TP6

Objectifs :

1. Comprendre le flot optique et comment il peut être utilisé pour analyser les mouvements dans les vidéos.
2. Implémenter deux méthodes différentes pour calculer le flot optique :
   - **Farneback** : une méthode classique basée sur les pyramides gaussiennes.
   - **RAFT** : un modèle plus avancé qui prédit le flot optique en utilisant des réseaux neuronaux convolutifs et récurrents.

Partie 1 : Calcul du flot optique avec la méthode de Farneback

1.1. Introduction

La méthode de Farneback est une méthode classique utilisée pour calculer le flot optique dense. Elle repose sur une estimation des déplacements de pixels entre deux frames successives à l'aide de la méthode des pyramides gaussiennes.

1.2. Implémentation

Utilisez la bibliothèque OpenCV pour implémenter le flot optique de Farneback.

Code d'implémentation :

```
import cv2
import numpy as np

# Charger les vidéos
cap = cv2.VideoCapture(0)
# Lire la première image
ret, frame1 = cap.read()
prev_gray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
# Préparer pour afficher les résultats
while(cap.isOpened()):
    ret, frame2 = cap.read()
    if not ret:
        break
    next_gray = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
    # Calcul du flot optique avec Farneback
    flow = cv2.calcOpticalFlowFarneback(prev_gray, next_gray, None, 0.5, 3, 15, 3, 5, 1.2, 0)
    # Visualiser le flot optique avec une carte de couleurs
    magnitude, angle = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    hsv = np.zeros_like(frame1)
    hsv[..., 1] = 255
    hsv[..., 0] = angle * 180 / np.pi / 2
    hsv[..., 2] = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)
    flow_rgb = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

```
    cv2.imshow('Flot optique Farneback', flow_rgb)
    # Passer à l'image suivante
    prev_gray = next_gray
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

1.3. Question sur le code

Changer les paramètres de la méthode calcOpticalFlowFarneback() pour voir leurs effets

Partie 2 : Implémentation du flot optique avec RAFT

3.1. Introduction

RAFT (Recurrent All-Pairs Field Transforms) est un modèle avancé pour prédire le flot optique. Il utilise une approche récurrente pour améliorer les prédictions du flot optique entre des images successives.

3.2. Installation des dépendances

- pip install torch torchvision opencv-python
  Clonez le dépôt de RAFT:
- git clone https://github.com/princeton-vl/RAFT
  cd RAFT
- Télécharger les modèles RAFT :
  https://dl.dropboxusercontent.com/s/4j4z58wuv8o0mfz/models.zip
- Décompresser le fichier zip dans le répertoire RAFT

1. Utiliser la ligne de commande suivante pour tester sur les images du répertoire « demo-frames » :

   python demo.py --model=models/raft-things.pth --path=demo-frames

2. Exécuter le script python permettant de tester un modèle RAFT sur une vidéo réelle (webcam ou vidéo sur HD).

```
import sys
import os
import argparse
import torch
import cv2
import numpy as np
from collections import OrderedDict

# Ajouter RAFT au path
raft_root = r"chemin\RAFT"
sys.path.append(raft_root)
sys.path.append(os.path.join(raft_root, "core"))
sys.path.append(os.path.join(raft_root, "utils"))
from raft import RAFT
from utils.utils import InputPadder

# Arguments RAFT
parser = argparse.ArgumentParser()
```

```
parser.add_argument('--small', action='store_true')
parser.add_argument('--mixed_precision', action='store_true')
parser.add_argument('--dropout', type=float, default=0.0)
args = parser.parse_args([])
args.small = False   # mettre True si vous voulez RAFT-small

# Charger modèle RAFT sur CPU
device = "cpu"
model = RAFT(args)
state_dict = torch.load(
    os.path.join(raft_root, "models/raft-things.pth"),
    map_location=device
)
# Correction des clés ("module.")
new_state_dict = OrderedDict()
for k, v in state_dict.items():
    name = k[7:] if k.startswith("module.") else k
    new_state_dict[name] = v
model.load_state_dict(new_state_dict)
model = model.to(device).eval()
# Webcam
cap = cv2.VideoCapture(0)
ret, prev_frame = cap.read()
prev_frame = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2RGB)
prev_frame = torch.from_numpy(prev_frame).float() / 255.0
prev_frame = prev_frame.permute(2, 0, 1)[None].to(device)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_tensor = torch.from_numpy(frame_rgb).float() / 255.0
    frame_tensor = frame_tensor.permute(2, 0, 1)[None].to(device)
    # Padding pour chaque paire d'images
    padder = InputPadder(prev_frame.shape)
    prev_pad, frame_pad = padder.pad(prev_frame, frame_tensor)
    with torch.no_grad():
        _, flow_up = model(prev_pad, frame_pad, iters=12, test_mode=True)
        # iters=12 pour accélérer sur CPU
    flow = flow_up[0].permute(1, 2, 0).cpu().numpy()
    # Visualisation HSV
    mag, ang = cv2.cartToPolar(flow[..., 0], flow[..., 1])
    hsv = np.zeros((flow.shape[0], flow.shape[1], 3), dtype=np.uint8)
    hsv[..., 0] = ang * 180 / np.pi / 2
    hsv[..., 1] = 255
    hsv[..., 2] = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX)
    bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
    cv2.imshow("RAFT Optical Flow (CPU)", bgr)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
    prev_frame = frame_tensor
cap.release()
cv2.destroyAllWindows()
```

2.1. Modifiez la valeur de l'attribut **raft_root** en remplaçant « chemin » par le chemin **absolu** menant au dossier RAFT sur votre machine.

2.2. Remplacez le modèle pré-entraîné **raft-things.pth** par l'un des modèles disponibles dans le répertoire **models** fourni avec le projet.

2.3. Remplacez la source d'entrée (**webcam**) par un fichier vidéo plutôt que d'un flux live.

2.4. Adaptez la visualisation du flux optique en modifiant les composantes de la représentation HSV :

- **Hue** → hsv[..., 0]
- **Saturation** → hsv[..., 1]
- **Value** → hsv[..., 2]

2.5. Si votre ordinateur dispose d'une carte graphique **NVIDIA GeForce**, configurez RAFT pour effectuer l'inférence sur le **GPU**. Pour cela, transférez le modèle vers le GPU en utilisant la commande suivante :

model = model.eval().cuda()