Rapport de TP : Traitement d'image

SOMMACAL Mathis
M2 EEA Parc I3A
Année universitaire 2024/2025

Université Perpignan Via Domitia
CRÉATRICE D'AVENIRS DEPUIS 1350

Table des matières

I TP1 **3**
- 0.1 Introduction . . . 4
- 0.2 Explication du Code . . . 4
- 0.3 Conclusion . . . 5

II TP 2 : Détection de véhicules **6**

III TP 3 : Images sattelites **10**
- 0.4 Résultats . . . 11
- 0.5 Conclusion . . . 11

IV TP 4 : Détection des contours **13**
- 0.6 Introduction . . . 14
- 0.7 Exercice 1 : Filtres gradients . . . 14
  - 0.7.1 Filtres de base sans toolbox . . . 14
  - 0.7.2 Combinaison des filtres . . . 14
- 0.8 Filtres de Sobel . . . 15
- 0.9 Filtre moyenneur . . . 15
- 0.10 Conclusion . . . 15

Université Perpignan Via Domitia
CRÉATRICE D'AVENIRS DEPUIS 1350

Introduction

Ce document regroupe l'ensemble des travaux pratiques réalisés dans le cadre du module de traitement d'images. Ces travaux ont permis de développer une compréhension approfondie des techniques fondamentales et avancées de manipulation et d'analyse d'images numériques.

À travers des exercices variés, nous avons exploré des concepts tels que les filtres de détection de contours, les transformations d'images en niveaux de gris, la suppression de bruits, l'extraction de caractéristiques et l'utilisation d'algorithmes d'amélioration d'images. Chaque TP visait à consolider des compétences pratiques en programmation MATLAB tout en approfondissant les connaissances théoriques nécessaires pour interpréter et analyser les résultats obtenus.

L'objectif de ce rapport est de présenter de manière synthétique les méthodes, les résultats, ainsi que les conclusions tirées de chaque TP, tout en mettant en avant les apprentissages clés et les perspectives pour le domaine de l'analyse d'images.

Première partie

TP1

0.1 Introduction

Dans ce TP, nous allons nous initier à différentes techniques de traitement d'image sur MATLAB. Dans un premier temps, nous avons créé et manipulé des images, chaque image pouvant être considérée comme un tableau de pixels. Nous avons dessiné des formes géométriques simples telles que des carrés et des rectangles, et les avons placées sur l'image pour nous entraîner à manipuler les données d'image.

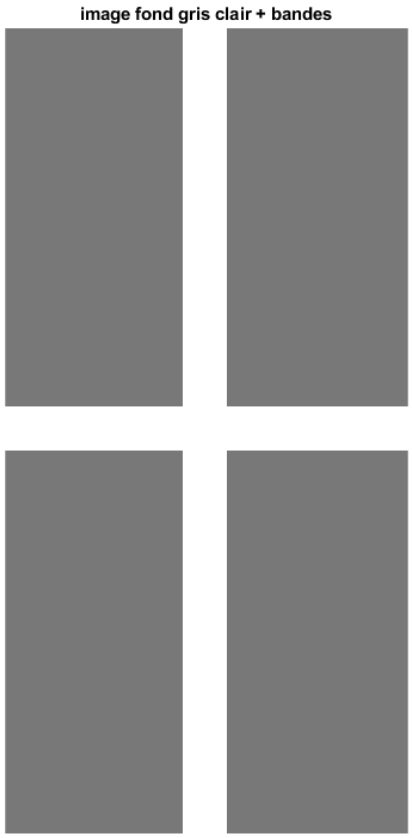

FIGURE 1 – Manipulation d'image

0.2 Explication du Code

Le code suivant est utilisé pour lire une image, copier son contenu, et réduire progressivement les valeurs des canaux de couleur rouge, vert et bleu.

```
Ig2 = imread('im1.jpg'); % Lire l'image 'im1.jpg'
I3 = Ig2; % Copier l'image
for i = 1:20
    I3(:,:,1) = Ig2(:,:,1)-i*10; % R duire la valeur du canal rouge
    I3(:,:,2) = Ig2(:,:,2)-i*10; % R duire la valeur du canal vert
    I3(:,:,3) = Ig2(:,:,3)-i*10; % R duire la valeur du canal bleu
    imshow(I3) % Afficher l'image
    pause(0.5) % Pause de 0.5 secondes
end
```

— `Ig2 = imread('im1.jpg');` : Cette ligne lit l'image `im1.jpg` et la stocke dans la variable `Ig2`.
— `I3 = Ig2;` : Cette ligne copie le contenu de `Ig2` dans la variable `I3`.
— `for i = 1:20` : Cette boucle `for` s'exécute 20 fois.
— `I3(:,:,1) = Ig2(:,:,1)-i*10;` : À chaque itération, la valeur du canal rouge est réduite de `i*10`.
— `I3(:,:,2) = Ig2(:,:,2)-i*10;` : À chaque itération, la valeur du canal vert est réduite de `i*10`.
— `I3(:,:,3) = Ig2(:,:,3)-i*10;` : À chaque itération, la valeur du canal bleu est réduite de `i*10`.
— `imshow(I3);` : Cette ligne affiche l'image modifiée.
— `pause(0.5);` : Cette ligne met en pause l'exécution du code pendant 0.5 secondes pour permettre de visualiser l'animation.

0.3 Conclusion

Ce code MATLAB permet de créer une animation où une image passe progressivement du blanc au noir en réduisant les valeurs de ses canaux de couleur. Chaque itération de la boucle `for` diminue les valeurs des canaux de couleur, créant ainsi un effet de fondu.

Université Perpignan Via Domitia
CRÉATRICE D'AVENIRS DEPUIS 1350

Deuxième partie

TP 2 : Détection de véhicules

Introduction

Dans ce TP, nous avons pour objectif de détecter et de suivre les véhicules dans une vidéo en utilisant des techniques de soustraction d'image et de segmentation. Pour cela, nous allons utiliser une approche basée sur des régions connectées pour identifier les objets en mouvement.

Structure du Code

Le programme principal est divisé en plusieurs sections :

1. Initialisation et lecture des images.
2. Détection des régions en mouvement.
3. Mise à jour des informations des véhicules détectés.
4. Affichage des résultats et comptage des véhicules.

Initialisation et Lecture des Images

```
clc;
close all;
clear;
```

Ces commandes permettent de nettoyer l'environnement MATLAB avant d'exécuter le programme.

```
function detectAndTrackVehicles2(videoPath)
    % Lire la vid o
    V = VideoReader(videoPath);

    % Lire la premi re image et convertir en niveaux de gris
    frame1 = rgb2gray(readFrame(V));
```

Explication :

— La fonction `VideoReader` permet de charger une vidéo depuis le chemin spécifié (`videoPath`).
— La première image est convertie en niveaux de gris pour simplifier les calculs.

Détection des Régions en Mouvement

```
    % Soustraction d'image pour d tection de mouvement
    diffImage2 = abs(double(frame2_g) - double(frame1));

    % Seuil pour binariser l'image
    threshold2 = 26;
    binaryImage2 = diffImage2 > threshold2;
```

Explication :

— La soustraction d'image permet d'identifier les zones de changement entre deux images consécutives.
— Un seuil est appliqué pour binariser l'image résultante, isolant ainsi les régions en mouvement.

Détection des Régions Connectées

```
function regions2 = detectConnectedRegions2(binaryImage2, min_area2)
    % Fonction pour d tecter les r gions connect es sans toolbox
    ...
```

FIGURE 2 – Détection des changements de valeurs de pixels entre 2 images

Explication :

— Cette fonction explore les pixels connectés dans l'image binaire.

— Les régions trop petites (`min_area2`) sont éliminées pour éviter les faux positifs.

Mise à Jour des Informations des Véhicules

```
    for k = 1:size(regions2, 1)
        % Obtenir le rectangle et le centre de la r gion
        ...
    end
```

Explication :

— Chaque région détectée est comparée avec les véhicules existants pour déterminer s'il s'agit d'un nouveau véhicule ou d'un véhicule déjà détecté.

— Les nouvelles régions sont assignées à un identifiant (`ID`) unique.

Affichage des Résultats et Comptage des Véhicules

```
    plot([line_x1 line_x2], [line_y2 line_y2], 'Color', line_color2, '
        LineWidth', 2);
    text(10, 10, ['Compteur de v hicules : ' num2str(compteur2)], 'Color',
         'g', 'FontSize', 6);
```

Explication :

— Une ligne de détection est tracée sur l'image, et un compteur est incrémenté lorsque des véhicules traversent cette ligne.

— Les résultats sont affichés en temps réel avec des rectangles autour des véhicules détectés.

Conclusion

Ce programme MATLAB illustre une méthode efficace pour détecter et suivre des véhicules dans une vidéo en utilisant des techniques de traitement d'image. Bien que simplifié, il peut être étendu pour intégrer des approches plus avancées, comme l'utilisation de réseaux de neurones pour la classification des objets.

FIGURE 3 – Image de la détection

Université Perpignan Via Domitia
CRÉATRICE D'AVENIRS DEPUIS 1350

Troisième partie

TP 3 : Images sattelites

Introduction

Dans ce TP, nous allons nous attaquer à un problème essentiel dans l'analyse d'images satellitaires : la suppression des nuages présents dans des séries d'images prises à la même heure sur plusieurs jours. La présence de nuages peut altérer l'interprétation et l'analyse des données géographiques ou climatiques extraites des images satellitaires.

Pour résoudre ce problème, nous mettrons en œuvre une méthode simple mais efficace :

1. **Conversion en niveaux de gris :** Les images en couleur (RGB) seront converties en niveaux de gris à l'aide de la fonction MATLAB `rgb2gray`. Cette étape permet de réduire la complexité des données tout en conservant les informations essentielles nécessaires pour identifier les zones couvertes par des nuages.
2. **Agrégation par minimisation :** Une fois les images converties, nous utiliserons la fonction MATLAB `min` pour comparer pixel par pixel les intensités de plusieurs images prises à la même heure sur des jours différents. Cette méthode repose sur le fait que les nuages, généralement plus clairs que les autres éléments de l'image (comme la végétation, l'eau ou le sol), peuvent être éliminés en sélectionnant les valeurs minimales d'intensité pour chaque pixel à travers l'ensemble des images.

Au final, cette approche produira une image composite « sans nuages », qui représente les éléments fixes et récurrents de la scène observée, comme le sol, les rivières, ou la végétation. Cette image peut ensuite être utilisée pour des applications telles que la cartographie, l'étude des écosystèmes ou la modélisation climatique.

0.4 Résultats

Dans cette section, nous présentons les résultats obtenus avant et après le traitement des images satellitaires. L'image de base contient des nuages, tandis que l'image traitée montre une scène sans nuages grâce à l'agrégation des valeurs minimales d'intensité.

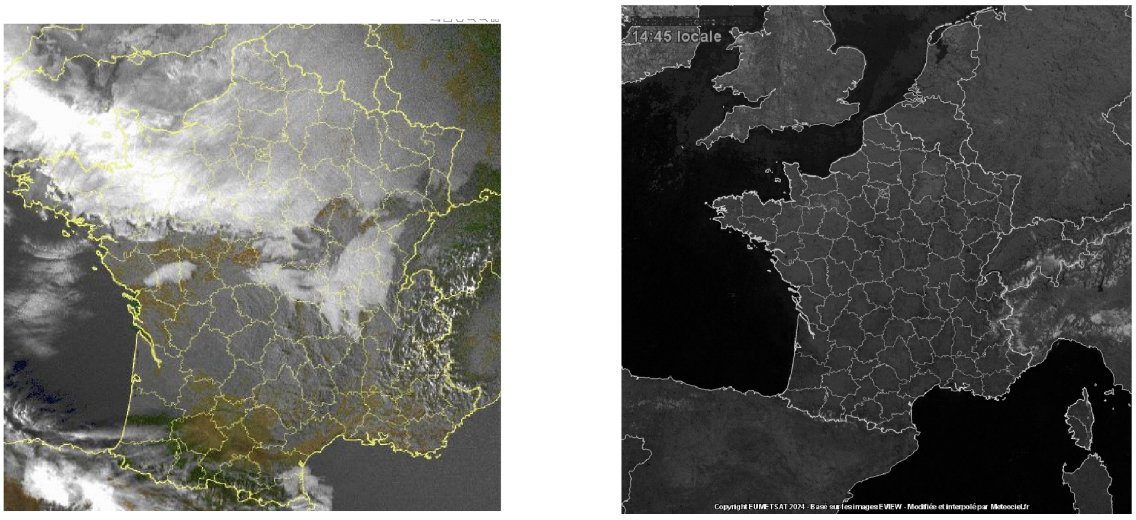

FIGURE 4 – Comparaison entre l'image de base (à gauche) et l'image traitée (à droite).

0.5 Conclusion

Ce TP a permis de mettre en œuvre une méthode efficace pour supprimer les nuages présents dans des images satellitaires. En combinant la conversion en niveaux de gris et l'agrégation par la minimisation des intensités pixel par pixel, nous avons obtenu une image finale claire et sans artefacts liés aux nuages.

Cette technique, bien qu'élémentaire, est particulièrement utile pour améliorer la qualité des données géographiques et climatiques, facilitant ainsi leur utilisation dans des domaines tels que la cartographie, la modélisation des écosystèmes, ou encore la prévision climatique. Ce travail nous a également permis de renforcer notre maîtrise de MATLAB pour le traitement et l'analyse d'images.

Université Perpignan Via Domitia
CRÉATRICE D'AVENIRS DEPUIS 1350

Quatrième partie

TP 4 : Détection des contours

0.6 Introduction

La détection de contours vise à identifier les points d'une image où se produisent des variations abruptes d'intensité lumineuse. Ces variations traduisent généralement des éléments significatifs, tels que des événements visuels importants ou des formes à détecter.

Certaines méthodes de détection s'appuient sur l'application de filtres linéaires, une opération souvent désignée sous le nom de rehaussement de contours. Ces filtres sont définis par un masque $h(m, n)$ de dimensions $k \times k$, qui est appliqué à l'image pour accentuer les zones de transitions lumineuses marquées.

0.7 Exercice 1 : Filtres gradients

0.7.1 Filtres de base sans toolbox

L'objectif est d'appliquer des filtres simples pour détecter les contours horizontaux et verticaux dans une image. Ces filtres sont définis par les masques suivants :

— $h_1 = [1 \ \ -1]$: Filtre horizontal.

— $h_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$: Filtre vertical.

Résultats

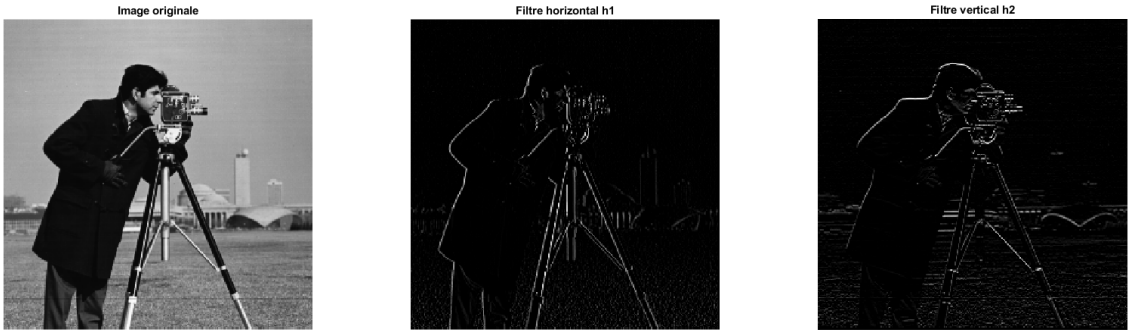

FIGURE 5 – De gauche à droite : Image originale, filtre horizontal (h_1), filtre vertical (h_2).

0.7.2 Combinaison des filtres

Les résultats obtenus en combinant les filtres h_1 et h_2 sont calculés par :

$$I_{\text{comb}}(i, j) = \sqrt{I_{h1}(i, j)^2 + I_{h2}(i, j)^2}.$$

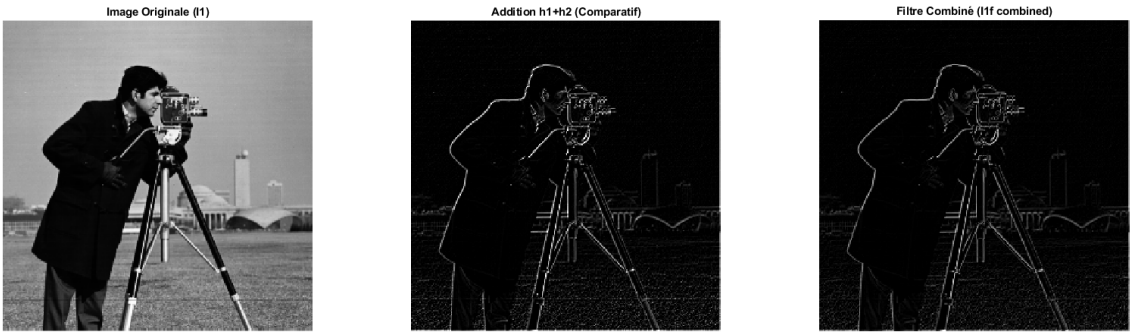

FIGURE 6 – À gauche : Addition des résultats des filtres h_1 et h_2. À droite : Filtre combiné.

0.8 Filtres de Sobel

Les filtres de Sobel permettent une détection des contours plus précise grâce à des masques de convolution plus complexes. Voici les masques utilisés :

- $h_3 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$: Filtre horizontal.
- $h_4 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$: Filtre diagonal.
- $h_5 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$: Filtre vertical.
- $h_6 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$: Filtre anti-diagonal.

Résultats

FIGURE 7 – Résultats des filtres de Sobel : h_3 (gauche) et h_4 (droite).

0.9 Filtre moyenneur

Le filtre moyenneur est appliqué pour lisser l'image en remplaçant chaque pixel par la moyenne des pixels environnants dans une fenêtre 3×3.

Résultats

0.10 Conclusion

Ce TP a permis d'explorer diverses techniques de traitement d'image pour la détection de contours et le lissage. Les filtres simples (h_1, h_2) offrent une première approche pour isoler les variations horizontales et verticales. Les filtres de Sobel améliorent la précision en intégrant des gradients diagonaux et verticaux. Enfin, le filtre moyenneur démontre son efficacité pour réduire le bruit tout en conservant une partie des détails structurels.

Ces outils sont fondamentaux pour de nombreuses applications, telles que la reconnaissance de formes, l'amélioration d'images ou encore la segmentation d'objets.

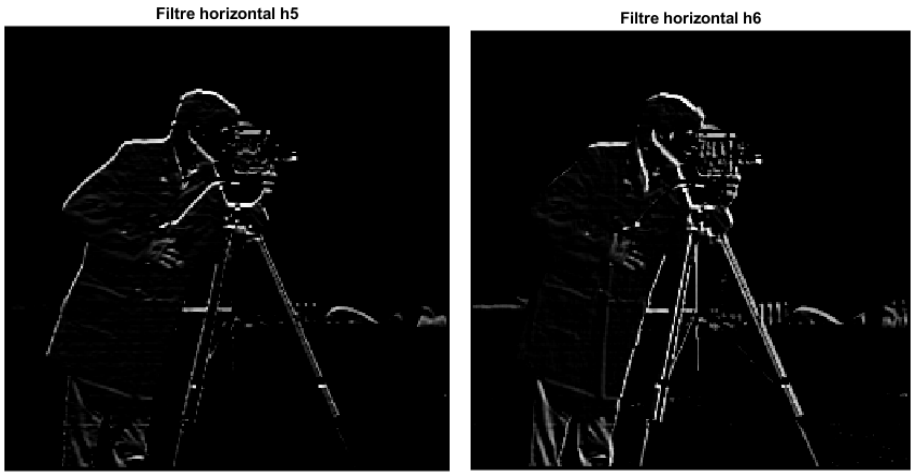

FIGURE 8 – Résultats des filtres de Sobel : h_5 (gauche) et h_6 (droite).

FIGURE 9 – À gauche : Image originale. À droite : Image après application du filtre moyenneur.