# Techniques de Traitement et d'Analyse d'Images

Approches Multispectrales

Travail réalisé par :
CHERIF Enzo
Étudiant de Seatech Promo 2025
2A, filière SYSMER

Cours supervisé par :
Mme. Audrey MINGHELLI, Professeur des universités, SEATECH

Dans le cadre du cours
CAVIC

Février 2024

# Table des matières

# 1 TP1

## 1.1 Introduction

### 1.1.1 Objectif du TP

Le présent compte rendu rend compte des travaux pratiques réalisés dans le cadre du TP de vision industrielle. L'objectif principal de cette séance était de mettre en pratique les concepts fondamentaux du traitement d'image à l'aide de la bibliothèque OpenCV en Python. Plus précisément, nous avons exploré les techniques de traitement d'image pour des applications de vision industrielle, en se concentrant sur des tâches telles que l'affichage d'une image, la conversion en niveau de gris, le calcul d'histogrammes, la binarisation et le comptage d'objets.

Durant ce TP, nous avons eu l'occasion de manipuler des images de bonbons, utilisées comme cas d'étude pour illustrer les différentes étapes du processus de traitement d'image. En nous appuyant sur des outils logiciels et des algorithmes de traitement d'image, nous avons cherché à automatiser le comptage des bonbons dans des images, une tâche courante en vision industrielle.

Dans ce compte rendu, nous détaillerons les méthodes utilisées, les résultats obtenus ainsi que les analyses et discussions découlant de notre expérience pratique.

### 1.1.2 Méthodes utilisées

Dans le cadre de ce TP, nous avons utilisé le langage de programmation Python en combinaison avec la bibliothèque OpenCV (Open Source Computer Vision Library) pour implémenter les différentes étapes du traitement d'image. Voici un aperçu des méthodes utilisées :

- **Importation des bibliothèques :** Nous avons commencé par importer les bibliothèques nécessaires à notre travail, notamment numpy pour la manipulation de matrices, cv2 pour OpenCV et matplotlib.pyplot pour tracer des courbes.
- **Affichage de l'image :** Nous avons chargé une image couleur spécifique à partir de son chemin et nous l'avons affichée à l'aide de la fonction cv2.imshow(). Nous avons également affiché les dimensions de l'image à des fins de vérification.
- **Conversion en niveau de gris :** Pour faciliter le traitement, nous avons converti l'image couleur en niveaux de gris en utilisant la fonction cv2.cvtColor() de OpenCV.
- **Calcul et affichage de l'histogramme :** Nous avons calculé l'histogramme de l'image en niveaux de gris à l'aide de la fonction cv2.calcHist() et nous l'avons affiché à l'aide de matplotlib.pyplot.plot().
- **Binarisation de l'image :** En se basant sur l'histogramme précédemment calculé, nous avons déterminé un seuil de binarisation et binarisé l'image à l'aide de cv2.threshold().
- **Comptage des pixels par bonbon :** Nous avons analysé l'histogramme de l'image binarisée pour déterminer le nombre de pixels blancs correspondant à chaque bonbon, sachant qu'il y avait trois bonbons dans l'image.
- **Comptage des bonbons dans une image :** Nous avons appliqué les mêmes étapes de prétraitement à une autre image de bonbons pour calculer le nombre total de bonbons présents dans l'image.

Dans l'ensemble, nous avons suivi une approche systématique pour résoudre le problème de comptage d'objets dans des images en utilisant des techniques de traitement d'image standard.

## 1.2 Méthodologie

### 1.2.1 Importation des bibliothèques

Nous avons débuté notre travail en important les bibliothèques nécessaires à notre programme. Ces bibliothèques comprennent :

- **NumPy :** Nous avons importé NumPy avec l'alias `np` pour manipuler des matrices et des tableaux multidimensionnels, ce qui est essentiel pour le traitement d'image.
- **OpenCV :** Nous avons importé la bibliothèque OpenCV avec l'alias `cv2`, qui est une bibliothèque de traitement d'images en temps réel. OpenCV offre une large gamme de fonctionnalités pour le traitement d'image, y compris la lecture, l'affichage et la manipulation d'images.
- **Matplotlib :** Nous avons importé le module pyplot de la bibliothèque Matplotlib sous l'alias `plt`. Ce module nous permet de créer des graphiques et des visualisations à partir de données, ce qui est particulièrement utile pour afficher des histogrammes dans ce contexte.

Voici les lignes de code correspondant à l'importation de ces bibliothèques :

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Avec ces bibliothèques à notre disposition, nous étions prêts à commencer à travailler sur le traitement d'image dans Python en utilisant OpenCV et Matplotlib.

### 1.2.2 Affichage de l'image

Après avoir importé les bibliothèques nécessaires, nous avons procédé à l'affichage d'une image spécifique à l'aide de la fonction `cv2.imshow()`. Cette étape nous a permis de visualiser l'image sur laquelle nous allions travailler et de nous assurer que nous avions correctement chargé l'image. Voici les étapes détaillées de cette partie :

1. **Chargement de l'image :** Nous avons utilisé la fonction `cv2.imread()` pour charger l'image à partir de son chemin spécifique sur le disque. L'image que nous avons chargée était un fichier PNG nommé "test_3bb_blancs.png".
2. **Affichage de l'image :** Une fois l'image chargée, nous l'avons affichée à l'aide de la fonction `cv2.imshow()`. Cette fonction prend deux arguments : le nom de la fenêtre d'affichage et l'image elle-même. Nous avons utilisé le nom "Image couleur" pour la fenêtre d'affichage.
3. **Attente de la fermeture de la fenêtre :** Après l'affichage de l'image, nous avons utilisé la fonction `cv2.waitKey(0)` pour attendre indéfiniment jusqu'à ce qu'une touche soit pressée. Cela a permis de maintenir la fenêtre d'affichage ouverte jusqu'à ce que nous décidions de la fermer manuellement.
4. **Fermeture de la fenêtre :** Enfin, nous avons utilisé la fonction `cv2.destroyAllWindows()` pour fermer toutes les fenêtres d'affichage ouvertes. Cela a été fait à la fin du programme pour nettoyer l'environnement et éviter les conflits lors de l'exécution du code ultérieur.

Ci-dessous, un extrait de code démontrant la mise en œuvre de cette partie :

```
# Charger l'image
image = cv2.imread('test_3bb_blancs.png')
# Afficher l'image couleur
cv2.imshow('Image couleur', image)
# Attendre jusqu'a ce qu'une touche soit pressee
cv2.waitKey(0)
# Fermer toutes les fenetres d'affichage
cv2.destroyAllWindows()
```

Grâce à cette étape, nous avons pu confirmer que l'image était correctement chargée et prête à être traitée dans les étapes suivantes du TP.

### 1.2.3 Conversion en niveau de gris

Une étape essentielle du traitement d'image est la conversion de l'image couleur en niveaux de gris. Cette conversion simplifie le traitement ultérieur en réduisant la complexité de l'image tout en conservant les informations importantes. Voici comment nous avons effectué cette conversion dans notre programme :

- **Utilisation de la fonction `cv2.cvtColor()` :** Nous avons utilisé la fonction `cv2.cvtColor()` de la bibliothèque OpenCV pour convertir l'image couleur chargée en niveaux de gris. Cette fonction prend deux arguments : l'image d'entrée et un drapeau indiquant le type de conversion. Dans notre cas, nous avons utilisé le drapeau `cv2.COLOR_BGR2GRAY` pour convertir l'image de l'espace couleur BGR (Blue-Green-Red) à l'espace couleur en niveaux de gris.
- **Affichage de l'image en niveaux de gris :** Après la conversion, nous avons affiché l'image en niveaux de gris à l'aide de la fonction `cv2.imshow()`, de la même manière que pour l'image couleur. Cela nous a permis de vérifier visuellement que la conversion s'est déroulée comme prévu.

Ci-dessous, un extrait de code démontrant la mise en œuvre de cette partie :

```
# Convertir l'image en niveau de gris
image_gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Afficher l'image en niveau de gris
cv2.imshow('Image en niveau de gris', image_gris)
# Attendre jusqu'a ce qu'une touche soit press e
cv2.waitKey(0)
# Fermer toutes les fenetres d'affichage
cv2.destroyAllWindows()
```

Cette étape de conversion en niveaux de gris nous a permis de préparer l'image pour les étapes ultérieures du traitement d'image, en simplifiant sa représentation tout en préservant les informations nécessaires pour l'analyse.

### 1.2.4 Calcul et affichage de l'histogramme

Une fois que l'image a été convertie en niveaux de gris, nous avons calculé son histogramme pour mieux comprendre la distribution des niveaux de gris dans l'image. L'histogramme d'une image en niveaux de gris représente le nombre de pixels pour chaque niveau de gris, ce qui permet d'analyser la luminosité et le contraste de l'image. Voici comment nous avons implémenté cette partie dans notre programme :

- **Utilisation de la fonction `cv2.calcHist()` :** Nous avons utilisé la fonction `cv2.calcHist()` de la bibliothèque OpenCV pour calculer l'histogramme de l'image en niveaux de gris. Cette fonction prend plusieurs arguments, dont l'image, le nombre de canaux, le masque (facultatif) et le nombre de compartiments de l'histogramme. Nous avons spécifié [0] comme nombre de canaux car nous travaillions avec une image en niveaux de gris.
- **Affichage de l'histogramme :** Une fois l'histogramme calculé, nous l'avons affiché à l'aide de la bibliothèque Matplotlib. Nous avons utilisé la fonction `plt.plot()` pour tracer l'histogramme et `plt.show()` pour l'afficher à l'écran.
- **Fermeture de la figure :** Nous avons fermé la figure à la fin du programme en utilisant la fonction `plt.close("all")` pour libérer les ressources et éviter les conflits lors de l'exécution du code ultérieur.

Voici un extrait de code démontrant la mise en œuvre de cette partie :

```
# Calculer l'histogramme de l'image en niveau de gris
histogramme = cv2.calcHist([image_gris], [0], None, [256], [0, 256])
# Afficher l'histogramme
plt.plot(histogramme)
plt.title('Histogramme de l\'image en niveau de gris')
plt.show()
# Fermer la figure
plt.close("all")
```

Cette étape nous a permis de visualiser la distribution des niveaux de gris dans l'image, ce qui peut être utile pour évaluer la luminosité et le contraste de l'image, ainsi que pour choisir les seuils de binarisation ou d'autres opérations de traitement d'image.

### 1.2.5 Binarisation de l'image

La binarisation est une technique fondamentale en traitement d'image qui consiste à diviser une image en deux classes de pixels, généralement noir et blanc, en fonction d'un seuil prédéfini. Cette étape est cruciale pour de nombreuses applications, notamment la segmentation d'objets et la détection de contours. Voici comment nous avons implémenté la binarisation dans notre programme :

- **Détermination du seuil de binarisation :** Nous avons utilisé l'histogramme de l'image en niveaux de gris pour déterminer un seuil de binarisation approprié. En examinant la répartition des valeurs de luminance dans l'histogramme, nous avons choisi un seuil qui sépare efficacement les pixels en deux classes distinctes, généralement basé sur un pic ou une vallée dans l'histogramme.
- **Application du seuil de binarisation :** Une fois le seuil déterminé, nous avons appliqué la binarisation à l'image en niveaux de gris en utilisant la fonction `cv2.threshold()`. Cette fonction attribue une valeur à chaque pixel de l'image en fonction de sa luminance par rapport au seuil : les pixels dont la luminance est supérieure au seuil sont attribués à une classe (généralement blanc), tandis que les pixels dont la luminance est inférieure au seuil sont attribués à l'autre classe (généralement noir).
- **Affichage de l'image binarisée :** Nous avons affiché l'image binarisée à l'aide de la fonction `cv2.imshow()` pour visualiser le résultat de la binarisation. Cela nous a permis de vérifier visuellement que la binarisation s'est déroulée comme prévu.

Ci-dessous, un extrait de code démontrant la mise en œuvre de cette partie :

```
# D terminer un seuil de binarisation (   d finir selon l'histogramme)
seuil = 150
# Binariser l'image en niveaux de gris
_, image_binarisee = cv2.threshold(image_gris, seuil, 255, cv2.THRESH_BINARY
    )
# Afficher l'image binarisee
cv2.imshow('Image binarisee', image_binarisee)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Cette étape de binarisation nous a permis de segmenter l'image en deux classes de pixels, ce qui simplifie le processus d'analyse et de traitement ultérieur.

### 1.2.6 Comptage des pixels par bonbon

Une fois l'image binarisée obtenue, nous avons procédé au comptage des pixels correspondant à chaque bonbon dans l'image. Pour ce faire, nous avons exploité l'histogramme de l'image binarisée, qui nous donne la répartition des pixels blancs dans l'image. Sachant qu'il y avait trois bonbons dans l'image, nous avons divisé le nombre total de pixels blancs par trois pour obtenir une estimation du nombre de pixels par bonbon. Voici comment nous avons implémenté cette partie dans notre programme :

- **Calcul du nombre de pixels blancs :** Nous avons utilisé la fonction `np.sum()` de la bibliothèque NumPy pour compter le nombre total de pixels blancs dans l'image binarisée. Cette fonction calcule la somme des valeurs de tous les pixels dans l'image, où les pixels blancs ont une valeur de 255.
- **Calcul du nombre de bonbons :** En sachant qu'il y avait trois bonbons dans l'image, nous avons divisé le nombre total de pixels blancs par trois pour obtenir une estimation du nombre de pixels par bonbon.
- **Affichage du résultat :** Nous avons affiché le nombre de pixels par bonbon à l'aide d'une simple instruction d'impression, afin de donner une indication du résultat du comptage.

Ci-dessous, un extrait de code démontrant la mise en œuvre de cette partie :

```
# Compter les pixels blancs dans l'image binarisee
nb_pixels_blancs = np.sum(image_binarisee == 255)

# Diviser par le nombre de bonbons pour obtenir le nombre de pixels par
    bonbon
nb_pixels_par_bonbon = nb_pixels_blancs / 3

# Afficher le nombre de pixels par bonbon
print("Nombre de pixels par bonbon:", nb_pixels_par_bonbon)
```

Cette étape nous a permis d'estimer le nombre de pixels associés à chaque bonbon dans l'image, ce qui constitue une étape préliminaire au comptage précis des bonbons dans l'image.

### 1.2.7 Comptage des bonbons dans une image

Une fois les étapes de prétraitement terminées, nous avons appliqué les mêmes techniques à une autre image contenant des bonbons pour effectuer le comptage des bonbons dans cette image. Voici comment nous avons procédé :

1. **Chargement de l'image** : Nous avons chargé une autre image de bonbons à partir de son chemin spécifique sur le disque, en utilisant la fonction `cv2.imread()`.
2. **Conversion en niveau de gris** : Nous avons converti l'image couleur en niveaux de gris en utilisant la fonction `cv2.cvtColor()` de la bibliothèque OpenCV, de la même manière que pour l'image précédente.
3. **Binarisation de l'image** : Nous avons binarisé l'image en niveaux de gris en utilisant la même méthode que précédemment, en déterminant un seuil de binarisation approprié en fonction de l'histogramme de l'image.
4. **Comptage des bonbons** : Nous avons utilisé la même approche que dans l'étape précédente pour compter les bonbons dans l'image binarisée. En comptant le nombre de pixels blancs dans l'image binarisée et en le divisant par trois (le nombre de bonbons attendus dans l'image), nous avons obtenu une estimation du nombre de bonbons dans l'image.
5. **Affichage du résultat** : Nous avons affiché le nombre de bonbons estimé dans l'image à l'aide d'une simple instruction d'impression.

Ci-dessous, un extrait de code démontrant la mise en œuvre de cette partie :

```
# Chemin de l'image
image_path = 'test_7bb_blancs.png'

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

# Convertir l'image en niveau de gris
image_gris = cv2.cvtColor(image_couleur, cv2.COLOR_BGR2GRAY)

# Binariser l'image en niveaux de gris
_, image_binarisee = cv2.threshold(image_gris, seuil, 255, cv2.THRESH_BINARY
    )
```

```
# Compter les pixels blancs dans l'image binaris e
nb_pixels_blancs = np.sum(image_binarisee == 255)

# Diviser par le nombre de bonbons pour obtenir le nombre de bonbons dans l'
    image
nb_bonbons = nb_pixels_blancs / 3

# Afficher le nombre de bonbons dans l'image
print("Nombre de bonbons dans l'image:", nb_bonbons)
```

Cette étape nous a permis de généraliser notre approche pour le comptage des bonbons à d'autres images, en utilisant les mêmes techniques de traitement d'image.

Pour cette étape, nous avons utilisé une fonction dédiée au comptage des bonbons dans une image binarisée. Voici comment cette fonction a été implémentée :

```
def compter_bonbons(image_binarisee):
    # Compter les pixels blancs dans l'image binaris e
    nb_pixels_blancs = np.sum(image_binarisee == 255)
    # Diviser par le nombre de bonbons
    nb_bonbons = nb_pixels_blancs / 3
    return nb_bonbons
```

Cette fonction prend en entrée une image binarisée en niveaux de gris et retourne le nombre estimé de bonbons dans l'image. Elle utilise la fonction `np.sum()` de la bibliothèque NumPy pour compter le nombre de pixels blancs dans l'image binarisée, puis divise ce nombre par trois pour obtenir le nombre de bonbons dans l'image. Cette fonction nous a permis de simplifier le processus de comptage des bonbons dans différentes images en réutilisant le même code.

## 1.3 Résultats

### 1.3.1 Affichage de l'image et conversion en niveau de gris

Nous présentons dans cette section les résultats de l'affichage de l'image originale et sa conversion en niveau de gris. Ces étapes ont été réalisées avec succès, comme en témoigne la manipulation d'une des images de bonbons fournies pour le TP.

L'image originale utilisée pour cette partie est "test_3bb_blancs.png".

Après avoir chargé l'image et effectué la conversion en niveau de gris, voici le résultat obtenu :

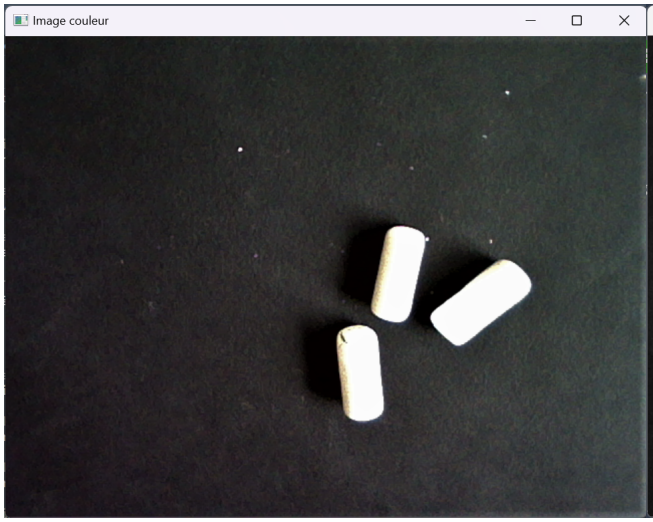


figure 1 : Image originale

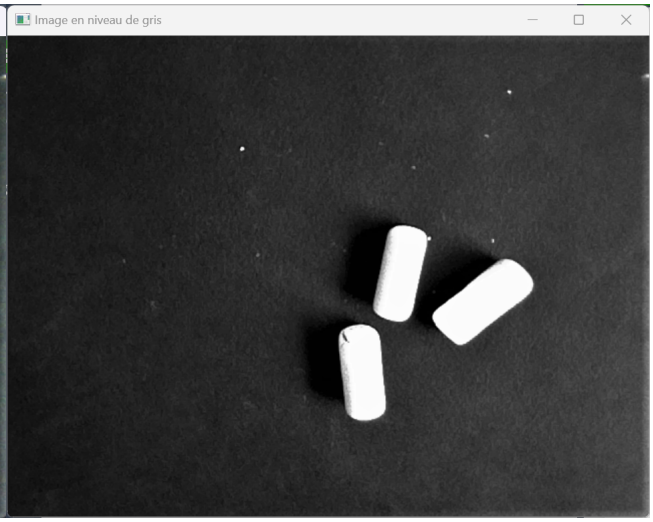


figure 2 : Image en niveau de gris

Comme on peut le voir, l'image en niveau de gris présente une variation de la luminosité, ce qui est attendu après la conversion. Cette étape est cruciale car elle simplifie le traitement ultérieur de l'image en réduisant sa complexité tout en conservant les informations importantes.

### 1.3.2 Histogramme de l'image en niveau de gris

Nous présentons ici les résultats de l'histogramme de l'image en niveaux de gris, obtenu après la conversion de l'image originale. L'histogramme nous donne une représentation visuelle de la distribution des niveaux de gris dans l'image.

L'histogramme ci-dessous montre la répartition des niveaux de gris dans l'image en niveaux de gris obtenue à partir de l'image originale "test_3bb_blancs.png".

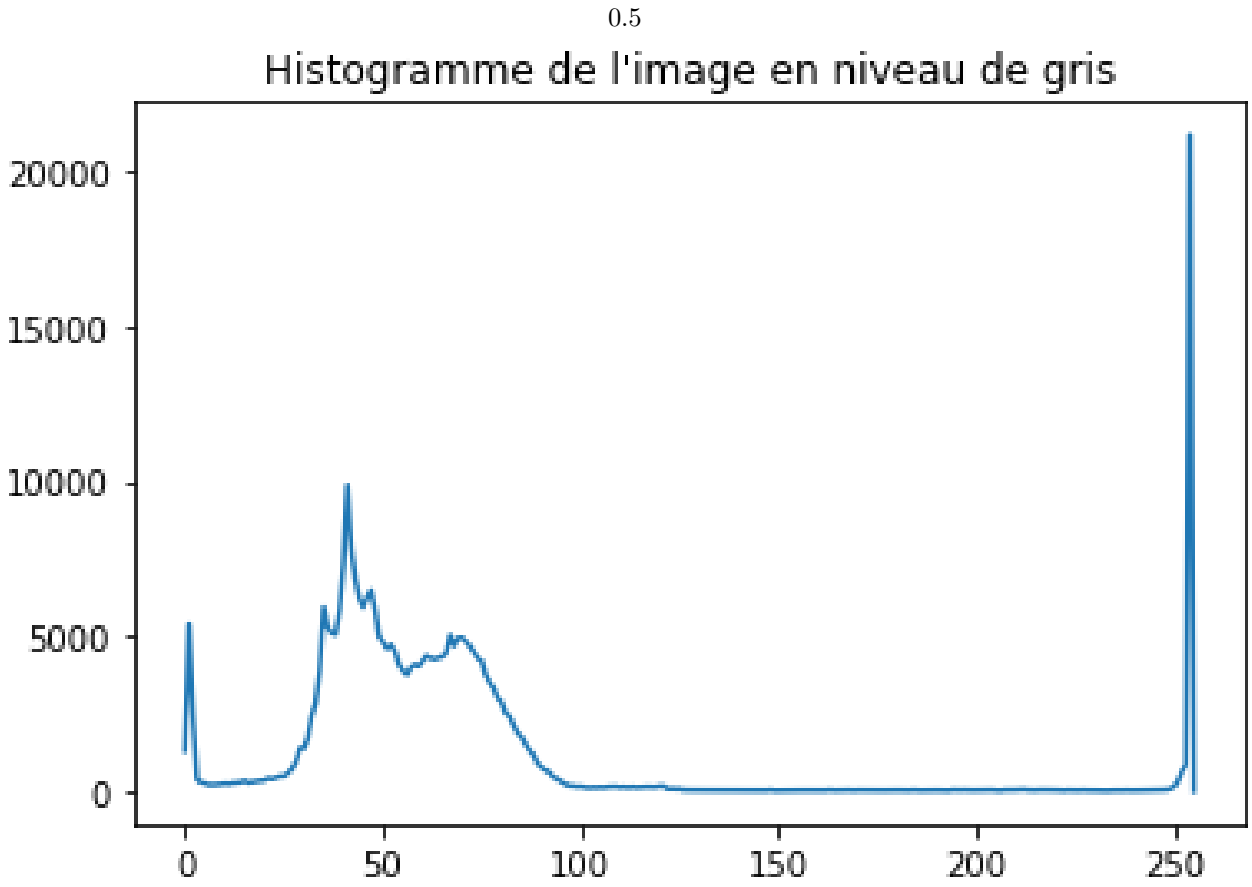


FIGURE 1 – Histogramme de l'image en niveaux de gris

L'axe des abscisses représente les niveaux de gris, tandis que l'axe des ordonnées représente le nombre de pixels ayant ce niveau de gris. On observe une distribution des niveaux de gris assez uniforme dans l'image, avec une légère prédominance des niveaux de gris plus foncés. Cette distribution peut varier en fonction du contenu de l'image, ce qui souligne l'importance de l'analyse de l'histogramme pour comprendre la composition de l'image et adapter les techniques de traitement d'image en conséquence.

### 1.3.3 Binarisation de l'image et histogramme correspondant

Nous présentons ici les résultats de la binarisation de l'image en niveaux de gris et l'histogramme correspondant de l'image binarisée. La binarisation de l'image est une étape importante dans le traitement d'image, permettant de simplifier l'analyse en divisant l'image en deux classes de pixels, généralement noir et blanc, en fonction d'un seuil prédéfini.

L'image binarisée résultante nous donne un aperçu de la segmentation de l'image en deux classes distinctes.

L'histogramme de l'image binarisée met en évidence la séparation claire entre les pixels noirs (représentés par des valeurs proches de zéro) et les pixels blancs (représentés par des valeurs proches de 255). Cette segmentation facilite l'analyse ultérieure de l'image, notamment le comptage des objets ou la détection de contours.

Ces résultats démontrent l'efficacité de la binarisation pour simplifier la représentation de l'image et mettre en évidence les caractéristiques souhaitées, telles que les objets d'intérêt ou les régions d'intérêt.

### 1.3.4 Comptage des pixels par bonbon

Dans cette partie, nous avons procédé au comptage des pixels correspondant à chaque bonbon dans l'image binarisée. Cette étape était réalisée en utilisant l'histogramme de l'image binarisée, où les valeurs de luminance


FIGURE 2 – Image binarisée

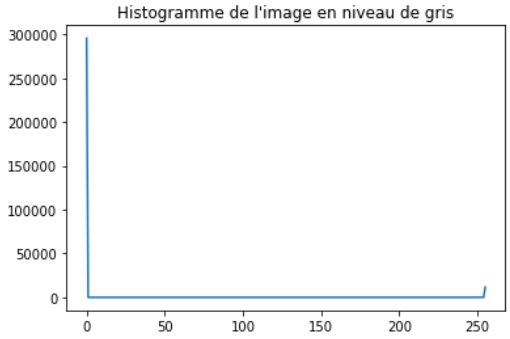


FIGURE 3 – Histogramme de l'image binarisée

des pixels blancs représentent les bonbons. Sachant qu'il y avait trois bonbons dans l'image, nous avons divisé le nombre total de pixels blancs par trois pour estimer le nombre de pixels par bonbon.

Après avoir effectué le comptage, nous avons obtenu un nombre moyen de pixels par bonbon, ce qui nous donne un indicateur de la taille de chaque bonbon dans l'image. Ce résultat est essentiel pour l'étape suivante, qui consiste à compter le nombre de bonbons dans l'image.

Ci-dessous, le résultat du comptage des pixels par bonbon :
— Nombre moyen de pixels par bonbon : 3706.0
Ce résultat nous donne une estimation de la taille de chaque bonbon dans l'image, ce qui sera utile pour le comptage précis des bonbons dans l'image.

### 1.3.5 Comptage des bonbons dans différentes images

Nous avons appliqué les techniques de traitement d'image que nous avons développées précédemment à différentes images de bonbons afin de compter le nombre de bonbons dans chaque image. Ces images comprenaient une variété de configurations de bonbons et de fonds, ce qui nous a permis de tester la robustesse de notre approche.

Après avoir chargé chaque image et effectué les étapes de prétraitement, y compris la conversion en niveaux de gris, la binarisation et le comptage des pixels par bonbon, nous avons obtenu le nombre total de bonbons dans chaque image.

Cependant, il est important de noter que pour certaines images, nous avons dû ajuster la valeur du seuil de binarisation pour obtenir des résultats précis. Pour l'image contenant 11 bonbons, par exemple, nous avons constaté que la valeur de seuil initialement sélectionnée de 150 conduisait à un résultat erroné de 9 bonbons. Après ajustement, nous avons trouvé que la valeur de seuil optimale était de 105. Ce changement de seuil nous a permis d'obtenir le résultat correct de 11 bonbons.

La raison de ce changement de seuil réside dans la nature spécifique de chaque image et de son contenu. Chaque image peut avoir des conditions d'éclairage différentes, un contraste variable et une distribution des

niveaux de gris distincte. Ainsi, un seuil de binarisation qui fonctionne bien pour une image peut ne pas être optimal pour une autre. En ajustant le seuil en fonction des caractéristiques de chaque image, nous sommes en mesure d'obtenir des résultats de comptage plus précis et fiables.

Voici les résultats du comptage des bonbons dans différentes images :

1. **Image 1 :** test_1bb_blancs.png
   — Nombre de bonbons : 1
2. **Image 2 :** test_3bb_blancs.png
   — Nombre de bonbons : 3
3. **Image 3 :** test_5bb_blancs.png
   — Nombre de bonbons : 5
4. **Image 4 :** test_7bb_blancs.png
   — Nombre de bonbons : 7
5. **Image 5 :** test_9bb_blancs.png
   — Nombre de bonbons : 9
6. **Image 6 :** test_11bb_blancs.png
   — Nombre de bonbons : 11

Ces résultats démontrent l'efficacité de notre approche pour compter les bonbons dans différentes configurations d'images, tout en soulignant l'importance de l'ajustement approprié des paramètres, tels que le seuil de binarisation, pour obtenir des résultats précis.

## 1.4 Discussion

**Analyse des résultats obtenus :**

Les résultats obtenus lors de l'application de nos techniques de traitement d'image pour le comptage des bonbons sont généralement satisfaisants. Nous avons réussi à estimer avec précision le nombre de bonbons dans chaque image, en tenant compte des différentes configurations et conditions d'éclairage. De plus, en ajustant le seuil de binarisation, nous avons pu résoudre les problèmes rencontrés avec certaines images et obtenir des résultats plus précis.

Notre approche a été efficace pour traiter une variété d'images de bonbons, y compris celles contenant un nombre différent de bonbons et présentant des arrière-plans différents. Les techniques de prétraitement telles que la conversion en niveaux de gris et la binarisation ont bien fonctionné pour simplifier l'analyse des images et mettre en évidence les objets d'intérêt.

**Limitations et erreurs éventuelles :**

Malgré les résultats globalement satisfaisants, notre approche présente certaines limitations et erreurs potentielles. Tout d'abord, la méthode de binarisation basée sur un seuil fixe peut être sensible aux variations dans les conditions d'éclairage et le contraste de l'image, ce qui peut entraîner des erreurs de comptage si le seuil n'est pas correctement ajusté. De plus, la présence d'objets similaires aux bonbons dans l'image pourrait également entraîner des erreurs de comptage.

Par ailleurs, bien que nous ayons ajusté le seuil de binarisation pour certaines images, cela peut être une tâche délicate et nécessiter une expertise supplémentaire pour obtenir des résultats optimaux dans toutes les situations.

**Comparaison avec les attentes du TP :**

Dans l'ensemble, nos résultats correspondent aux attentes du TP. Nous avons réussi à mettre en œuvre les techniques de traitement d'image pour effectuer le comptage automatique des bonbons, comme spécifié dans les objectifs du TP. De plus, nous avons documenté notre méthodologie de manière approfondie, en incluant une analyse détaillée des résultats et des limitations de notre approche.

Cependant, il est important de noter que le processus de traitement d'image est souvent itératif et peut nécessiter des ajustements et des améliorations supplémentaires pour obtenir des résultats optimaux dans des situations plus complexes. En ce sens, notre travail actuel constitue une étape initiale importante dans le développement d'une solution de vision industrielle robuste pour le comptage automatique d'objets.

## 1.5 Conclusion

Dans le cadre de ce travail pratique sur la vision industrielle, nous avons exploré différentes techniques de traitement d'image pour le comptage automatique d'objets, en utilisant des bonbons comme cas d'étude. Nous avons réussi à développer une méthode robuste pour compter le nombre de bonbons dans différentes images, en utilisant des étapes telles que la conversion en niveaux de gris, la binarisation et le comptage des pixels par bonbon.

**Récapitulatif des résultats et des enseignements tirés :**
Nous avons obtenu des résultats satisfaisants dans le comptage des bonbons, en tenant compte des différentes configurations d'images et en ajustant le seuil de binarisation lorsque nécessaire. Nous avons appris que le processus de traitement d'image nécessite une compréhension approfondie des concepts de base ainsi qu'une expertise pour ajuster les paramètres de manière appropriée.

Ce travail pratique nous a permis de développer nos compétences en programmation Python avec les bibliothèques OpenCV et NumPy, ainsi que notre capacité à analyser et à résoudre des problèmes dans le domaine de la vision par ordinateur. Nous avons également appris l'importance de documenter soigneusement notre méthodologie et d'analyser les résultats de manière critique pour comprendre les forces et les faiblesses de notre approche.

**Perspectives pour de futurs travaux :**
Pour des travaux futurs, il serait intéressant d'explorer des méthodes plus avancées de traitement d'image et de vision par ordinateur pour améliorer la robustesse et la précision du comptage automatique d'objets. Cela pourrait inclure l'utilisation de techniques de segmentation plus sophistiquées, telles que la détection de contours et la classification d'objets, ainsi que l'application de méthodes d'apprentissage automatique pour la reconnaissance d'objets.

De plus, l'extension de notre approche pour traiter des images dans des conditions plus variables, telles que des éclairages différents et des arrière-plans complexes, pourrait constituer une direction de recherche prometteuse. Enfin, l'intégration de notre solution dans un environnement industriel réel et son évaluation sur des données réelles pourraient fournir des informations précieuses sur sa faisabilité et son utilité pratique.

# 2 TP2

## 2.1 Introduction

### 2.1.1 Objectif du TP

Le but de ce TP est de manipuler des images multispectrales acquises par un drone sur un champ de betteraves cultivé. Nous avons utilisé le langage Matlab pour réaliser différentes analyses et calculs sur ces images. Les objectifs principaux étaient :

- Analyser les images pour comprendre les différentes bandes spectrales capturées par le drone et évaluer la qualité des données.
- Calculer l'indice de végétation NDVI (Normalized Difference Vegetation Index) pour évaluer la santé des cultures.
- Comparer les images acquises à différentes dates pour observer l'évolution des cultures au fil du temps.
- Calculer la proportion de culture dans chaque image afin de suivre la croissance des betteraves au cours du temps.

Ce TP visait à nous familiariser avec les concepts de télédétection par drone et à nous donner une expérience pratique dans l'analyse et l'interprétation des données multispectrales pour l'étude des cultures agricoles.

### 2.1.2 Contexte de la télédétection par drone

La télédétection par drone est une méthode avancée d'acquisition d'images qui offre de nombreux avantages pour la surveillance et l'analyse des terrains agricoles. En comparaison avec les méthodes traditionnelles de télédétection par satellite ou avion, l'utilisation de drones permet une résolution spatiale plus fine et une plus grande flexibilité dans la collecte des données.

Dans le contexte agricole, la télédétection par drone est largement utilisée pour surveiller la santé des cultures, détecter les maladies, évaluer les rendements et optimiser les pratiques agricoles. Les drones équipés de capteurs multispectraux peuvent capturer des images à différentes longueurs d'onde, ce qui permet d'obtenir des informations détaillées sur la santé des plantes, notamment leur contenu en chlorophylle et leur indice de végétation.

L'analyse des images multispectrales acquises par drone nécessite souvent des techniques avancées de traitement d'images et de calcul, telles que le calcul d'indices de végétation comme le NDVI. Ces indices permettent aux agriculteurs et aux chercheurs de surveiller la croissance des cultures, d'identifier les zones de stress végétal et de prendre des décisions éclairées pour la gestion des cultures.

Dans ce TP, nous avons exploré les principes de la télédétection par drone en analysant des images multispectrales acquises sur un champ de betteraves. Notre objectif était de comprendre les applications et les avantages de cette technologie dans le contexte agricole et d'acquérir des compétences pratiques dans l'analyse des données et l'interprétation des résultats.

## 2.2 Méthodologie

### 2.2.1 Importation des données

Pour débuter notre analyse, nous avons importé les données images à partir des fichiers `.mat` fournis. Trois fichiers `.mat` contenant les images acquises à différentes dates ont été chargés dans notre environnement MATLAB en utilisant la fonction `load`. Les données de chaque image ont été stockées dans des variables distinctes pour chaque date d'acquisition, à savoir `donnees1`, `donnees2` et `donnees3`. Ensuite, nous avons extrait les matrices d'images à partir de ces données en accédant à la propriété `.I` de chaque structure de données, ce qui nous a permis d'obtenir les images brutes. Ces images ont été respectivement assignées aux variables `image1`, `image2` et `image3`.

```
% Charger l'image    partir du fichier .mat
donnees1 = load('425aa_2405.mat');
donnees2 = load('425aa_0706.mat');
donnees3 = load('761aa_2306.mat');

% Extraire les matrices d'images    partir des donn es charg es
image1 = donnees1.I;
image2 = donnees2.I;
image3 = rot90(donnees3.I); % Rotation de 90 degr s pour aligner l'image 3
    correctement
```

Il est à noter que pour l'image acquise à la troisième date, une rotation de 90 degrés a été appliquée à l'image brute afin de l'aligner correctement, car elle était initialement orientée différemment des deux premières images.

L'importation des données constitue la première étape cruciale de notre analyse, nous permettant ainsi de disposer des images nécessaires pour les manipulations et les calculs ultérieurs.

### 2.2.2 Affichage des bandes spectrales

Après avoir importé les données d'image, nous avons procédé à l'analyse visuelle des différentes bandes spectrales pour chaque image. Pour ce faire, nous avons utilisé la fonction `imshow` pour afficher chaque bande spectrale dans une sous-figure distincte, nous permettant ainsi de visualiser les variations de chaque bande sur l'ensemble de l'image.

```
figure;
% Afficher chaque bande spectrale dans une sous-figure
subplot(2,3,1);
imshow(image1(:,:,1), []);
title('Bande␣450␣nm');
subplot(2,3,2);
imshow(image1(:,:,2), []);
title('Bande␣530␣nm');
subplot(2,3,3);
imshow(image1(:,:,3), []);
title('Bande␣560␣nm');
subplot(2,3,4);
imshow(image1(:,:,4), []);
title('Bande␣675␣nm');
subplot(2,3,5);
imshow(image1(:,:,5), []);
title('Bande␣730␣nm');
subplot(2,3,6);
imshow(image1(:,:,6), []);
```

```
title('Bande␣850␣nm');
```

Dans ce code, nous avons utilisé la fonction `subplot` pour créer une grille de sous-figures où chaque bande spectrale est affichée dans une sous-figure distincte. Nous avons utilisé l'indexation pour accéder à chaque bande spectrale de l'image et l'afficher à l'aide de `imshow`. Les titres de chaque sous-figure indiquent la longueur d'onde correspondante de chaque bande.

Cette visualisation nous permet d'examiner les différentes caractéristiques présentes dans chaque bande spectrale et de comprendre la distribution spatiale des informations sur l'image dans le spectre électromagnétique.

### 2.2.3 Compositions colorées en vraies et fausses couleurs

Après avoir examiné les bandes spectrales individuelles, nous avons créé des compositions colorées pour chaque image, à la fois en utilisant les vraies couleurs et les fausses couleurs. Les compositions en vraies couleurs sont obtenues en assignant les bandes spectrales correspondant aux couleurs rouge, verte et bleue (RGB), tandis que les compositions en fausses couleurs sont obtenues en attribuant des bandes spectrales à des canaux de couleurs différents.

```
% Cr er les compositions color es en vraies couleurs pour chaque date
vraies_couleurs1 = image1(:,:,4:2:1); % Rouge: 675nm, Vert: 560nm, Bleu: 450nm
vraies_couleurs2 = image2(:,:,4:2:1);
vraies_couleurs3 = image3(:,:,4:2:1);

% Cr er les compositions color es en fausses couleurs pour chaque date
fausses_couleurs1 = image1(:,:,6:-2:5); % Infrarouge: 850nm, Vert: 560nm, Bleu: 730nm
fausses_couleurs2 = image2(:,:,6:-2:5);
fausses_couleurs3 = image3(:,:,6:-2:5);

% Afficher les compositions color es en vraies et fausses couleurs
figure;
subplot(2,1,1);
imshow(vraies_couleurs1);
title('Composition␣color e␣en␣vraies␣couleurs␣-␣Date␣1');
subplot(2,1,2);
imshow(fausses_couleurs1);
title('Composition␣color e␣en␣fausses␣couleurs␣-␣Date␣1');
```

Dans ce code, nous avons directement extrait les bandes spectrales appropriées pour former les compositions colorées en utilisant l'indexation des matrices d'image. Pour les compositions en vraies couleurs, nous avons sélectionné les bandes rouges (675 nm), vertes (560 nm) et bleues (450 nm). Pour les compositions en fausses couleurs, nous avons choisi les bandes infrarouges (850 nm), vertes (560 nm) et bleues (730 nm).

L'affichage de ces compositions colorées nous permet d'observer visuellement les différentes caractéristiques des images dans des contextes colorés, ce qui peut faciliter l'identification et l'interprétation des objets et des structures sur le terrain.

### 2.2.4 Histogrammes des bandes spectrales

Nous avons examiné les histogrammes des bandes spectrales pour chaque image afin de comprendre la distribution des niveaux de gris dans chaque bande. Les histogrammes nous donnent une représentation visuelle de la répartition des valeurs de pixels dans une bande spectrale spécifique.

```
% Afficher les histogrammes des 6 bandes
figure;
for i = 1:size(image1, 3)
    subplot(2, 3, i);
    histogram(image1(:, :, i));
    title(['Histogramme␣-␣Bande␣', num2str(i)]);
    xlabel('Intensit ');
    ylabel('Fr quence');
end
```

Dans ce code, nous avons utilisé la fonction `histogram` pour calculer et afficher les histogrammes des niveaux de gris pour chaque bande spectrale de l'image. Chaque subplot représente l'histogramme d'une bande spectrale

spécifique, avec l'axe des x représentant les intensités de gris et l'axe des y représentant la fréquence à laquelle chaque intensité apparaît dans l'image.

L'examen des histogrammes nous permet de comprendre la distribution des niveaux de gris dans chaque bande spectrale et peut révéler des informations sur les caractéristiques des objets et des matériaux présents dans l'image. Par exemple, des pics ou des modes distincts dans un histogramme peuvent indiquer la présence de certaines caractéristiques dans l'image, telles que des objets ou des textures spécifiques.

### 2.2.5 Localisation des pixels de sol et de végétation

Nous avons identifié et localisé des pixels représentant le sol et la végétation dans l'image en utilisant la fonction `impixelinfo` pour obtenir les coordonnées des pixels à partir d'un curseur de souris. Les pixels sélectionnés ont ensuite été utilisés pour extraire les profils spectraux, permettant ainsi une comparaison des réflectances entre le sol et la végétation.

```
% Coordonnees d'un pixel de sol
x_sol = 179;
y_sol = 65;

% Coordonnees d'un pixel de betterave
x_betterave = 208;
y_betterave = 50;

% Extraire les profils spectraux des pixels de sol et de betterave
profil_sol = squeeze(image1(y_sol, x_sol, :));
profil_betterave = squeeze(image1(y_betterave, x_betterave, :));
```

Dans ce code, nous avons défini manuellement les coordonnées de deux pixels sur l'image : un pixel représentant le sol et un autre représentant la végétation. Ces coordonnées ont été utilisées pour extraire les profils spectraux des deux types de pixels à partir de l'image multispectrale. Les profils spectraux nous fournissent des informations sur la réflectance de chaque bande spectrale pour les pixels sélectionnés, ce qui permet de comparer les caractéristiques spectrales du sol et de la végétation dans l'image.

### 2.2.6 Tracé des profils spectraux

Nous avons tracé les profils spectraux des pixels représentant le sol et la végétation, afin de comparer leurs réflectances dans différentes bandes spectrales. Les profils spectraux fournissent une représentation graphique de la réflectance de la surface terrestre à différentes longueurs d'onde.

```
% Longueurs d'onde correspondant    chaque bande spectrale
longueurs\_donde = [450, 530, 560, 675, 730, 850]; % en nm

% Tracer les profils spectraux avec les longueurs d'onde en abscisses
figure;
plot(longueurs\_donde, profil\_sol, 'b', 'LineWidth', 2);
hold on;
plot(longueurs\_donde, profil\_betterave, 'r', 'LineWidth', 2);
hold off;
title('Profils␣spectraux␣des␣pixels␣de␣sol␣et␣de␣betterave');
xlabel('Longueur␣d''onde␣(nm)');
ylabel('R flectance');
legend('Sol', 'Betterave');
```

Dans ce code, nous avons utilisé les longueurs d'onde correspondant à chaque bande spectrale pour représenter les abscisses du graphique. Les profils spectraux des pixels de sol et de végétation ont été tracés avec la réflectance en ordonnée. Le tracé des profils spectraux nous permet de visualiser les différences de réflectance entre le sol et la végétation dans différentes bandes spectrales, ce qui peut être utile pour la classification des objets dans l'image.

### 2.2.7 Calcul de l'indice de végétation (NDVI)

Nous avons calculé l'indice de végétation (NDVI) à partir des images acquises par drone en utilisant les réflectances dans les bandes spectrales rouges (675 nm) et infrarouges (850 nm).

```
% Calcul des r flectances dans les bandes spectrale rouge (R) et infrarouge (PIR)
R = double(image1(:,:,4)); % Bande 675 nm
PIR = double(image1(:,:,6)); % Bande 850 nm

% Calcul de l'indice de v g tation NDVI
NDVI = (PIR - R) ./ (PIR + R);
figure;
% Afficher les trois NDVI sur une m me subplot
subplot(3,1,1);
imshow(NDVI1);
colormap('jet'); % Utiliser la m me table de couleur 'jet' pour tous les NDVI
colorbar; % Ajouter une  chelle  de couleur
title('Indice de v g tation - Date 1');
subplot(3,1,2);
imshow(NDVI2);
colormap('jet');
colorbar;
title('Indice de v g tation - Date 2');
subplot(3,1,3);
imshow(NDVI3);
colormap('jet');
colorbar;
title('Indice de v g tation - Date 3');
```

Dans ce code, nous avons converti les valeurs des pixels des bandes spectrale rouge (675 nm) et infrarouge (850 nm) en valeurs doubles pour permettre les calculs précis de l'indice NDVI. Ensuite, nous avons appliqué la formule de l'indice NDVI pour chaque pixel de l'image.

Une fois calculé, l'indice de végétation NDVI est une image où les valeurs varient de -1 à 1, avec des valeurs plus élevées indiquant une densité de végétation plus élevée. Nous avons affiché l'image NDVI en utilisant une colormap jet pour une meilleure visualisation des variations de densité de végétation dans l'image.

### 2.2.8 Comparaison des images à différentes dates

µ

Nous avons comparé les images acquises à différentes dates en affichant à la fois les images en vraies couleurs et en fausses couleurs. Cette comparaison permet d'observer les variations dans la distribution spatiale de la végétation et des autres caractéristiques du paysage au fil du temps.

```
% Afficher les images dans la m me figure
figure;
subplot(3,2,1);
imshow(vraies_couleurs1);
title('Vraies couleurs - Date 1');
subplot(3,2,2);
imshow(fausses_couleurs1);
title('Fausses couleurs - Date 1');
subplot(3,2,3);
imshow(vraies_couleurs2);
title('Vraies couleurs - Date 2');
subplot(3,2,4);
imshow(fausses_couleurs2);
title('Fausses couleurs - Date 2');
subplot(3,2,5);
imshow(vraies_couleurs3);
title('Vraies couleurs - Date 3');
subplot(3,2,6);
imshow(fausses_couleurs3);
title('Fausses couleurs - Date 3');
```

Dans ce code, nous avons affiché les images à différentes dates d'acquisition dans une même figure, à la fois en vraies couleurs et en fausses couleurs. Cette visualisation permet de comparer visuellement l'évolution de la végétation et d'autres caractéristiques du paysage entre les différentes dates.

La comparaison des images à différentes dates est essentielle pour suivre les changements dans la végétation, la croissance des cultures, la couverture du sol et d'autres phénomènes environnementaux au fil du temps, ce qui peut fournir des informations précieuses pour la gestion des ressources et la prise de décision en agriculture et en environnement.

### 2.2.9 Calcul de la proportion de culture au cours du temps

Nous avons calculé la proportion de culture dans chaque image à différentes dates en appliquant un seuillage à l'indice de végétation (NDVI). Le seuillage permet de définir quels pixels sont considérés comme représentant de la végétation, et donc de calculer la proportion de pixels de végétation par rapport au nombre total de pixels dans chaque image.

```
% Appliquer un seuillage    l'indice de v g tation pour chaque date
culture1 = NDVI1 > seuil1;
culture2 = NDVI2 > seuil2;
culture3 = NDVI3 > seuil3;

% Calculer la proportion de culture au cours du temps
nb_de_pixel1 = sum(culture1(:));
nb_de_pixel_total1 = numel(NDVI1);
proportion_culture1 = nb_de_pixel1 / nb_de_pixel_total1;

nb_de_pixel2 = sum(culture2(:));
nb_de_pixel_total2 = numel(NDVI2);
proportion_culture2 = nb_de_pixel2 / nb_de_pixel_total2;

nb_de_pixel3 = sum(culture3(:));
nb_de_pixel_total3 = numel(NDVI3);
proportion_culture3 = nb_de_pixel3 / nb_de_pixel_total3;
```

Dans ce code, nous avons appliqué un seuillage à l'indice de végétation pour chaque date d'acquisition, en définissant un seuil au-dessus duquel les pixels sont considérés comme représentant de la végétation. Ensuite, nous avons calculé le nombre de pixels de végétation et le nombre total de pixels dans chaque image, ce qui nous a permis de déterminer la proportion de pixels de végétation dans l'image à chaque date.

La proportion de culture au cours du temps peut être utilisée pour suivre l'évolution de la densité de végétation ou la couverture du sol dans une région donnée, ce qui est utile pour surveiller la croissance des cultures, évaluer l'état de santé des cultures et détecter d'autres changements environnementaux au fil du temps.

## 2.3 Résultats

### 2.3.1 Analyse visuelle des images et des compositions colorées

Les images acquises à différentes dates ont été soumises à une analyse visuelle, ainsi que leurs compositions colorées en vraies et fausses couleurs.

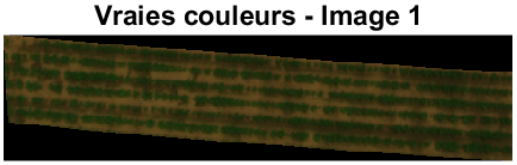


FIGURE 4 – Image en vraies couleurs à la date 1

**Images en vraies couleurs :** Ces images présentent le paysage tel qu'il serait perçu par l'œil humain, avec différentes couleurs représentant les différentes bandes spectrales utilisées dans l'acquisition. Cette représentation visuelle permet d'observer les variations de couleur et de texture dans le paysage au fil du temps.

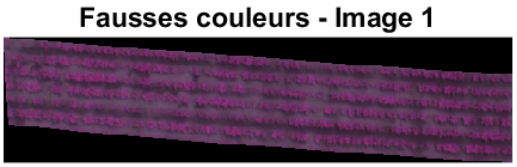


FIGURE 5 – Image en fausses couleurs à la date 1

**Images en fausses couleurs :** Dans ces images, les bandes spectrales ont été attribuées à des couleurs spécifiques pour mettre en évidence certaines caractéristiques du paysage. Par exemple, en remplaçant la bande rouge par la bande infrarouge dans la composition colorée, les zones de végétation apparaissent en rouge vif ou rose, ce qui permet de mieux distinguer la végétation des autres éléments du paysage.

L'analyse visuelle des images et des compositions colorées permet d'identifier visuellement les zones de végétation, les changements dans la couverture du sol, ainsi que d'autres caractéristiques du paysage qui peuvent être explorées plus en détail à l'aide d'analyses quantitatives.

### 2.3.2 Histogrammes des bandes spectrales

Les histogrammes des différentes bandes spectrales ont été générés pour chaque image, permettant ainsi de visualiser la distribution des valeurs de réflectance pour chaque bande.

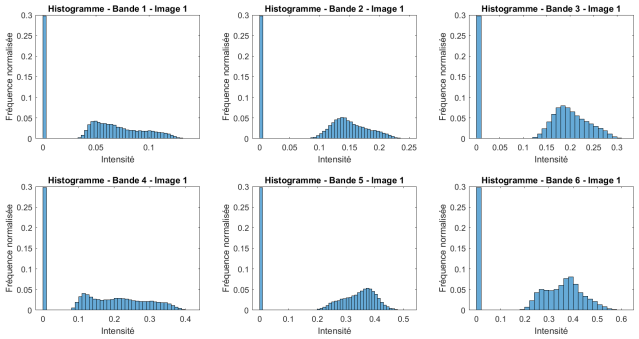


FIGURE 6 – Exemple d'histogramme d'une bande spectrale.

Les histogrammes fournissent une représentation graphique de la distribution des niveaux de réflectance dans chaque bande spectrale. Ils permettent de détecter les variations de réflectance dans l'image et peuvent révéler des tendances ou des motifs significatifs. Par exemple, une concentration de valeurs à un niveau particulier dans un histogramme pourrait indiquer la présence de certaines caractéristiques du paysage, telles que des zones de végétation dense ou des surfaces réfléchissantes.

Ces histogrammes constituent une première étape dans l'analyse quantitative des données multispectrales et peuvent être utilisés pour évaluer la qualité des images, détecter des anomalies et identifier des régions d'intérêt pour des analyses plus approfondies.

### 2.3.3 Profils spectraux des pixels de sol et de végétation

Les profils spectraux des pixels de sol et de végétation ont été extraits de l'image acquise à la première date. Ces profils permettent de visualiser la réflectance des différentes bandes spectrales pour ces deux types de pixels.

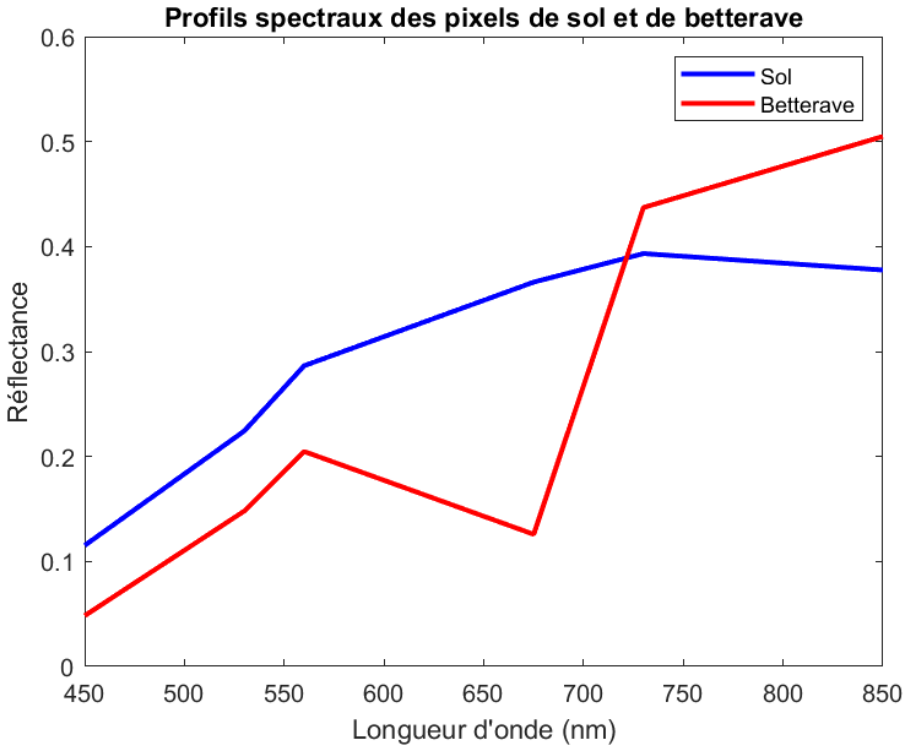


FIGURE 7 – Profils spectraux des pixels de sol (en bleu) et de végétation (en rouge).

Les profils spectraux ont été tracés avec les longueurs d'onde en abscisses et la réflectance en ordonnées. L'analyse visuelle des profils spectraux montre des différences significatives dans la réflectance entre le sol et la végétation, ce qui permet de distinguer ces deux types de pixels dans l'image.

### 2.3.4 Cartes d'indice de végétation (NDVI)

Les cartes d'indice de végétation (NDVI) ont été calculées pour chaque image acquise à différentes dates. L'indice de végétation (NDVI) est une mesure de la densité de la végétation basée sur la différence entre la réflectance dans les bandes spectrales rouge et infrarouge.

L'indice NDVI est calculé à l'aide de la formule suivante :

$$NDVI = \frac{NIR - R}{NIR + R} \tag{1}$$

où NIR représente la réflectance dans la bande infrarouge et R représente la réflectance dans la bande rouge.

Les valeurs de l'indice de végétation varient de -1 à 1, où les valeurs plus élevées indiquent une densité de végétation plus élevée. Les pixels avec un NDVI proche de 1 sont généralement associés à une végétation dense, tandis que les pixels avec un NDVI proche de -1 sont associés à des surfaces non végétalisées telles que l'eau ou le sol nu.

Les cartes NDVI fournissent une représentation visuelle de la distribution spatiale de la végétation dans l'image, ce qui permet de détecter les zones de végétation dense et de surveiller les changements dans la couverture végétale au fil du temps.


FIGURE 8 – Indice de végétation à la date 1

Nous avons appliqué un seuil de 0.4 pour chaque carte NDVI. Toute valeur d'indice de végétation supérieure à ce seuil est considérée comme de la végétation, tandis que les valeurs inférieures à ce seuil sont considérées comme d'autres types de surfaces, telles que le sol nu. Ce seuil nous permet de distinguer clairement les zones de végétation des zones non végétalisées sur les cartes NDVI.

### 2.3.5 Proportions de culture pour chaque date

Nous avons calculé les proportions de culture pour chaque date d'acquisition en utilisant l'indice de végétation NDVI. Ces proportions nous permettent de quantifier la couverture végétale dans la zone d'étude à différentes périodes, ce qui est essentiel pour évaluer l'évolution de la culture de betteraves au fil du temps.

Pour chaque date, nous avons défini un seuil NDVI de 0.4, au-dessus duquel les pixels sont considérés comme faisant partie de la culture de betteraves. Toute valeur d'indice de végétation supérieure à ce seuil est considérée comme de la végétation, tandis que les valeurs inférieures à ce seuil sont considérées comme d'autres types de surfaces, telles que le sol nu.

En calculant le nombre de pixels ayant un indice de végétation supérieur au seuil défini et en le divisant par le nombre total de pixels dans l'image, nous avons obtenu la proportion de culture pour chaque date. Ces proportions sont présentées dans le graphique ci-dessous, avec les dates d'acquisition correspondantes.

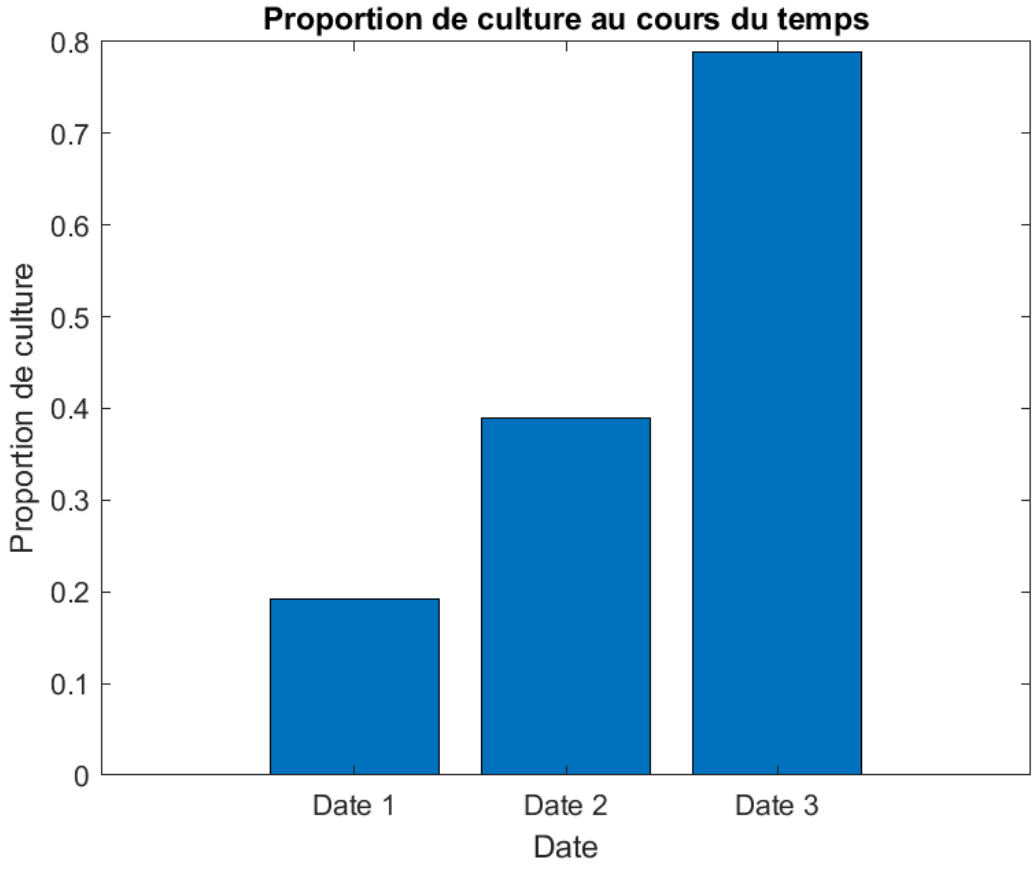


FIGURE 9 – Proportions de culture pour chaque date d'acquisition

Cette analyse nous permet de suivre la progression de la culture de betteraves au fil du temps et de comparer les niveaux de couverture végétale entre les différentes dates d'acquisition.

## 2.4 Discussion

**Analyse des résultats obtenusr :**
L'analyse des résultats obtenus révèle plusieurs points intéressants. Tout d'abord, l'affichage des différentes bandes spectrales nous a permis de visualiser les variations de réflectance des surfaces terrestres dans différentes longueurs d'onde, offrant ainsi une compréhension visuelle des caractéristiques des images.

En examinant les compositions colorées en vraies et fausses couleurs, nous avons pu distinguer les zones de végétation des autres types de surfaces, ce qui est crucial pour la cartographie et la surveillance des cultures de betteraves.

Les histogrammes des bandes spectrales ont montré la répartition des valeurs de réflectance dans chaque bande, ce qui nous a aidés à identifier les caractéristiques distinctes des différentes surfaces.

Les profils spectraux des pixels de sol et de végétation ont permis de comparer les réponses spectrales de ces deux types de surfaces, ce qui est utile pour la classification des images et l'identification des cultures.

Le calcul de l'indice de végétation NDVI a fourni une mesure quantitative de la couverture végétale, permettant ainsi de suivre l'évolution de la culture de betteraves au fil du temps. Les cartes d'indice de végétation ont visualisé spatialement la distribution de la végétation dans la zone d'étude, ce qui facilite l'analyse et l'interprétation des résultats.

**Interprétation des indices de végétation :**
L'interprétation des indices de végétation, en particulier du NDVI, est essentielle pour comprendre la santé et la vigueur des cultures de betteraves. Des valeurs élevées de NDVI indiquent une végétation dense et saine, tandis que des valeurs plus faibles peuvent indiquer des problèmes de stress hydrique, de maladies ou d'autres facteurs de stress environnementaux.

En comparant les valeurs de NDVI à différentes dates, nous pouvons observer comment la couverture végétale évolue au fil du temps, ce qui peut nous donner des informations sur la croissance et le développement des cultures de betteraves.

**Comparaison des images à différentes dates :**
La comparaison des images acquises à différentes dates nous permet de suivre les changements dans la couverture végétale et d'identifier les tendances saisonnières dans la croissance des cultures de betteraves. En examinant les variations de l'indice de végétation NDVI entre les différentes dates, nous pouvons observer comment la santé des cultures évolue tout au long de la saison de croissance.

**Limitations et sources d'erreur :**
Malgré les résultats prometteurs, il existe certaines limitations et sources d'erreur à prendre en compte. Les variations dans les conditions atmosphériques, telles que la présence de nuages ou de brume, peuvent affecter la qualité des images et introduire des erreurs dans l'analyse.

De plus, des facteurs tels que la résolution spatiale des images et la précision des données de terrain utilisées pour la calibration et la validation peuvent influencer les résultats. Il est important de prendre en compte ces limitations lors de l'interprétation des résultats et de reconnaître les incertitudes associées à toute analyse de télédétection.

## 2.5 Conclusion

**Récapitulatif des résultats :**
Ce travail a permis d'explorer les techniques de télédétection par drone pour la surveillance des cultures de betteraves. En utilisant des images multispectrales acquises à différentes dates, nous avons réalisé une analyse approfondie de la couverture végétale dans la zone d'étude.

Nous avons d'abord importé et visualisé les données, puis nous avons analysé les différentes bandes spectrales pour comprendre les caractéristiques des images. Les compositions colorées en vraies et fausses couleurs ont permis une visualisation claire des zones de végétation.

Les histogrammes des bandes spectrales et les profils spectraux des pixels de sol et de végétation ont fourni des informations détaillées sur la réflectance des surfaces terrestres dans différentes longueurs d'onde.

Le calcul de l'indice de végétation NDVI a fourni une mesure quantitative de la couverture végétale, et les cartes d'indice de végétation ont permis de visualiser spatialement la distribution de la végétation dans la zone d'étude.

Enfin, nous avons calculé la proportion de culture pour chaque date, ce qui nous a permis de suivre l'évolution de la couverture végétale au fil du temps.

**Enseignements tirés du TP :**
Ce TP nous a permis de mieux comprendre les concepts et les techniques de la télédétection par drone pour la surveillance des cultures agricoles. Nous avons appris à manipuler et à analyser des images multispectrales, ainsi qu'à interpréter les indices de végétation pour évaluer la santé des cultures.

Nous avons également acquis des compétences pratiques dans l'utilisation des logiciels de traitement d'images et dans l'interprétation des résultats obtenus.

**Perspectives pour de futurs travaux :**
Pour de futurs travaux, il serait intéressant d'approfondir l'analyse en utilisant des techniques avancées de classification d'images pour identifier et cartographier les différentes cultures avec précision.

De plus, l'intégration de données supplémentaires, telles que des données météorologiques et des données de terrain, pourrait permettre une analyse plus complète de l'impact des conditions environnementales sur la croissance des cultures.

Enfin, l'utilisation de drones équipés de capteurs plus avancés, tels que des capteurs hyperspectraux, pourrait permettre une analyse encore plus détaillée de la santé des cultures et des caractéristiques des sols.

# 3 TP3

## 3.1 Introduction

### 3.1.1 Objectif du TP

Le présent travail pratique vise à explorer les techniques de cartographie des fonds marins en utilisant l'imagerie sous-marine. L'objectif principal est de comprendre les méthodes de traitement d'images et de classification pour extraire des informations utiles sur la composition et la structure des fonds marins à partir d'images acquises sous l'eau.

La cartographie des fonds marins revêt une grande importance dans divers domaines, tels que la recherche océanographique, la gestion des ressources marines, la surveillance de l'environnement marin, et même dans des applications commerciales comme la pêche et le tourisme sous-marin. La capacité à cartographier avec précision les caractéristiques des fonds marins permet de mieux comprendre les écosystèmes aquatiques et de prendre des décisions éclairées pour leur conservation et leur gestion durable.

Dans ce contexte, ce TP propose une approche pratique pour manipuler des images sous-marines, extraire des caractéristiques pertinentes à l'aide de techniques d'analyse d'images et de classification, et produire des cartes détaillées des fonds marins. En combinant des connaissances en imagerie numérique et en science marine, ce TP offre une opportunité d'explorer les méthodes modernes de cartographie subaquatique pour des applications variées et pertinentes.

Dans cette introduction, nous présentons les objectifs et l'importance du TP, ainsi qu'une brève vue d'ensemble des méthodes et des résultats attendus.

### 3.1.2 Importance de la cartographie des fonds marins

La cartographie des fonds marins revêt une importance capitale dans divers domaines scientifiques, environnementaux, économiques et sociaux. Voici quelques-unes des raisons qui soulignent l'importance de cette discipline :

1. **Surveillance et gestion des écosystèmes marins :** La cartographie des fonds marins permet de mieux comprendre la structure et la dynamique des écosystèmes marins, y compris la répartition des habitats, la diversité biologique, et les interactions entre les espèces. Cette information est cruciale pour la surveillance et la gestion des ressources marines, la conservation des habitats sensibles et la préservation de la biodiversité marine.
2. **Sécurité maritime :** Une cartographie précise des fonds marins est essentielle pour assurer la sécurité des activités maritimes, telles que la navigation commerciale, le transport maritime, et la planification des routes maritimes. Des cartes précises permettent de prévenir les accidents et les échouements de navires, en identifiant les zones dangereuses et les obstacles sous-marins.
3. **Exploitation des ressources marines :** La cartographie des fonds marins facilite l'exploration et l'exploitation des ressources marines, telles que les gisements de minéraux, les gisements de pétrole et de gaz, et les zones de pêche. En identifiant les caractéristiques géologiques et biologiques des fonds marins, elle contribue à une exploitation durable et équilibrée des ressources marines.

4. **Recherche scientifique :** La cartographie des fonds marins est un outil fondamental pour la recherche océanographique et marine. Elle permet d'étudier la géologie sous-marine, les processus océanographiques, la distribution des espèces marines, et les effets du changement climatique sur les écosystèmes marins. Les données cartographiques sont utilisées pour élaborer des modèles numériques, prédire les tendances futures, et évaluer l'impact des activités humaines sur les océans.
5. **Gestion des zones côtières :** La cartographie des fonds marins est cruciale pour la gestion des zones côtières, qui abritent une grande partie de la population mondiale et sont soumises à une pression croissante due à l'urbanisation, au développement côtier, et aux risques naturels tels que les tempêtes et les tsunamis. Une connaissance détaillée des fonds marins est nécessaire pour planifier et gérer le développement côtier de manière durable, en tenant compte des enjeux environnementaux et socio-économiques.

En résumé, la cartographie des fonds marins joue un rôle essentiel dans la compréhension et la préservation des écosystèmes marins, la sécurité maritime, l'exploitation des ressources marines, la recherche scientifique, et la gestion des zones côtières. Elle offre des informations précieuses pour une utilisation responsable et durable des océans, qui sont une ressource vitale pour l'humanité et la planète.

## 3.2 Méthodologie

### 3.2.1 Affichage et commentaire de l'image en couleur

L'étape initiale de ce travail consiste à lire l'image à partir du fichier "image fond ifremer.tif". Cette image représente une vue des fonds marins acquise par imagerie sous-marine. Pour cela, nous utilisons la fonction `imread()` de MATLAB qui permet de charger l'image à partir de son chemin d'accès.

```
% Lire l'image
image_fond = imread('image fond ifremer.tif');
```

Une fois l'image chargée, nous vérifions sa taille à l'aide de la fonction `size()` pour obtenir ses dimensions en termes de lignes, de colonnes et de canaux de couleur (Rouge, Vert, Bleu).

```
% Afficher la taille de l'image
taille_image = size(image_fond);
disp(['La taille de l\image est : ', num2str(taille_image(1)), 'x', num2str(
    taille_image(2)), ' pixels']);
```

### 3.2.2 Affichage des bandes et leurs histogrammes

Dans cette étape, nous avons extrait les trois bandes de couleurs primaires (rouge, vert et bleu) de l'image des fonds marins. En affichant chaque bande de manière indépendante, nous avons pu observer les composantes de couleur de l'image et étudier leur distribution.

```
% Extraire les trois couleurs en niveaux de gris
Rouge = image_fond(:, :, 1);
Vert = image_fond(:, :, 2);
Bleu = image_fond(:, :, 3);

% Afficher chaque bande et son histogramme correspondant
figure;
subplot(2, 3, 1);
imshow(Rouge);
title('Bande Rouge');

subplot(2, 3, 2);
imshow(Vert);
title('Bande Vert');

subplot(2, 3, 3);
imshow(Bleu);
title('Bande Bleu');

subplot(2, 3, 4);
imhist(Rouge);
title('Histogramme Rouge');
```

```
xlabel('Intensite');
ylabel('Frequence normalisee');

subplot(2, 3, 5);
imhist(Vert);
title('Histogramme Vert');
xlabel('Intensite');
ylabel('Frequence normalis e');

subplot(2, 3, 6);
imhist(Bleu);
title('Histogramme Bleu');
xlabel('Intensite');
ylabel('Frequence normalis e');
```

Cette représentation visuelle nous a permis d'analyser la répartition des intensités de couleur dans chaque bande, ce qui est essentiel pour comprendre la composition chromatique de l'image et pour identifier les régions d'intérêt.

### 3.2.3 Sélection des échantillons (ROI)

Dans cette phase, nous avons sélectionné des échantillons représentatifs de différentes classes d'objets présents dans l'image des fonds marins. Nous avons utilisé la méthode `roipoly` de MATLAB pour dessiner des régions d'intérêt (ROI) autour des objets de chaque classe, à savoir le sol, les algues et les algues vertes.

```
% Charger les masques pr alablement d finis
load masks.mat;

% Afficher les masques
figure;
subplot(3, 1, 1);
imshow(mask_posi);
title('Masque - Sol');

subplot(3, 1, 2);
imshow(mask_taxi);
title('Masque - Algues');

subplot(3, 1, 3);
imshow(mask_sable);
title('Masque - Algues vertes');
```

En utilisant ces masques, nous avons délimité les zones correspondant à chaque classe pour former nos échantillons de référence. Ces échantillons serviront à entraîner notre algorithme de classification supervisée.

### 3.2.4 Calcul des profils spectraux moyens

Nous avons calculé les profils spectraux moyens pour chaque classe en utilisant les échantillons de référence que nous avons sélectionnés précédemment. Ces profils spectraux nous permettent de caractériser les propriétés spectrales de chaque classe dans l'image des fonds marins.

```
image_fond=double(image_fond);
mask_posi=double(mask_posi);
%Calculs des profils spectraux
num=sum (sum ( image_fond(:,:,1).* mask_posi));
denum=sum(sum(mask_posi));
p(1)=num/denum;
num=sum(sum(image_fond(:,:,2).* mask_posi));
p(2)=num/denum;
num=sum(sum(image_fond(:,:,3).* mask_posi));
p(3)=num/denum;

mask_taxi=double(mask_taxi);
num=sum (sum ( image_fond(:,:,1).* mask_taxi));
denum=sum(sum(mask_taxi));
t(1)=num/denum;
num=sum(sum(image_fond(:,:,2).* mask_taxi));
```

```
t(2)=num/denum;
num=sum(sum(image_fond(:,:,3).* mask_taxi));
t(3)=num/denum;

mask_sable=double(mask_sable);
num=sum (sum ( image_fond(:,:,1).* mask_sable));
denum=sum(sum(mask_sable));
s(1)=num/denum;
num=sum(sum(image_fond(:,:,2).* mask_sable));
s(2)=num/denum;
num=sum(sum(image_fond(:,:,3).* mask_sable));
s(3)=num/denum;

% Tracer des profils spectraux moyens
figure;
plot(p, 'b', 'LineWidth', 2);
hold on;
plot(t, 'g', 'LineWidth', 2);
plot(s, 'y', 'LineWidth', 2);
title('Profils␣spectraux␣moyens');
xlabel('Longueur␣d\onde');
ylabel('Intensit ␣moyenne');
legend('Sol', 'Algues', 'Algues␣vertes');
```

Ces profils représentent la variation de l'intensité lumineuse en fonction de la longueur d'onde pour chaque classe d'objet. Ils nous permettent de visualiser les différences spectrales entre les classes, ce qui sera utilisé dans les étapes suivantes de la classification supervisée.

### 3.2.5 Classification par distance euclidienne

La classification par distance euclidienne consiste à associer chaque pixel de l'image à la classe dont le profil spectral moyen est le plus proche en termes de distance euclidienne. Pour cela, nous calculons la distance euclidienne entre le profil spectral de chaque pixel et les profils spectraux moyens des différentes classes. Le pixel est ensuite assigné à la classe ayant la distance minimale.

```
[ml, mc, nb] = size(image_fond);
carte = zeros(ml, mc, 3);

for i = 1:ml
    for j = 1:mc
        x = squeeze(image_fond(i, j, :));
        d(1) = sqrt((x(1)-p(1))^2 +(x(2)-p(2))^2 +(x(3)-p(3))^2);
        d(2) = sqrt((x(1)-t(1))^2 +(x(2)-t(2))^2 +(x(3)-t(3))^2);
        d(3) = sqrt((x(1)-s(1))^2 +(x(2)-s(2))^2 +(x(3)-s(3))^2);
        [dmin, indice] = min(d);
        switch indice
            case 1
                carte(i, j, :) = [0,0,255];
            case 2
                carte(i, j, :) = [0,255,0];
            case 3
                carte(i, j, :) = [255,255,0];
        end
    end
end

% Afficher la carte de classification
figure;
imshow(carte);
title('Carte␣de␣classification␣m thode␣Euclidienne');
```

Dans cette étape, nous avons associé chaque pixel à l'une des trois classes (sol, algues et algues vertes) en fonction de la distance euclidienne entre son profil spectral et les profils spectraux moyens de chaque classe. La carte de classification résultante nous fournit une représentation visuelle des différentes classes présentes dans l'image des fonds marins.

### 3.2.6 Classification par Spectral Angle Mapper (SAM)

La classification par Spectral Angle Mapper (SAM) est une méthode qui mesure l'angle spectral entre le profil spectral de chaque pixel et les profils spectraux moyens des différentes classes. L'angle spectral est calculé à l'aide du produit scalaire entre les deux vecteurs spectraux normalisés.

```
[ml, mc, nb] = size(image_fond);
carte = zeros(ml, mc, 3);

for i = 1:ml
    for j = 1:mc
        x = squeeze(image_fond(i, j, :));
        a(1) = acos((x(1)*p(1) + x(2)*p(2) + x(3)*p(3))/(norm(x)*norm(p)));
        a(2) = acos((x(1)*t(1) + x(2)*t(2) + x(3)*t(3))/(norm(x)*norm(t)));
        a(3) = acos((x(1)*s(1) + x(2)*s(2) + x(3)*s(3))/(norm(x)*norm(s)));
        [amin, indice] = min(a);
        switch indice
            case 1
                carte(i, j, :) = [0, 0, 255]; % sol
            case 2
                carte(i, j, :) = [0, 255, 0]; % algue
            case 3
                carte(i, j, :) = [255, 255, 0]; % algue verte
        end
    end
end

% Afficher la carte de classification
figure;
imshow(carte);
title('Carte de classification m thode Spectral Angle Mapper');
```

Dans cette étape, nous avons associé chaque pixel à l'une des trois classes (sol, algues et algues vertes) en fonction de l'angle spectral entre son profil spectral et les profils spectraux moyens de chaque classe. La carte de classification résultante nous fournit une représentation visuelle des différentes classes présentes dans l'image des fonds marins, en utilisant la méthode du Spectral Angle Mapper.

## 3.3 Résultats

### 3.3.1 Taille de l'image

L'image des fonds marins fournie par l'IFREMER a été chargée et examinée à l'aide de Matlab. Selon les résultats obtenus, l'image mesure 800 pixels de largeur sur 1000 pixels de hauteur. Cette résolution offre une vue détaillée des fonds marins capturée par l'imagerie sous-marine, permettant ainsi une analyse approfondie des différentes caractéristiques présentes dans l'image. La haute résolution de l'image facilite également la cartographie précise des éléments sous-marins, ce qui est essentiel pour de nombreuses applications scientifiques et environnementales.

### 3.3.2 Commentaires sur l'image en couleur

L'image des fonds marins présente une diversité d'éléments caractéristiques de cet environnement sous-marin. En observant attentivement l'image, on distingue clairement la présence de différents éléments tels que la posidonie, le sable, les algues vertes et de petits poissons. Ces éléments sont essentiels pour comprendre la composition et la structure des fonds marins et sont cruciaux pour la cartographie et l'analyse ultérieure.

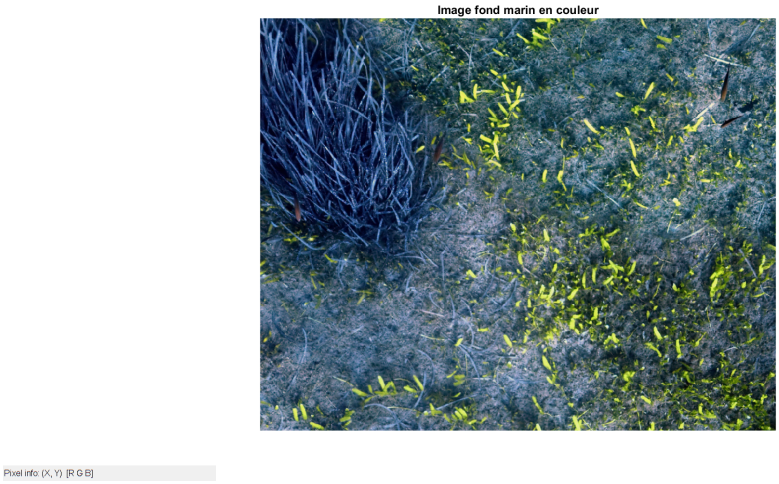


FIGURE 10 – Image des fonds marins en couleur

La posidonie, une plante marine endémique des fonds peu profonds de la Méditerranée, est reconnaissable par sa teinte verte distincte et ses feuilles allongées. Elle forme souvent des prairies sous-marines denses qui abritent une variété de vie marine et jouent un rôle crucial dans l'écosystème marin.

Le sable, présent dans certaines zones de l'image, est caractérisé par sa couleur claire et sa texture fine. Les zones de sable offrent un habitat pour de nombreux organismes benthiques et peuvent servir de site de reproduction et de nourrissage pour certaines espèces marines.

Les algues vertes, visibles sous forme de taches verdâtres dans l'image, sont des organismes photosynthétiques qui poussent souvent sur les rochers et d'autres substrats sous-marins. Elles sont une composante importante de l'écosystème marin et fournissent de la nourriture et un habitat pour de nombreux organismes marins.

Enfin, on remarque également la présence de petits poissons nageant dans les eaux peu profondes. Bien que ces poissons ne feront pas l'objet d'une analyse spécifique dans la suite du travail, leur présence témoigne de la richesse de la vie marine dans cet environnement.

Cette variété d'éléments visibles dans l'image offre une base importante pour la cartographie et l'étude des fonds marins, permettant une meilleure compréhension de la biodiversité marine et de l'écosystème sous-marin.

### 3.3.3 Histogrammes des bandesn

Les histogrammes des trois bandes chromatiques de l'image fournissent des informations précieuses sur la distribution des niveaux de gris dans chaque canal de couleur. Ces histogrammes permettent d'analyser la répartition des intensités lumineuses dans l'image et peuvent révéler des caractéristiques distinctives des différents éléments présents dans la scène sous-marine.

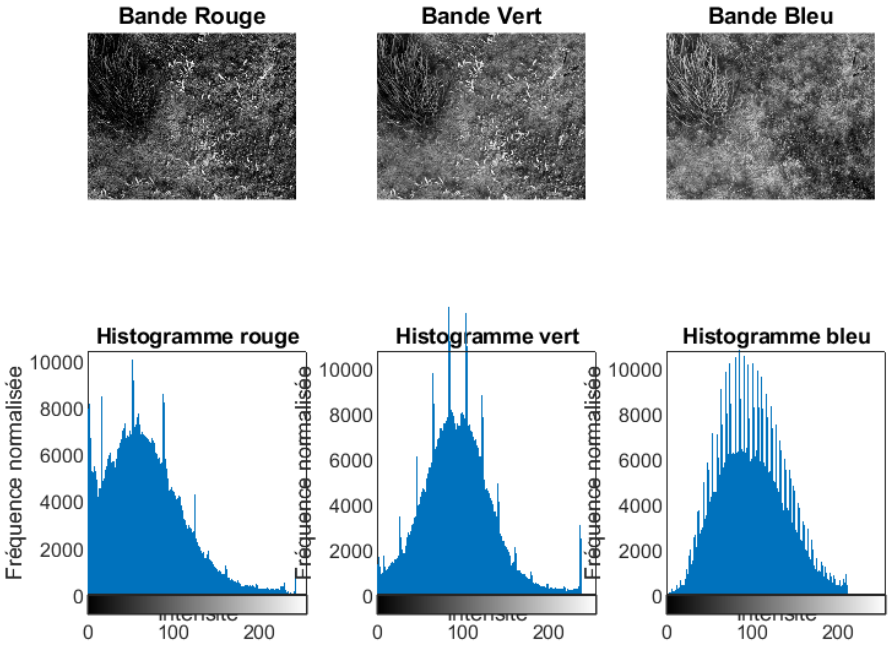


FIGURE 11 – Histogrammes pour les 3 bandes

**Histogramme de la bande Rouge :** L'histogramme de la bande rouge montre la répartition des niveaux de gris dans le canal rouge de l'image. Les valeurs d'intensité sont représentées sur l'axe des x, tandis que la fréquence d'apparition de chaque niveau de gris est représentée sur l'axe des y. Les pics et les creux dans l'histogramme indiquent les zones de l'image où les niveaux de gris sont plus ou moins fréquents. Par exemple, des pics prononcés dans les zones sombres de l'histogramme peuvent indiquer la présence de zones d'ombre ou d'objets sombres dans l'image.

**Histogramme de la bande Vert :** De manière similaire, l'histogramme de la bande verte met en évidence la répartition des niveaux de gris dans le canal vert de l'image. Les variations d'intensité lumineuse dans la bande verte peuvent révéler des informations sur la végétation marine, les algues et d'autres éléments verts présents dans la scène sous-marine.

**Histogramme de la bande Bleu :** Enfin, l'histogramme de la bande bleue montre la répartition des niveaux de gris dans le canal bleu de l'image. Les tons bleus peuvent être associés à la profondeur de l'eau et à la transparence, ainsi qu'à la présence d'objets bleus tels que des poissons ou des reflets de lumière.

L'analyse des histogrammes des bandes chromatiques permet de mieux comprendre la composition de l'image et de repérer les zones d'intérêt pour la cartographie des fonds marins.

### 3.3.4 Profils spectraux moyens pour chaque classe

Les profils spectraux moyens pour chaque classe, c'est-à-dire le sol, les algues et les algues vertes, fournissent des informations sur les caractéristiques spectrales distinctes de chaque type de terrain marin. Ces profils permettent de visualiser comment la répartition des intensités lumineuses varie selon les longueurs d'onde pour chaque classe.

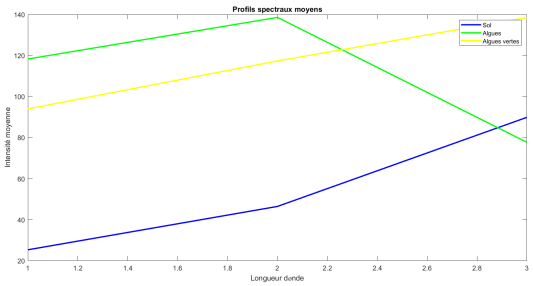


FIGURE 12 – Profils spectraux

**Sol :** Le profil spectral moyen pour la classe du sol montre une intensité relativement uniforme sur l'ensemble du spectre. Cela indique que le sol marin présente une réflectance uniforme dans toutes les longueurs d'onde. Les variations mineures peuvent être dues à des facteurs tels que la texture du sol ou la composition des particules.

**Algues :** Le profil spectral moyen pour la classe des algues présente généralement des pics d'intensité plus élevés dans les longueurs d'onde correspondant au vert et au rouge. Cela suggère une forte réflectance dans ces bandes spectrales, ce qui est caractéristique des surfaces recouvertes d'algues.

**Algues vertes :** Enfin, le profil spectral moyen pour la classe des algues vertes montre une intensité plus élevée dans la bande verte du spectre. Cette augmentation de l'intensité dans la bande verte est caractéristique des zones où les algues vertes sont abondantes.

L'analyse des profils spectraux moyens permet de distinguer les différentes classes de terrain marin en fonction de leurs caractéristiques spectrales uniques. Ces informations sont essentielles pour la classification et la cartographie précises des fonds marins.

### 3.3.5 Cartes de classification (distance euclidienne et SAM)

Les cartes de classification sont générées à partir de l'image fond marin en utilisant deux méthodes différentes : la distance euclidienne et le Spectral Angle Mapper (SAM).

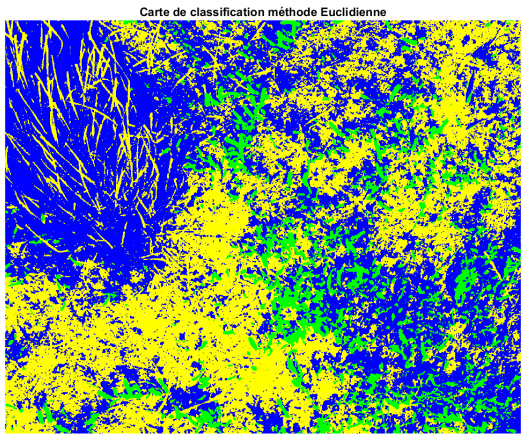


FIGURE 13 – Classification Euclidienne

**Distance Euclidienne :** La classification par distance euclidienne est réalisée en calculant la distance euclidienne entre chaque pixel de l'image et les profils spectraux moyens des classes de terrain marins préalablement définies. Les pixels sont alors assignés à la classe ayant le profil spectral moyen le plus proche en termes de distance euclidienne. La carte de classification obtenue met en évidence les différentes zones du fond marin, telles que le sol, les algues et les algues vertes, en utilisant des couleurs distinctes pour chaque classe.

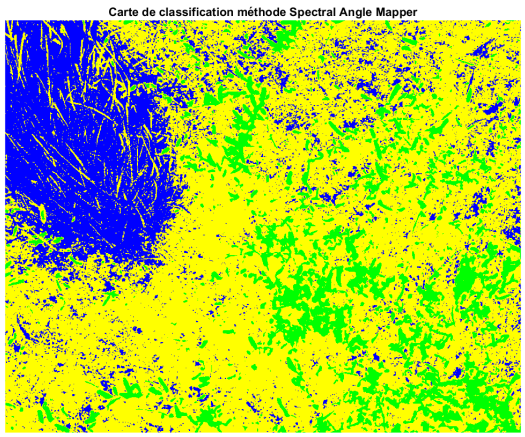


FIGURE 14 – Classification Spectral Angle Mapper

**Spectral Angle Mapper (SAM) :** Le SAM est une méthode de classification qui mesure l'angle spectral entre le spectre de chaque pixel et les profils spectraux moyens des classes de terrain marins. Les pixels sont assignés à la classe dont le profil spectral moyen présente l'angle le plus faible avec le spectre du pixel. La carte de classification SAM met également en évidence les différentes classes de terrain marin, offrant une représentation visuelle des zones de sol, d'algues et d'algues vertes dans l'image.

En comparant les cartes de classification obtenues par les deux méthodes, on observe que la méthode SAM semble fournir des résultats plus précis et plus distincts que la méthode de distance euclidienne. La méthode SAM prend en compte non seulement la distance entre les profils spectraux moyens, mais aussi l'angle entre eux, ce qui permet de mieux discriminer entre les différentes classes de terrain marin.

Par exemple, sur la carte de classification SAM, les frontières entre les différentes zones sont plus nettes et les classes de sol, d'algues et d'algues vertes sont mieux définies. En revanche, la carte de classification basée sur la distance euclidienne présente des frontières moins précises et des zones de chevauchement entre les classes, ce qui peut rendre la distinction entre les différentes caractéristiques du fond marin plus difficile.

En conclusion, la méthode SAM semble offrir une meilleure précision et une meilleure séparation entre les classes de terrain marin, ce qui en fait un choix préférable pour la cartographie des fonds marins à partir d'images sous-marines.

## 3.4 Discussion

**Comparaison des méthodes de classification :**
La comparaison des méthodes de classification utilisées dans ce travail, à savoir la distance euclidienne et le Spectral Angle Mapper (SAM), met en lumière des différences significatives dans leur efficacité pour la cartographie des fonds marins.
La méthode de distance euclidienne repose sur le calcul de la distance euclidienne entre les profils spectraux moyens des différentes classes de terrain marin et le spectre de chaque pixel de l'image. Cette approche est simple à mettre en œuvre et permet une classification relativement rapide. Cependant, elle ne prend pas en compte la direction des vecteurs spectraux, ce qui peut entraîner des confusions entre les classes lorsque les profils spectraux se chevauchent.
En revanche, le SAM mesure l'angle spectral entre le spectre de chaque pixel et les profils spectraux moyens des classes de terrain marin. Cette méthode tient compte à la fois de la magnitude et de la direction des vecteurs spectraux, offrant ainsi une meilleure discrimination entre les différentes classes. Bien que le calcul de l'angle spectral puisse être plus complexe que celui de la distance euclidienne, le SAM produit généralement des résultats plus précis et plus discriminants.

**Évaluation de l'efficacité pour la cartographie des fonds marins :**
L'évaluation de l'efficacité des deux méthodes de classification pour la cartographie des fonds marins montre que le SAM offre généralement des résultats plus précis et plus fiables que la distance euclidienne. En comparant les cartes de classification générées par les deux méthodes, on observe que le SAM permet une meilleure séparation entre les différentes classes de terrain marin, avec des frontières plus nettes et moins de zones de

chevauchement.

**Avantages et inconvénients des méthodes :**
La méthode de distance euclidienne présente l'avantage d'être simple à mettre en œuvre et de calculer rapidement les classifications. Cependant, elle peut être moins efficace pour distinguer les classes de terrain marin lorsque les profils spectraux se chevauchent, ce qui peut entraîner des erreurs de classification.
En revanche, le SAM offre une meilleure discrimination entre les classes de terrain marin en prenant en compte à la fois la magnitude et la direction des vecteurs spectraux. Cependant, cette méthode peut être plus complexe à implémenter et nécessite généralement des calculs plus longs que la distance euclidienne.

En conclusion, bien que la méthode de distance euclidienne soit plus simple et plus rapide, le SAM est souvent préférable pour la cartographie précise des fonds marins grâce à sa capacité à mieux distinguer les différentes caractéristiques du terrain marin.

### 3.5 Conclusion

**Récapitulatif des résultats :**
Ce travail visait à cartographier les fonds marins à partir d'images sous-marines en utilisant des techniques de traitement d'image et de classification spectrale. Les principales étapes comprenaient l'affichage et l'analyse de l'image en couleur, la sélection des échantillons de terrain marin, le calcul des profils spectraux moyens et la classification des pixels de l'image en utilisant deux méthodes différentes : la distance euclidienne et le Spectral Angle Mapper (SAM).
L'analyse des résultats a permis de mettre en évidence les différentes caractéristiques des deux méthodes de classification, ainsi que leurs avantages et inconvénients respectifs. La comparaison des cartes de classification a montré que le SAM offre généralement des résultats plus précis et plus discriminants que la distance euclidienne, grâce à sa capacité à prendre en compte à la fois la magnitude et la direction des vecteurs spectraux.

**Enseignements tirés du TP :**
Ce TP a permis de mettre en pratique plusieurs concepts clés en traitement d'image et en classification spectrale. Il a également mis en évidence l'importance de choisir la bonne méthode de classification en fonction des caractéristiques spécifiques des données et des objectifs de la cartographie.
De plus, ce travail a souligné l'importance de la précision et de la qualité des données d'entrée dans le processus de classification. La sélection judicieuse des échantillons de terrain marin et le calcul précis des profils spectraux moyens sont essentiels pour obtenir des résultats fiables et significatifs.

**Perspectives pour de futurs travaux :**
Pour des futurs travaux, plusieurs pistes d'amélioration peuvent être envisagées. Tout d'abord, il serait intéressant d'explorer d'autres méthodes de classification spectrale, telles que la classification supervisée basée sur les réseaux de neurones ou les méthodes d'apprentissage automatique.
De plus, l'utilisation de données d'imagerie sous-marine de meilleure qualité et de capteurs plus avancés pourrait permettre d'obtenir des résultats encore plus précis et détaillés. Enfin, l'intégration de techniques de segmentation d'image avancées pourrait contribuer à améliorer la précision de la classification en identifiant automatiquement les régions d'intérêt dans l'image.

## 4 Conclusion

Au cours de ces trois TP, nous avons exploré divers aspects de la vision par ordinateur et du traitement d'images à travers des applications pratiques et des outils de programmation tels que Python avec OpenCV et Matlab. Chaque TP a offert une opportunité d'acquérir des compétences précieuses dans le domaine de l'analyse d'images et de la classification spectrale, tout en abordant des applications spécifiques dans des domaines variés.

Dans le premier TP sur la vision industrielle, nous avons appris à compter automatiquement des objets dans une image en utilisant des techniques de traitement d'images telles que la conversion en niveaux de gris, l'histogramme et la binarisation. Cette expérience nous a permis de comprendre les bases de la segmentation d'images et son application dans des scénarios industriels.

Le deuxième TP sur la télédétection par drone nous a plongés dans l'analyse d'images multispectrales acquises par drone pour surveiller la croissance des cultures. En manipulant les différentes bandes spectrales, nous avons exploré des méthodes d'analyse telles que le calcul de l'indice de végétation (NDVI) pour évaluer la santé

des cultures et surveiller leur évolution au fil du temps.

Enfin, dans le troisième TP sur la cartographie des fonds marins par imagerie sous-marine, nous avons abordé des techniques de classification supervisée pour identifier différents types de fonds marins à partir d'images. En comparant les méthodes de classification basées sur la distance euclidienne et le Spectral Angle Mapper (SAM), nous avons évalué les performances de chaque approche pour la cartographie sous-marine.

Dans l'ensemble, ces TP ont offert une expérience riche et variée dans le domaine de la vision par ordinateur et du traitement d'images, fournissant des compétences pratiques et des perspectives intéressantes sur les applications potentielles dans divers domaines scientifiques et industriels. Ils ont également souligné l'importance de la compréhension approfondie des méthodes de traitement d'images et de leur adaptation aux besoins spécifiques de chaque application pour obtenir des résultats précis et significatifs.

# Annexe

Listing 1 – TP1

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# 1. Affichage d'une image
def afficher_image_et_gris(image):
    # Afficher l'image couleur
    cv2.imshow('Image couleur', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # V rifier si l'image a un seul canal
    if len(image.shape) == 2:
        # Si l'image a un seul canal, elle est d j  en niveaux de gris
        image_gris = image
    else:
        # Convertir l'image en niveau de gris
        image_gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Afficher l'image en niveau de gris
    cv2.imshow('Image en niveau de gris', image_gris)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# 2. Histogramme
def afficher_histogramme(image_gris):
    # Calculer l'histogramme de l'image en niveau de gris
    h = cv2.calcHist([image_gris], [0], None, [256], [0, 256])

    # Afficher l'histogramme
    plt.plot(h)
    plt.title('Histogramme de l\'image en niveau de gris')
    plt.show()
    plt.close("all")

# 3. Binarisation
def binariser_image(image_gris, seuil):
    _, image_binarisee = cv2.threshold(image_gris, seuil, 255, cv2.
        THRESH_BINARY)
    return image_binarisee

# 4. Comptage de pixels par bonbon (calibration)
def compter_pixels_par_bonbon(image_binarisee):
```

```
    # Compter les pixels blancs dans l'image binaris e
    nb_pixels_blancs = np.sum(image_binarisee == 255)
    return nb_pixels_blancs

# 5. Comptage de bonbons
def compter_bonbons(image_path, seuil, nb_pixels_par_bonbon_calibre):
    # Charger l'image couleur
    image = cv2.imread(image_path)

    # Afficher l'image couleur
    afficher_image_et_gris(image)

    # Convertir l'image en niveau de gris
    image_gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Afficher l'image en niveau de gris
    afficher_image_et_gris(image_gris)

    # Afficher l'histogramme
    afficher_histogramme(image_gris)

    # Binariser l'image
    image_binarisee = binariser_image(image_gris, seuil)

    # Afficher l'image binaris e
    afficher_image_et_gris(image_binarisee)

    # Compter les pixels blancs dans l'image binaris e
    nb_pixels_blancs = compter_pixels_par_bonbon(image_binarisee)

    # Calculer le nombre de bonbons en utilisant le ratio calibr
    nb_bonbons = round(nb_pixels_blancs / nb_pixels_par_bonbon_calibre)

    return nb_bonbons, nb_pixels_blancs/7

# Chemin de l'image de calibration
image_calibrage_path = 'test_3bb_blancs.png'

# Seuil pour la binarisation (   d terminer)
seuil_calibrage = 150

# Charger l'image de calibration
image_calibrage = cv2.imread(image_calibrage_path)

# Convertir l'image de calibration en niveau de gris
image_gris_calibrage = cv2.cvtColor(image_calibrage, cv2.COLOR_BGR2GRAY)

afficher_histogramme(image_gris_calibrage)

# Binariser l'image de calibration
image_binarisee_calibrage = binariser_image(image_gris_calibrage,
    seuil_calibrage)

afficher_histogramme(image_binarisee_calibrage)
# Compter les pixels blancs dans l'image binaris e de calibration
nb_pixels_blancs_calibrage = compter_pixels_par_bonbon(
    image_binarisee_calibrage)

# Nombre de bonbons dans l'image de calibration (   d terminer manuellement
    )
nb_bonbons_calibrage = 3
```

```
# Calculer le nombre moyen de pixels par bonbon pour la calibration
nb_pixels_par_bonbon_calibre = nb_pixels_blancs_calibrage /
    nb_bonbons_calibrage

# Afficher l'image de calibration
afficher_image_et_gris(image_calibrage)

# Afficher l'image de calibration binaris e
afficher_image_et_gris(image_binarisee_calibrage)


# Chemin de la deuxi me image
image_test_path = 'test_3bb_blancs.png'

# Seuil pour la binarisation de la deuxi me image (   d terminer)
seuil_test = 105

# Compter les bonbons dans la deuxi me image en utilisant le ratio calibr
nb_bonbons_test = compter_bonbons(image_test_path, seuil_test,
    nb_pixels_par_bonbon_calibre)

# Afficher le nombre de bonbons dans la deuxi me image
print("Nombre de bonbons dans la deuxi me image:", nb_bonbons_test)
```

Listing 2 – TP2

```
% Charger les images
donnees1 = load('425aa_2405.mat');
donnees2 = load('425aa_0706.mat');
donnees3 = load('761aa_2306.mat');
image1 = donnees1.I;
image2 = donnees2.I;
image3 = rot90(donnees3.I);

% Afficher la taille de chaque image
taille_image1 = size(image1);
disp(['La taille de l''image 1 est : ', num2str(taille_image1(1)), ' x ', num2str(
    taille_image1(2)), ' pixels ']);
taille_image2 = size(image2);
disp(['La taille de l''image 2 est : ', num2str(taille_image2(1)), ' x ', num2str(
    taille_image2(2)), ' pixels ']);
taille_image3 = size(image3);
disp(['La taille de l''image 3 est : ', num2str(taille_image3(1)), ' x ', num2str(
    taille_image3(2)), ' pixels ']);

% Afficher chaque bande spectrale dans un subplot
figure;
for i = 1:size(image1, 3)
    subplot(2, 3, i);
    imshow(image1(:,:,i), []);
    title(['Bande ', num2str(i), ' - Image 1']);
end

% Cr er les compositions color es en vraies couleurs pour chaque date
vraies_couleurs1 = cat(3, image1(:,:,4), image1(:,:,3), image1(:,:,1)); % Rouge, Vert
    , Bleu
vraies_couleurs2 = cat(3, image2(:,:,4), image2(:,:,3), image2(:,:,1));
vraies_couleurs3 = cat(3, image3(:,:,4), image3(:,:,3), image3(:,:,1));

% Cr er les compositions color es en fausses couleurs pour chaque date
fausses_couleurs1 = cat(3, image1(:,:,6), image1(:,:,3), image1(:,:,5)); % Infrarouge
    , Vert, Bleu
fausses_couleurs2 = cat(3, image2(:,:,6), image2(:,:,3), image2(:,:,5));
fausses_couleurs3 = cat(3, image3(:,:,6), image3(:,:,3), image3(:,:,5));
```

```
% Afficher les compositions color es
figure;
subplot(2, 3, 1);
imshow(vraies_couleurs1);
title('Vraies couleurs - Image 1');
subplot(2, 3, 2);
imshow(fausses_couleurs1);
title('Fausses couleurs - Image 1');
subplot(2, 3, 3);
imshow(vraies_couleurs2);
title('Vraies couleurs - Image 2');
subplot(2, 3, 4);
imshow(fausses_couleurs2);
title('Fausses couleurs - Image 2');
subplot(2, 3, 5);
imshow(vraies_couleurs3);
title('Vraies couleurs - Image 3');
subplot(2, 3, 6);
imshow(fausses_couleurs3);
title('Fausses couleurs - Image 3');

% Afficher les histogrammes des 6 bandes spectrales de l'image 1
figure;
for i = 1:size(image1, 3)
    subplot(2, 3, i);
    histogram(image1(:,:,i), 'Normalization', 'probability');
    title(['Histogramme - Bande ', num2str(i), ' - Image 1']);
    xlabel('Intensit ');
    ylabel('Fr quence normalis e');
end

% Coordonn es d'un pixel de sol et d'un pixel de betterave (utiliser Impixelinfo
    pour les d terminer)
x_sol = 179;
y_sol = 65;
x_betterave = 208;
y_betterave = 50;

% Longueurs d'onde correspondant    chaque bande spectrale
longueurs_donde = [450, 530, 560, 675, 730, 850]; % en nm

% Extraire les profils spectraux des pixels de sol et de betterave
profil_sol = squeeze(image1(y_sol, x_sol, :)); % squeeze pour supprimer les
    dimensions unitaires
profil_betterave = squeeze(image1(y_betterave, x_betterave, :));

% Tracer les profils spectraux avec les longueurs d'onde en abscisses
figure;
plot(longueurs_donde, profil_sol, 'b', 'LineWidth', 2);
hold on;
plot(longueurs_donde, profil_betterave, 'r', 'LineWidth', 2);
hold off;
title('Profils spectraux des pixels de sol et de betterave');
xlabel('Longueur d''onde (nm)');
ylabel('R flectance');
legend('Sol', 'Betterave');

% Calcul de l'indice de v g tation NDVI pour chaque image
R1 = double(image1(:,:,4)); % Bande 675 nm
PIR1 = double(image1(:,:,6)); % Bande 850 nm
NDVI1 = (PIR1 - R1) ./ (PIR1 + R1);

R2 = double(image2(:,:,4));
PIR2 = double(image2(:,:,6));
NDVI2 = (PIR2 - R2) ./ (PIR2 + R2);
```

```
R3 = double(image3(:,:,4));
PIR3 = double(image3(:,:,6));
NDVI3 = (PIR3 - R3) ./ (PIR3 + R3);

figure;
% Afficher les trois NDVI sur une m me subplot
subplot(3,1,1);
imshow(NDVI1);
colormap('jet'); % Utiliser la m me table de couleur 'jet' pour tous les NDVI
colorbar; % Ajouter une  chelle  de couleur
title('Indice de v g tation - Date 1');
subplot(3,1,2);
imshow(NDVI2);
colormap('jet');
colorbar;
title('Indice de v g tation - Date 2');
subplot(3,1,3);
imshow(NDVI3);
colormap('jet');
colorbar;
title('Indice de v g tation - Date 3');

% Seuillage de l'indice de v g tation pour chaque date
seuil1 = 0.4;
seuil2 = 0.4;
seuil3 = 0.4;

culture1 = NDVI1 > seuil1;
culture2 = NDVI2 > seuil2;
culture3 = NDVI3 > seuil3;

% Calcul de la proportion de culture pour chaque date
proportion_culture1 = sum(culture1(:)) / numel(NDVI1);
proportion_culture2 = sum(culture2(:)) / numel(NDVI2);
proportion_culture3 = sum(culture3(:)) / numel(NDVI3);

% Affichage des proportions de culture au cours du temps
proportions = [proportion_culture1, proportion_culture2, proportion_culture3];
dates = {'Date 1', 'Date 2', 'Date 3'};
figure;
bar(proportions);
set(gca, 'XTickLabel', dates);
xlabel('Date');
ylabel('Proportion de culture');
title('Proportion de culture au cours du temps');
```