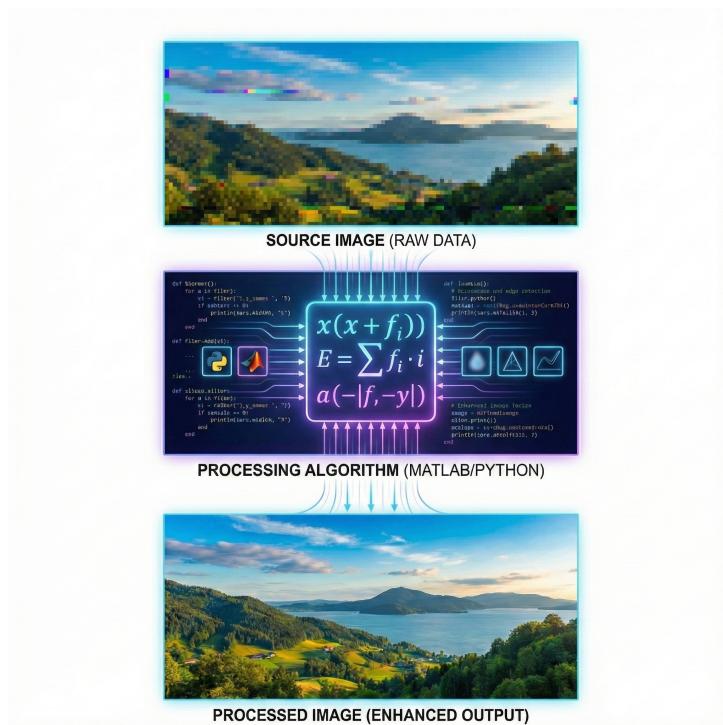


Mini-Projet de Traitement d'Images Numériques

MATLAB et Python



Réalisé par :
 Elghazi Haitam
 Elmoutaouakkil Noureddine
 Enfissi Monssif
 Abderrahmane El-Fennane
 Charaf Es-Sail

Encadré par :
 Pr. Hafiane

Supervisé par :
 Pr. Aboutabit

Table des matières

Introduction	3
Contexte du problème : Traitement des images	4
Le codage des images	5
1 Partie I : Opérations d'entrée et de sortie sur les images	6
1.1 Afficher une image	6
1.1.1 Python	6
1.1.2 MATLAB	6
1.1.3 Application	7
1.2 Ouvrir et sauvegarder une image	7
1.2.1 Python	7
1.2.2 MATLAB	7
1.2.3 Explication des fonctions utilisées	7
2 Partie II : Images Noir et Blanc	9
2.1 Création d'images	9
2.1.1 Python	9
2.1.2 MATLAB	9
2.2 Négatif	10
2.2.1 Python	10
2.2.2 MATLAB	10
2.2.3 Explication des fonctions utilisées	10
3 Partie III : Images en niveaux de gris	12
3.1 Luminance, contraste et profondeur	12
3.1.1 Python	12
3.1.2 MATLAB	12
3.1.3 Explication des fonctions utilisées	12
4 Partie IV : Opérations élémentaires sur images en mode gris	14
4.1 Reshape	14
4.1.1 Python	14
4.1.2 MATLAB	14
4.2 Inverser, Flip et Concaténation	14
4.2.1 Python	14
4.2.2 MATLAB	15
4.2.3 Explication des fonctions utilisées	15

5 Partie V : Images RGB	18
5.1 Initialisation et symétrie	18
5.1.1 Python	18
5.1.2 MATLAB	19
5.2 Conversion en niveaux de gris	19
5.2.1 Python	19
5.2.2 MATLAB	19
5.2.3 Explication des fonctions utilisées	19
Bilan détaillé	22
Difficultés rencontrées	23
Conclusion	24
Remerciements	25

Introduction

Dans le cadre de ce mini-projet, nous avons réalisé une étude complète sur le traitement des images numériques en utilisant deux environnements très répandus : MATLAB et Python. L'objectif principal est de comprendre la représentation des images, les opérations élémentaires possibles et la manipulation des images sous différentes formes (noir et blanc, niveaux de gris et RGB).

Ce projet nous a permis de consolider nos connaissances théoriques par une implémentation pratique et comparative.

Contexte du problème : Traitement des images

Le traitement d'images numériques est un domaine fondamental de l'informatique et de l'ingénierie. Il consiste à appliquer des algorithmes sur des images afin d'en extraire des informations, d'améliorer leur qualité ou de modifier leur contenu.

Les images sont largement utilisées dans des domaines tels que la médecine, la vision par ordinateur, la reconnaissance faciale et la robotique.

Le codage des images

Une image numérique est représentée sous forme de matrice de pixels. Chaque pixel contient une valeur qui dépend du type d'image :

- Noir et blanc : valeurs binaires (0 ou 1)
- Niveaux de gris : valeurs entre 0 et 255
- RGB : trois composantes (Rouge, Vert, Bleu)

Chapitre 1

Partie I : Opérations d'entrée et de sortie sur les images

1.1 Afficher une image

1.1.1 Python

Description : Cette fonction permet d'afficher une image sans les axes pour une meilleure lisibilité.

```
1 import matplotlib.pyplot as plt
2
3 def AfficherImg(img):
4     plt.axis("off")
5     plt.imshow(img, interpolation="nearest")
6     plt.show()
```

1.1.2 MATLAB

```
1 function AfficherImg(img)
2     imshow(img);
3 end
```

1.1.3 Application



FIGURE 1.1 – Affichage de l'image originale

1.2 Ouvrir et sauvegarder une image

1.2.1 Python

```
1 def ouvrirImage(chemin):
2     img = plt.imread(chemin)
3     return img
4
5 def saveImage(img):
6     plt.imsave("image1.png", img)
```

1.2.2 MATLAB

```
1 function img = Ouvrir(chemin)
2     img = imread(chemin);
3 end
4
5 function saveImage(img)
6     imwrite(img, 'image1.png');
7 end
```

1.2.3 Explication des fonctions utilisées

Dans cette partie, nous expliquons brièvement le rôle de chaque fonction employée dans les opérations d'entrée et de sortie sur les images.

Fonction AfficherImg

La fonction `AfficherImg` prend en argument une image sous forme de matrice et l'affiche à l'écran. Elle utilise la bibliothèque `matplotlib`, qui est une bibliothèque graphique très utilisée en Python pour la visualisation des données et des images.

L'affichage est réalisé sans axes afin d'améliorer la lisibilité de l'image, et la fonction `imshow` est utilisée pour représenter l'image à l'écran.

Fonction ouvrirImage

La fonction `ouvrirImage` utilise également la bibliothèque `matplotlib`. Elle permet de lire une image depuis le disque dur à partir de son chemin d'accès.

L'image est chargée en mémoire et retournée sous forme de matrice, ce qui permet par la suite d'effectuer différentes opérations de traitement d'images telles que la modification, l'analyse ou l'affichage.

Fonction saveImage

La fonction `saveImage` permet d'enregistrer une image sur le disque dur. Elle utilise la bibliothèque `matplotlib` pour sauvegarder l'image sous un format spécifié (par exemple PNG ou JPG).

Cette fonction est essentielle pour conserver les résultats obtenus après le traitement des images et permettre leur réutilisation ultérieure.

Chapitre 2

Partie II : Images Noir et Blanc

2.1 Création d'images

2.1.1 Python

```
1 import numpy as np
2
3 def image_noire(h, l):
4     return np.zeros((h, l))
5
6 def image_blanche(h, l):
7     return np.ones((h, l))
8
9 def creerImgBlancNoir(h, l):
10    return np.array([[[(i+j)%2 for i in range(l)] for j in range(h)]
11                   ])
```

2.1.2 MATLAB

```
1 function img = image_noire(h,l)
2     img = zeros(h,l);
3 end
4
5 function img = image_blanche(h,l)
6     img = ones(h,l);
7 end
8
9 function img = creerImgBlancNoir(h,l)
10    img = zeros(h,l);
11    for i=1:h
12        for j=1:l
13            img(i,j) = mod(i+j,2);
14        end
15    end
16 end
```

2.2 Négatif

2.2.1 Python

```
1 def negatif(img):
2     return 1 - img
```

2.2.2 MATLAB

```
1 function imgN = negatif(img)
2     imgN = 1 - img;
3 end
```

2.2.3 Explication des fonctions utilisées

Cette section présente une explication détaillée des différentes fonctions utilisées pour la création et la manipulation des images en noir et blanc.

Fonction `image_noire`

La fonction `image_noire` permet de générer une image entièrement noire. Elle prend en paramètres la hauteur h et la largeur l de l'image, et retourne une matrice de taille $(h \times l)$ dont tous les pixels sont initialisés à la valeur 0. Cette valeur correspond à la couleur noire dans une image binaire.

Fonction `image_blanche`

La fonction `image_blanche` est similaire à la fonction précédente. Elle génère une image de dimensions $(h \times l)$, mais initialise tous les pixels à la valeur 1, correspondant à la couleur blanche dans une image binaire.

Fonction `creerImgBlancNoir`

La fonction `creerImgBlancNoir` permet de créer une image noir et blanc de type damier. La valeur de chaque pixel est déterminée par la formule :

$$(i + j) \bmod 2$$

où i et j représentent respectivement les indices de ligne et de colonne. Cette formule permet d'alterner automatiquement les pixels noirs et blancs, créant ainsi un motif régulier.

Fonction `negatif`

La fonction `negatif` permet de construire le négatif d'une image binaire. Elle inverse les valeurs de chaque pixel : un pixel noir (0) devient blanc (1), et un pixel blanc (1) devient noir (0). Cette opération est réalisée simplement en soustrayant chaque pixel à 1.

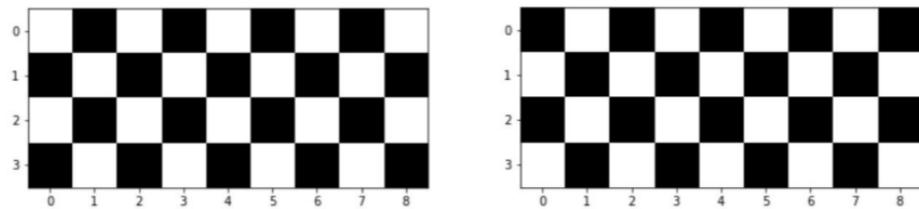


FIGURE 2.1 – Image principale et son négatif

Chapitre 3

Partie III : Images en niveaux de gris

3.1 Luminance, contraste et profondeur

3.1.1 Python

```
1 def luminance(img):
2     return img.mean()
3
4 def contraste(img):
5     mu = img.mean()
6     return ((img - mu)**2).mean()
7
8 def profondeur(img):
9     return img.max()
```

3.1.2 MATLAB

```
1 function L = luminance(img)
2     L = mean(img(:));
3 end
4
5 function C = contraste(img)
6     mu = mean(img(:));
7     C = mean((img(:)-mu).^2);
8 end
9
10 function p = profondeur(img)
11     p = max(img(:));
12 end
```

3.1.3 Explication des fonctions utilisées

Cette section présente les fonctions utilisées pour l'analyse des images en niveaux de gris. Ces fonctions permettent de calculer différentes statistiques décrivant le contenu radiométrique d'une image.

Fonction luminance

La fonction **luminance** retourne la luminance d'une image en niveaux de gris. La luminance correspond à la moyenne de tous les pixels de l'image. Elle permet d'évaluer la clarté globale de l'image : une valeur élevée indique une image plus claire, tandis qu'une valeur faible correspond à une image plus sombre.

Fonction contraste

La fonction **contraste** retourne le contraste d'une image en niveaux de gris. Le contraste est défini comme la variance des niveaux de gris de l'image. Il mesure la dispersion des valeurs des pixels autour de la moyenne. Un contraste élevé signifie que l'image contient des variations importantes d'intensité, tandis qu'un contraste faible indique une image plus uniforme.

Fonction profondeur

La fonction **profondeur** retourne la valeur maximale d'un pixel dans l'image. Cette valeur représente la profondeur maximale des niveaux de gris présents dans l'image et permet d'évaluer l'intensité la plus élevée contenue dans celle-ci.

Récapitulatif

Ces fonctions permettent de calculer des statistiques essentielles sur les images en niveaux de gris, telles que la luminance, le contraste et la profondeur. Elles sont utilisées pour analyser les caractéristiques visuelles des images et peuvent être combinées avec les fonctions de lecture d'images depuis le disque dur pour réaliser des traitements plus avancés.

Chapitre 4

Partie IV : Opérations élémentaires sur images en mode gris

4.1 Reshape

4.1.1 Python

```
1 def resharpe(img):
2     return img[:, :, 0]
```

4.1.2 MATLAB

```
1 function ImgC = resharpe(img)
2     ImgC = img(:, :, 1);
3 end
```

4.2 Inverser, Flip et Concaténation

4.2.1 Python

```
1 def inverser(img):
2     return 255 - img
3
4 def flipH(img):
5     return img[:, ::-1]
6
7 def flipV(img):
8     return img[::-1]
9
10 def poserV(img1, img2):
11     return img1 + img2
12
13 def poserH(img1, img2):
14     return [img1[i] + img2[i] for i in range(len(img1))]
```

4.2.2 MATLAB

```
1 function inv_img = inverser(img)
2     inv_img = 255 - img;
3 end
4
5 function output = flipH(img)
6     output = img(:,end:-1:1);
7 end
8
9 function output = flipV(img)
10    output = img(end:-1:1,:);
11 end
12
13 function output = poserV(img1,img2)
14    output = [img1; img2];
15 end
16
17 function output = poserH(img1,img2)
18    output = [img1 img2];
19 end
```

4.2.3 Explication des fonctions utilisées

Cette section présente une description détaillée des fonctions employées pour les transformations géométriques et les opérations de combinaison d'images.

Conversion d'une image couleur en image à un seul canal (`reshape`)

Le code utilisant la fonction `reshape` semble tenter de convertir une image à trois canaux (rouge, vert et bleu), généralement stockée dans un tableau NumPy de dimension $(h \times l \times 3)$, en une image à un seul canal. Cette opération permet de transformer une image couleur en une image en niveaux de gris ou en une représentation matricielle simplifiée, facilitant ainsi les traitements ultérieurs.

Fonction `inverser`

La fonction `inverser` renvoie l'image inverse de celle passée en argument. Elle réalise une inversion du ton de chaque pixel en soustrayant sa valeur au niveau maximal possible (par exemple 255 pour une image codée sur 8 bits). Cette opération permet d'obtenir le négatif de l'image originale.

Fonction `flipH`

La fonction `flipH` renvoie l'image transformée par une symétrie d'axe vertical passant par le milieu de l'image. Autrement dit, l'image est retournée horizontalement : la partie gauche devient la partie droite et inversement.

Fonction flipV

La fonction `flipV` renvoie l'image transformée par une symétrie d'axe horizontal passant par le milieu de l'image. Cette transformation correspond à un retournement vertical de l'image, où la partie supérieure devient la partie inférieure.

Fonction poserV

La fonction `poserV` prend en argument deux images de même largeur et de même profondeur. Elle renvoie une nouvelle image obtenue en posant la première image au-dessus de la seconde, ce qui correspond à une concaténation verticale des deux images.

Fonction poserH

La fonction `poserH` est similaire à la fonction précédente, mais réalise une concaténation horizontale. Elle prend deux images de même hauteur et de même profondeur, et renvoie une nouvelle image dans laquelle la seconde image est positionnée à droite de la première.

Récapititatif

Ces fonctions permettent de réaliser des transformations géométriques (inversion, symétries) ainsi que des opérations de combinaison d'images (concaténation verticale et horizontale). Elles constituent des outils fondamentaux pour le traitement et la manipulation des images numériques.



(a) Application de la fonction flipH



(b) Application de la fonction flipV



(c) Concaténation verticale



(d) Concaténation horizontale



(e) Image originale et son négatif

FIGURE 4.1 – Résultats des opérations de traitement d’images

Chapitre 5

Partie V : Images RGB

Introduction

Cette partie consiste en l'implémentation de trois fonctions permettant de manipuler des images RGB. Ces fonctions permettent respectivement d'initialiser une image couleur et d'appliquer des transformations de symétrie horizontale et verticale. Les images RGB sont largement utilisées en traitement d'images et en vision par ordinateur, car elles permettent de représenter une large gamme de couleurs proches de la perception humaine.

Représentation des couleurs en informatique

En informatique, une couleur est généralement représentée à partir du modèle RGB (*Red, Green, Blue*), basé sur le principe de la synthèse additive des couleurs. Dans ce modèle, chaque couleur est obtenue par la combinaison de trois composantes correspondant aux intensités de rouge, de vert et de bleu.

Ces composantes sont directement liées au spectre visible, qui correspond à la partie du spectre électromagnétique perceptible par l'œil humain. Le rouge, le vert et le bleu sont choisis comme couleurs primaires car leur combinaison permet de reconstruire la majorité des couleurs du spectre visible.

Chaque composante est généralement codée sur 8 bits, ce qui permet de représenter 256 niveaux d'intensité par canal, allant de 0 à 255. Ainsi, une image RGB est stockée sous la forme d'une matrice tridimensionnelle de taille $(h \times l \times 3)$, où chaque pixel contient un triplet (R, G, B) décrivant sa couleur.

La variation des intensités des trois canaux permet de produire différentes couleurs : par exemple, une intensité élevée de rouge combinée à de faibles intensités de vert et de bleu produit une couleur rouge, tandis que des intensités égales sur les trois canaux produisent des niveaux de gris.

5.1 Initialisation et symétrie

5.1.1 Python

```
1 from random import randrange
2
3 def initImageRGB(img):
4     for i in range(400):
```

```

5     img.append([])
6     for j in range(600):
7         img[i].append([randrange(256) for _ in range(3)])
8     return img
9
10    def symetrieV(img):
11        return img[::-1]
12
13    def symetrieH(img):
14        return img[:,::-1]

```

5.1.2 MATLAB

```

1 function img = initImageRGB(h,1)
2     img = randi([0 255],h,1,3);
3 end
4
5 function img_out = symetrieV(img)
6     img_out = flip(img,1);
7 end
8
9 function img_out = symetrieH(img)
10    img_out = flip(img,2);
11 end

```

5.2 Conversion en niveaux de gris

5.2.1 Python

```

1 def grayscale(img):
2     for i in range(len(img)):
3         for j in range(len(img[0])):
4             a = max(img[i][j])
5             b = min(img[i][j])
6             img[i][j] = (a+b)//2
7     return img

```

5.2.2 MATLAB

```

1 function imgG = grayscale(img)
2     imgG = uint8((max(img,[],3)+min(img,[],3))/2);
3 end

```

5.2.3 Explication des fonctions utilisées

Cette section décrit les fonctions implémentées pour la manipulation et le traitement des images RGB.

Fonction initImageRGB

La fonction `initImageRGB` permet d'initialiser une image RGB de manière aléatoire. Elle retourne un tableau à trois dimensions représentant une image couleur, où chaque pixel est défini par un triplet correspondant aux composantes rouge (R), verte (G) et bleue (B). Les valeurs des composantes sont générées aléatoirement dans l'intervalle [0, 255], ce qui permet de créer une image RGB contenant une grande variété de couleurs.

Fonction symetrie

La fonction `symetrie` retourne une image symétrique par rapport à l'axe horizontal ou à l'axe vertical de l'image passée en argument. Cette transformation géométrique permet de modifier l'orientation de l'image sans en altérer les informations de couleur, et elle est fréquemment utilisée dans les opérations de prétraitement et d'analyse d'images.

Fonction grayscale

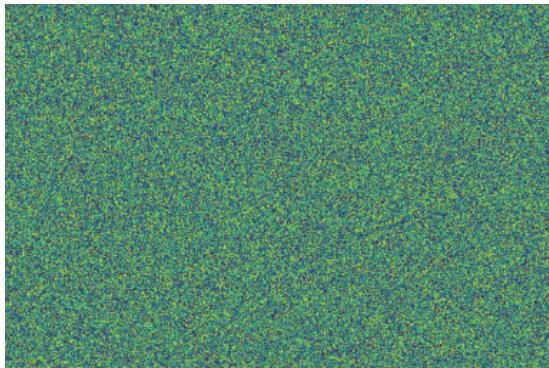
La fonction `grayscale` permet de convertir une image RGB en une image en niveaux de gris. Elle suit l'algorithme suivant : pour chaque pixel, les valeurs maximale et minimale sont recherchées parmi les trois composantes (R, G et B). Ensuite, la partie entière de la moyenne de ces deux valeurs est calculée. La valeur obtenue correspond au nouveau pixel en niveaux de gris, permettant ainsi de réduire l'image couleur à une représentation monochromatique tout en conservant une information pertinente sur la luminosité.



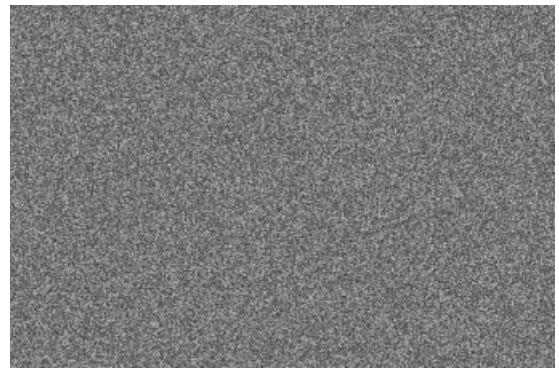
(a) application de fonction symetrie horizontale



(b) application de fonction symetrie verticale



(c) fonction initialserRGB



(d) fonction GRAYSCALE

FIGURE 5.1 – Conversion d'une image couleur en niveaux de gri

Bilan détaillé

Le programme a été structuré en plusieurs parties distinctes, chacune correspondant à un ensemble cohérent de fonctions assurant un rôle précis dans le traitement des images.

La première partie est consacrée aux opérations d'entrée et de sortie sur les images. Elle regroupe les fonctions `AfficherImg`, `ouvrirImage` et `saveImage`, qui permettent respectivement d'afficher, de charger et d'enregistrer des images.

La deuxième partie traite des images en noir et blanc et comprend les fonctions `image_noire`, `image_blanche`, `creerImgBlancNoir` et `negatif`, dédiées à la génération et à la manipulation d'images binaires.

La troisième partie concerne les images en niveaux de gris et regroupe les fonctions `luminance`, `contraste` et `profondeur`, permettant le calcul de caractéristiques statistiques des images.

La quatrième partie porte sur les opérations élémentaires appliquées aux images en niveaux de gris, notamment à travers les fonctions `inverser`, `flipH`, `poserV` et `poserH`, qui réalisent des transformations géométriques et des opérations de combinaison d'images.

Enfin, la dernière partie est consacrée aux images RGB et inclut les fonctions `initImageRGB`, `symetrie` et `grayscale`, permettant l'initialisation, la transformation et la conversion des images couleur.

L'utilisation du programme repose sur l'appel des fonctions appropriées en fournissant les arguments nécessaires. Le travail réalisé a consisté en l'implémentation des différentes fonctions définies dans le cahier des charges. Certaines d'entre elles se sont révélées plus complexes à mettre en œuvre, en particulier celles impliquant des opérations de calcul sur les pixels des images. Toutefois, grâce à l'exploitation de la documentation disponible et à l'étude d'exemples pertinents, ces difficultés ont pu être surmontées. Dans l'ensemble, les objectifs fixés ont été atteints et les consignes du cahier des charges ont été respectées, ce qui permet de considérer le résultat obtenu comme satisfaisant.

Difficultés rencontrées

Nous avons rencontré des difficultés liées à la gestion des dimensions des images, aux différences de types de données entre MATLAB et Python, ainsi qu'à l'affichage correct des images.

Conclusion

Conclusion

Au terme de ce travail, il apparaît que MATLAB et Python constituent deux outils puissants et complémentaires pour le traitement d'images. MATLAB se distingue par sa simplicité d'utilisation et son environnement intégré, particulièrement adapté au prototypage rapide et à la manipulation matricielle. Ses fonctions natives dédiées au traitement d'images permettent une implémentation efficace et intuitive des algorithmes.

Python, quant à lui, offre une grande flexibilité et une richesse importante de bibliothèques telles que `NumPy`, `Matplotlib` et `OpenCV`. Son caractère open-source et sa large communauté en font un outil privilégié pour le développement d'applications plus complexes et pour l'intégration du traitement d'images dans des projets de plus grande envergure.

Ainsi, le choix entre MATLAB et Python dépend essentiellement du contexte d'utilisation : MATLAB est particulièrement adapté à l'enseignement et au prototypage scientifique, tandis que Python est plus approprié pour le développement, la portabilité et le déploiement d'applications réelles. L'utilisation conjointe de ces deux environnements permet de tirer profit des avantages de chacun et d'enrichir l'approche du traitement d'images.

Remerciements

Nous remercions notre encadrant pour son accompagnement, ainsi que toutes les personnes ayant contribué à la réussite de ce projet.