afin de passer de RGB à LAB par exemple. Il est aussi moins populaire que certains autres espaces de couleur, et il n’est pas adapté aux écrans ou à l’impression. Il sera alors utilisé en traitement d’image, dans certains progiciels.

# Implémentation

Algorithmes utilisés

## Conversion entre les espaces de couleurs

Pour chacun des algorithmes de conversion de couleurs, le pseudo-code présenté sera concis, comme ce sont beaucoup de calculs mathématiques de base, enchainés et faisant appel à des concepts standards d’imagerie.

Ce qui est donc important est de la manière que nous les avons construits. Pour plus de précision sur les mathématiques derrière les calculs, le fichier de code Python *color\_conversion.py* sera important comme tout s’y trouve.

**rgb\_2\_hsv(r,g)**

• Normaliser les valeurs RGB dans la plage [0, 1]

• Déterminer les valeurs max et min des canaux RGB

• Calculer la différence (delta) entre max et min

• Calculer la teinte (hue) en fonction du canal RGB dominant

// Calcul de la teinte (H)

SI delta == 0 ALORS

h ← 0

SINON SI valeur\_max == r ALORS

h ← (60 × ((green - blue) / delta) + 360) Modulo 360

SINON SI c\_max == g ALORS

h ← (60 × ((blue - red) / delta) + 120) Modulo 360

SINON

h ← (60 × ((red - green) / delta) + 240) Modulo 360

FIN SI

• Calculer la saturation comme delta divisé par la valeur maximale

SI valeur\_max == 0 ALORS

s ← 0

SINON

s ← delta / c\_max

FIN SI

• La valeur (value) est la valeur maximale parmi RGB

// Calcul de la valeur (V)

v ← valeur\_max

Retourner H en degrés, S et V en pourcentage

**hsv\_2\_rgb**

• Normaliser S et V en [0, 1]

• Normaliser H en [0, 1]

• Calculer le chroma (C), la valeur X et le décalage m

• Déterminer les valeurs RGB en fonction de H

// Déterminer les valeurs RGB temporaires en fonction de h

SI 0 ≤ h < 1/6 ALORS

(r, g, b) ← (c, x, 0)

SINON SI 1/6 ≤ h < 2/6 ALORS

(r, g, b) ← (x, c, 0)

SINON SI 2/6 ≤ h < 3/6 ALORS

(r, g, b) ← (0, c, x)

SINON SI 3/6 ≤ h < 4/6 ALORS

(r, g, b) ← (0, x, c)

SINON SI 4/6 ≤ h < 5/6 ALORS

(r, g, b) ← (x, 0, c)

SINON

(r, g, b) ← (c, 0, x)

FIN SI

• Appliquer le décalage m et convertir en plage [0, 255]

• Retourner les valeurs RGB

**rgb\_2\_cmyk**

• Si RGB est (0,0,0), retourner (0, 0, 0, 100) (noir pur)

• Normaliser RGB dans la plage [0, 1]

• Calculer K (noir) comme 1 - max(R', G', B')

• Calculer C, M et Y en fonction de K

// Calcul de C, M, Y (éviter la division par zéro)

SI k < 1 ALORS

c ← (1 - red\_prime - k) / (1 - k)

m ← (1 - green\_prime - k) / (1 - k)

y ← (1 - blue\_prime - k) / (1 - k)

SINON

c, m, y = 0, 0, 0

FIN SI

• Convertir C, M, Y, K en pourcentage [0, 100]

• Retourner les valeurs CMYK

**cmyk\_2\_rgb**

• Normaliser CMYK dans la plage [0, 1]

• Calculer R, G, B en fonction des formules inverses

• Assurer que les valeurs sont dans [0, 255]

• Retourner les valeurs RGB

**normalize\_rgb**

Retourner les valeurs RGB normalisées entre [0, 1]

**denormalize\_rgb**

Convertir les valeurs RGB de [0, 1] en [0, 255]

Retourner les valeurs RGB

**rgb\_2\_lab**

• Convertir RGB en XYZ

rgb\_2\_xyz

• Convertir XYZ en LAB

xyz\_2\_lab

• Retourner LAB

**rgb\_2\_xyz**

• Normaliser RGB

• Appliquer la transformation gamma correction

r ← r / 12.92 SI r ≤ 0.04045 SINON ((r + 0.055) / 1.055) ^ 2.4

g ← g / 12.92 SI g ≤ 0.04045 SINON ((g + 0.055) / 1.055) ^ 2.4

b ← b / 12.92 SI b ≤ 0.04045 SINON ((b + 0.055) / 1.055) ^ 2.4

• Appliquer la matrice de transformation RGB -> XYZ

x ← 0.4124 × red + 0.3576 × green + 0.1805 × blue

y ← 0.2126 × red + 0.7152 × green + 0.0722 × blue

z ← 0.0193 × red + 0.1192 × green + 0.9505 × blue

• Retourner les valeurs XYZ

**xyz\_2\_lab**

• Normaliser XYZ avec les références D65

• Appliquer la transformation f(t)

SI t > (6/29)^3 ALORS

RETOURNER t^(1/3)

SINON

RETOURNER (t / (29^2)) + 4/29

FIN FONCTION

• Calculer L, A et B en utilisant les formules LAB

Retourner les valeurs LAB

**lab\_2\_rgb**

• Convertir LAB en XYZ

FONCTION lab\_2\_xyz

• Convertir XYZ en RGB

FONCTION xyz\_2\_rgb

Retourner RGB

**lab\_2\_xyz**

• Normaliser L, A, B

• Appliquer la transformation inverse f(t)

// Fonction inverse de f(t)

FONCTION f\_inv(t)

SI t > 6/29 ALORS

RETOURNER t^3

SINON

RETOURNER (t - 16/116) / 7.787

FIN FONCTION

• Convertir LAB en XYZ

Retourner XYZ

**xyz\_2\_rgb**

• Appliquer la matrice de transformation XYZ -> RGB

// Matrice de transformation XYZ → RGB

r ← 3.2406 × x - 1.5372 × y - 0.4986 × z

g ← -0.9689 × x + 1.8758 × y + 0.0415 × z

b ← 0.0556 × x - 0.2040 × y + 1.0570 × z

• Appliquer la transformation inverse gamma correction

• Convertir les valeurs en plage [0, 255]

Retourner les valeurs RGB

## Décompositions

La décomposition des couleurs repose sur un principe applicable à tous les espaces de couleurs. On commence par séparer l’image en ses différents canaux de couleur propres à de l’espace de couleur voulue (par exemple, les canaux **R, G, B** pour l’espace **RGB**). Par la suite, on mappe ces canaux à un dégradé d’une autre couleur voulue représentative, facilitant ainsi la visualisation de son intensité et de sa répartition au sein de l’image.

C’est une manière plus visuelle d’évaluer la composition de chacune des couleurs d’une image spécifique.

Initialiser un tableau CMYK avec les dimensions (hauteur de l’image, largeur de l’image, 4)

Pour chaque pixel (i, j) de l'image :

Extraire les valeurs RGB

Convertir RGB en CMYK

Stocker les valeurs CMYK dans le tableau

Pour chaque canal CMYK (Cyan, Magenta, Jaune, Noir) :

Sélectionner la carte de couleurs correspondante (GnBu, RdPu, YlOrRd, gray)

Afficher le canal en utilisant la carte de couleurs sélectionnée

Afficher l’image finale

## Transformations

Pour la transformation, nous convertissons l’image en format HSV, avant de jouer avec son contraste (*Saturation*) et sa luminosité (*Value*). Puis finalement nous la reconvertissions en RGB afin de l’afficher à l’écran. Le processus est le suivant.

Convertir l'image de RGB à HSV

Appliquer le contraste :

Multiplier la saturation (S) par le facteur de contraste

Limiter les valeurs entre 0 et 255

Appliquer la luminosité :

Ajouter la luminosité à la valeur (V)

Limiter les valeurs entre 0 et 255

Convertir l'image de HSV à RGB

Retourner l'image transformée

## Librairies utilisées

Nous avons utilisé les librairies OpenCV et Pillow. Nous allons présenter l’utilité de chacune de celle-ci dans notre projet.

Pour *OpenCV*, cette librairie a surtout permis de faire la troisième partie du laboratoire, soit celle des transformations. Afin de convertir notre image en format HSV et de modifier le contraste (le “S” ou la saturation de HSV) et la luminosité (le “V” ou la valeur), il était plus facile de le faire directement avec cette librairie, comme celle-ci implémente les algorithmes de conversion beaucoup plus efficacement que notre fichier de conversion de couleur *color\_conversion.py*. Elle nous a donc permis de convertir l’image en HSV, de séparer les trois composantes de cette espace de couleurs, de modifier le “S” et le “V”, puis de représenter l’image à l’utilisateur en format RGB.

La librairie *Pillow* nous a permis d’afficher la décomposition de chacun des canaux de couleurs pour chacun des espaces de couleurs demandés. Cette librairie nous permettait de prendre une image, de la séparer en chacun de ses canaux de couleurs, puis de faire correspondre chacun de ces canaux avec des *cmap (ColorMap)* prédéfinis de cette librairie, afin de bien pouvoir voir à quel point chacun des canaux de couleurs sont présents. Par exemple, pour la décomposition RGB, le canal de couleur rouge apparait plus ou moins rouge (donc allant du blanc au rouge très foncé) en fonction de la présence du rouge (*R*) dans les différents pixels de l’image voulue.

# Résultats et discussion

Résultats

## Curseur/Interpolation CMYK

Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

Description générée automatiquement

Figure 1 - Interface interpolation CMYK

## Curseur/Interpolation HSV

Une image contenant capture d’écran, Logiciel multimédia

Description générée automatiquement

Figure 2 - Interface interpolation HSV

## Curseur/Interpolation LAB

Une image contenant capture d’écran, Logiciel multimédia, logiciel

Description générée automatiquement

Figure 3 - Interface interpolation LAB

Commentaires :

Notre interface d'interpolation LAB fonctionne correctement dans la mesure où les valeurs RGB sont correctement converties en LAB, garantissant ainsi que la couleur modifiée correspond bien à celle souhaitée. Par exemple, un rouge pur en RGB (255,0,0) est fidèlement représenté en LAB (53.24, 80.09, 67.2). Cependant, lorsque les curseurs LAB sont ajustés en dessous d’environ 20, les valeurs RGB résultantes deviennent moins précises en raison des limites du calcul de conversion. Malgré cela, la couleur affichée à l’écran reste visuellement correcte.

## Décomposition CMYK

Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement

Figure 4 - Interface de décomposition CMYK

## Décomposition HSV

Une image contenant capture d’écran, texte, Logiciel multimédia, logiciel

Description générée automatiquement

Figure 5 - Interface de décomposition HSV

## Décomposition LAB

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 6 - Interface de décomposition LAB

## Transformations

Une image contenant capture d’écran, Logiciel multimédia, logiciel, texte

Description générée automatiquement

Figure 7 - Interface de transformations

## Difficultés rencontrées

**Conversion RGB en LAB**

Beaucoup de difficulté ont été rencontrés, surtout sur le comment appliquer les calculs mathématiques. Bien que les formules théoriques aient été disponibles dans les notes de cours, leur mise en œuvre sur des images s’est révélée plus complexe. L’un des principaux défis concernait l’espace de couleur LAB, qui repose sur des transformations non linéaires impliquant plusieurs étapes de conversion, notamment RGB → XYZ → LAB et vice-versa. L’application de ces formules dans le code demandait une compréhension approfondie des équations, en particulier la gestion des valeurs gamma et des seuils non linéaires dans la conversion RGB → XYZ.

**Démarche LAB:**

Pour surmonter cette difficulté, nous avons mené des recherches afin de trouver les équations précises correspondant à chaque étape de la conversion.

Une fois ces équations identifiées, nous les avons codées du mieux possible en veillant à respecter les seuils et transformations spécifiques à chaque espace de couleur.

Afin de valider nos conversions, nous avons comparé nos résultats avec ceux obtenus à l’aide de deux outils en ligne : [*Color Designer*](https://colordesigner.io/convert/labtorgb) et [*ColorMine*](https://colormine.org/convert/rgb-to-lab)*.*

**Transformation/Décomposition**

De plus, la transformation des images s'est avérée être un défi, notamment en raison de la lenteur du processus lorsque nous avons tenté d'utiliser les fonctions de conversion de couleurs que nous avions développées lors de la partie 2 du laboratoire. Nous avons donc décidé d’utiliser la librairie OpenCV avec ses fonctions préfaites. Cette approche nous a permis non seulement de simplifier l'opération, mais aussi d'obtenir des résultats bien plus rapidement et efficacement, tout en appliquant une solution que nous avions déjà expérimentée, mais dans un contexte légèrement différent. La décomposition des images a également représenté une étape complexe. Cependant, une fois que nous avons trouvé la méthode adéquate, cette tâche s'est réalisée de manière beaucoup plus fluide.

Principalement, les parties 2 et 3 du laboratoire ont été les plus délicates à gérer, présentant des problématiques plus complexes et nécessitant davantage d'adaptation et de réflexion pour parvenir à des résultats satisfaisants.

## Améliorations possibles ou éventuelles

Le cahier des charges aurait pu comprendre des fonctionnalités supplémentaires, comme l’ajout de certains espaces de couleur que nous n’avons pas utilisés. Nous aurions aussi pu implémenter certaines fonctions d’imagerie numérique comme le gamma, ou la méthode de Otsu, pour le contraste. Ce sont cependant des concepts complexes qui auraient représenté une autre section à part de ce laboratoire.

# Conclusion

Ce premier laboratoire nous a permis d'approfondir nos connaissances sur les espaces de couleurs. Tout d’abord, nous avons appris beaucoup de choses durant ce laboratoire, notamment sur la manipulation différentes représentations des couleurs (RGB, CMYK, HSV, LAB) et à comprendre leur pertinence dans des contextes spécifiques. De plus, nous vu comment bien appliquer les formules mathématiques théoriques dans un contexte plus réel et concret. De plus, ce laboratoire nous a permis de mieux comprendre les transformations d'images, telles que la modification de la luminosité et du contraste, qui sont essentielles pour l'amélioration de la qualité visuelle dans de nombreux logiciels professionnels. Nous avons aussi pris conscience des limites et des défis techniques rencontrés, en particulier lors de la décomposition des images et des ajustements des curseurs de transformation. Nous avons aussi vu comment certains espaces de couleurs peuvent interagir entre eux afin d’apporter des modifications à des images, ce qui fut très intéressant.

# Annexe : Manuel d’utilisateur

## Installation des dépendances

Pour installer les différentes librairies, on peut utiliser la commande suivant (avec l’utilitaire *pip*):

pip install numpy opencv-python matplotlib PyQt5

## Exécution du code

On exécute en terminal la commande Python:

python Lab\_1\_imagerie/GTI411 - LAB 1/main\_window.py