Cours : INFO4304 - Vision par ordinateur
Travail : Equipe de 2
Note : 100 points

Date de livraison : 5 novembre 2025, 23h30 (-3 pts/jour de retard)
Outil : OpenCV (C++ ou Python)

Partie 1 : Fusion par Pyramide Laplacienne

Introduction et rappels Théoriques

La **fusion d'images** vise à créer une image unique et plus informative en combinant des données de plusieurs images d'une même scène. La méthode par **Pyramide Laplacienne** est un classique pour réaliser une fusion sans couture (*seamless blending*).

Concepts Clés

1. **Pyramide Gaussienne (GP)** : Une séquence d'images où chaque niveau est obtenu en floutant l'image précédente et en la sous-échantillonnant (pyrDown).
2. **Pyramide Laplacienne (LP)** : Représente les **détails** perdus lors de la construction de la pyramide Gaussienne. Elle est la différence entre un niveau Gaussien et la version étendue (pyrUp) du niveau suivant.

   $$L_i = G_i - \text{pyrUp}(G_{i+1})$$
3. **Reconstruction** : L'image originale peut être entièrement reconstruite en ajoutant itérativement les Laplaciens étendus au niveau de base (le dernier niveau de la GP).

Objectif de cette partie

Implémenter la technique de fusion d'images par Pyramide Laplacienne afin de combiner des images prises sous différentes conditions (par exemple, exposition). Le code doit être **modulaire** et **généralisable**.

I. Fusion par Pyramide Laplacienne de 2 Images

1. Pyramide Laplacienne

Implémenter une fonction **modulaire** build_laplacian_pyramid(image, levels) qui retourne la pyramide Laplacienne de l'image. Vous pouvez utiliser les fonctions pyrDown et pyrUp d'OpenCV.

2. Calcul des Maximums des Laplaciens

Implémenter la fonction max_lap_pyr(lp1,lp2) qui prend deux pyramides Laplaciennes (lp1, lp2) et retourne une liste (pyramide) de maximums (MaxLp).

1. Calculer lp1 et lp2.
2. Pour **chaque niveau supérieur** (Laplaciens, mais pas le dernier niveau), comparer pixel par pixel les deux Laplaciens et garder l'image où le pixel a la **magnitude maximale**.

3. Fusion de l'image du dernier niveau

Calculer l'image FusLastLevel en prenant la **moyenne arithmétique** des deux images du dernier niveau (les plus petites, qui correspondent aux derniers niveaux des Pyramides Gaussiennes).

4. Fusion et Reconstruction

Implémenter la fonction fus_lap_pyr(Image1,Image2,Level).

- **Étapes de Reconstruction** :
  1. Initialiser l'image fusionnée actuelle avec FusLastLevel (section 3).
  2. Itérer à rebours : Étendre l'image actuelle en utilisant pyrUp.
  3. Ajouter à ce résultat l'image Laplacienne maximum (MaxLp) du niveau correspondant (section 2).
  4. Répéter les étapes 2 et 3 jusqu'au niveau de l'image originale. Le dernier niveau étendu est le résultat de la fusion.

5. Tests de fusion

Utiliser les images fournies et comparer la fusion pour 3 paires. Tester les niveaux de pyramide **4, 5 et 6**.

II. Fusion Multi-Images (N images)

Généraliser l'approche précédente pour fusionner N images.

La fonction à implémenter : FusImg=fuseLap_N_Images (ImageList,NumberImages,Level).

1. **Fusion des Laplaciens** : Pour chaque niveau, sélectionner le Laplacien dont la magnitude (valeur absolue) est la **maximale** parmi les N images.
2. **Fusion du Dernier Niveau** : Calculer la **moyenne arithmétique** des N images du dernier niveau Gaussien.
3. **Reconstruction** : Utiliser la même méthode d'expansion et d'addition que dans la section I.4.

4. Tests et Analyse

Tester la fusion sur les dossiers fournis avec les niveaux **4, 5 et 6**. **Analyser et commenter votre résultat** dans le rapport.

III. Livrables

Deux dossiers séparés (Fus_2images, Fus_Nimages) dans un fichier .zip. Le code doit être commenté et fonctionnel.

Rapport dans lequel vous repondez aux questions et commentez les resultats obtenus pour chaque scenario.

Partie 2: Détection de visages avec OpenCV (35 points)

Objectif de cette partie :

L'objectif de cette partie est de comprendre et d'implémenter une méthode de détection de visages à l'aide de la bibliothèque OpenCV. Les étudiants apprendront à :
- Charger et manipuler une image avec OpenCV.
- Utiliser des classifieurs en cascade de Haar (CascadeClassifier) pour détecter des visages.
- Afficher et interpréter les résultats de la détection.
- Ajuster les paramètres pour améliorer les performances et la visualisation.

Concepts manipulés

- Vision par ordinateur : Détection d'objets dans une image à l'aide d'un modèle préentraîné.
- Cascade de Haar : Méthode classique de détection de visages reposant sur des caractéristiques simples.
- Traitement d'images : Conversion en niveaux de gris, redimensionnement, affichage.
- Coordonnées et repères d'images : Gestion des rectangles de détection et transformation des coordonnées lors du flip horizontal.

Matériel nécessaire

- Python 3
- Bibliothèque OpenCV (cv2)
- Fichiers XML de classifieurs préentraînés : haarcascade_frontalface_alt2.xml et haarcascade_profileface.xml (disponibles sur clic et peuvent etre telecharges depuis opencv/data/haarcascades at master · opencv/opencv · GitHub)
- Des images d'entrée

Introduction:

La cascade de Haar est une méthode classique de détection d'objets, notamment utilisée pour la détection de visages.
Elle repose sur l'utilisation de caractéristiques simples, appelées *features de Haar*, qui comparent la différence d'intensité lumineuse entre des zones claires et sombres d'une image (par exemple, entre les yeux et les joues).

Ces caractéristiques sont ensuite combinées dans une cascade de classifieurs entraînée à l'aide d'un grand nombre d'exemples de visages et de non-visages.
Cette approche permet de rejeter rapidement les régions sans visage et de concentrer le calcul sur les zones les plus probables, rendant la détection à la fois rapide et efficace.

Bien que remplacée aujourd'hui par des méthodes plus modernes comme les réseaux de neurones convolutionnels, la cascade de Haar reste un excellent outil pour comprendre les bases de la vision par ordinateur et des classifieurs d'images.

Réalisation :

Étape 1 – Chargement et préparation de l'image

1. Importez les bibliothèques nécessaires (cv2, operator, sys).
2. Chargez l'image avec cv2.imread() et vérifiez son ouverture.
3. Convertissez l'image en niveaux de gris (cv2.cvtColor()).
4. Affichez l'image d'origine pour vérifier son chargement.

Question 1: Pourquoi est-il préférable de travailler sur une image en niveaux de gris pour la détection de visages Haar ?

Étape 2 – Détection des visages

1. Créez les objets classifieurs pour le visage frontal et le profil (face_cascade et profile_cascade) en utilisant la fonction cv2.CascadeClassifier().
2. Appliquez la méthode detectMultiScale() pour détecter les visages frontaux.
3. Répétez la détection pour les profils de visage (face de profil).
4. Appliquez une transformation geometrique sur l'image (symetrie horizontale) (cv2.flip()) et appliquez à nouveau la détection de profil.

Question 2: Pourquoi retourne-t-on l'image horizontalement pour la détection des profils ?
Question 3: Quels sont les rôles des paramètres scaleFactor et minNeighbors dans detectMultiScale() ?

Étape 3 – Traitement et affichage des résultats

1. Concaténez toutes les détections dans une seule liste list_face.
2. Dessinez un rectangle rouge autour de chaque visage détecté :
   cv2.rectangle(image, (x, y), (x2, y2), (0, 0, 255), 2)
4. Affichez le nombre total de visages détectés.

Question 3: Que représentent les quatre coordonnées (x, y, x2, y2) ? Que representent les nombres (0,0,255) ?
Question 4: Que se passe-t-il si plusieurs rectangles se chevauchent ? Comment pourrait-on les fusionner ?

Étape 4 – Amélioration de la visualisation

1. Redimensionnez l'image pour l'affichage (par exemple, réduction à 25% de la taille originale).
2. Affichez l'image annotée dans une fenêtre OpenCV.
3. Essayez d'augmenter l'épaisseur du rectangle (thickness=4 ou 6) et observez la différence.

Question 5: Pourquoi le redimensionnement est-il utile lors de l'affichage d'images de grande taille ?

Étape 5 – Validation et astuces

• Vérifiez que tous les visages présents sur l'image ont bien été détectés.
• Si certains manquent, essayez de modifier les paramètres de détection (scaleFactor, minNeighbors) pour améliorer les résultats.

Bonus (15 points points)

- Ajoutez un numéro au-dessus de chaque visage détecté.
-Au lieu d'utiliser une image statique, utiliser une capture video continue a partir de votre webcam comme entree de votre programme. Le resultat devrait detecter les visages en temps reel.
- Implémentez une version améliorée utilisant la détection DNN d'OpenCV (cv2.dnn).

Livrables

- Le script Python complet, fonctionnel et commenté. Nommez le fichier detect_visag_stat.py
- L'image annotée avec les rectangles autour des visages.
- Un rapport répondant aux questions posées. N'oubliez pas de commenter les resultats obtenus pour chaque image de test.