Traitement d’images

Cours 4 - Séparations des plans d’une image

Table des matières

Introduction

La séparation des plans d'une image est une tâche fondamentale en traitement d'images et en vision par ordinateur. Dans cette partie, nous nous concentrerons uniquement sur la séparation entre le premier-plan et l'arrière-plan d'une image. En anglais ces plans sont appelés **foreground** et **background** et la séparation de ces plans est appelée « background substraction ».

Il est important de noter que le terme « séparation » des plans est très générique et peut englober diverses techniques et approches mais le coeur de la tâche est de distinguer les objets d'intérêt du reste de l'image.

FIGURE 1 – Exemple de séparation premier/arrière-plan, à gauche l'image originale, à droite après traitement.

Dans la figure 1, nous pouvons voir que l'algorithme a réussi globalement à isoler les personnes mais pas parfaitement, en effet, l'algorithme a également exclu les ombres portées au sol par les personnes mais également le bandeau de sécurité.

? Selon vous, pourquoi le bandeau de sécurité et les ombres portées ont-ils été inclus dans le premier-plan dans la figure 1 ?

Les applications sont multitples, notamment :

- Pré-traitement pour la détection ou le suivi d'objets
- Surveillance vidéo
- Réalité augmentée
- Cinéma et effets spéciaux
- Conférences vidéo
- Photographie
- ...

Une première approche

La séparation des plans repose souvent sur certaines suppositions de base concernant les données à disposition. En fonction des données à disposition des approches plus ou moins simples peuvent être envisagées.

Les suppositions

Nous supposerons que nous avons à disposition une vidéo issue d'une caméra fixe. C'est à dire que nous avons une suite d'images (les frames de la vidéo) où le décor ne change pas, seuls les objets en mouvement changent d'une image à l'autre.

Modélisation statistique de l'arrière-plan

Nous avons vu qu'une image est une matrice de pixels, chaque pixel étant représenté par une ou plusieurs valeurs. Nous pouvons envisager une vidéo comme une suite d'images, c'est à dire une nouvelle matrice avec une dimension supplémentaire représentant le temps.

L'approche la plus intuitive pour modéliser l'arrière-plan est de considérer, pour chaque pixel, la valeur moyenne de ce pixel : c'est à dire que que nous allons calculer une image moyenne. L'intuition derrière cette approche est que les objets intéressants (le premier-plan) sont en mouvement et n'apparaissent pas toujours au même endroit, tandis que l'arrière-plan reste constant, ainsi, en moyennant les images, les objets en mouvement devraient ne pas apparaître dans l'image moyenne.

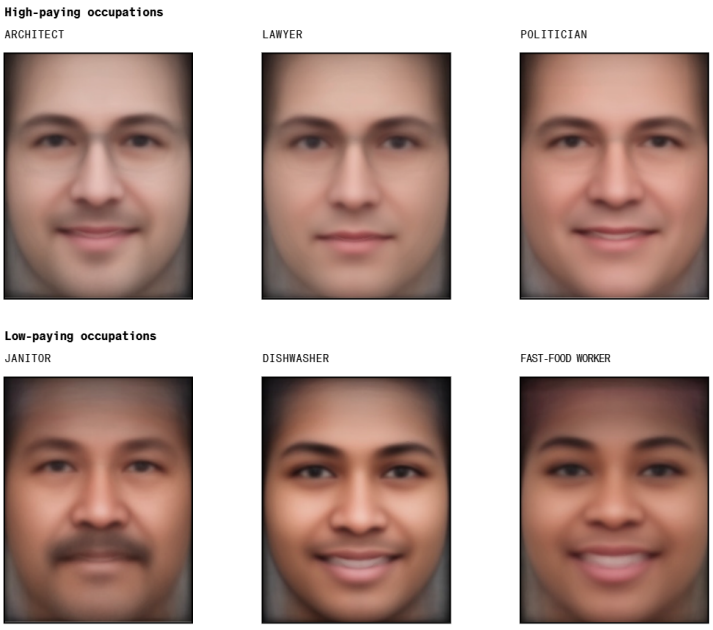

FIGURE 2 – Moyenne d'images générées pour une intelligence artificielle générative (Stable diffusion) afin d'en exhiber les biais [2]

Une autre approche possible est de considérer la médiane des valeurs de chaque pixel à travers les images. La médiane est moins sensible aux valeurs extrêmes, ce qui peut être avantageux si les objets en mouvement sont très lumineux ou très sombres par rapport à l'arrière-plan.

2. Humans Are Biased. Generative AI Is Even Worse

Soustraction de l'arrière plan

Une fois que nous avons une estimation de l'arrière-plan (par moyenne ou médiane), nous pouvons soustraire cette image d'arrière-plan de chaque image de la vidéo. L'idée est que les pixels correspondant à l'arrière-plan auront des valeurs proches de zéro après la soustraction, tandis que les pixels du premier-plan auront des valeurs significativement différentes de zéro.

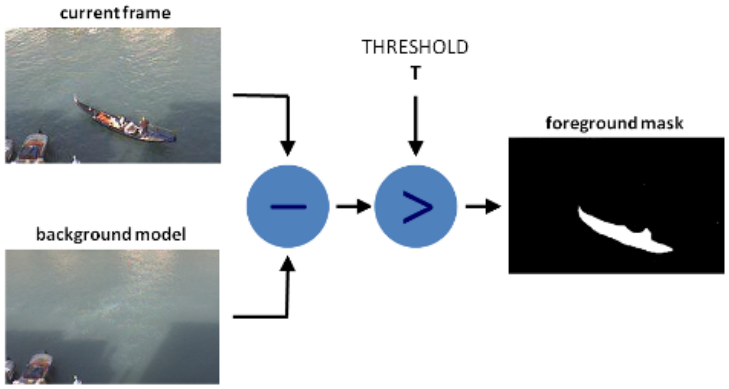

FIGURE 3 – Approche générique [3]

Pour obtenir un masque binaire du premier-plan, nous pouvons appliquer un seuil aux valeurs résultantes. Par exemple, si la valeur absolue d'un pixel après soustraction est supérieure à un certain seuil, nous le considérons comme faisant partie du premier-plan.

3. https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

Une deuxième approche

La première approche que nous avons vue repose sur une modélisation statistique simple de l'arrière-plan. Une approche souvent utilisée, notamment dans le cinéma, est d'utiliser un fond vert (green screen).

? Pourquoi le vert est-il souvent utilisé comme couleur de fond ?

L'idée derrière cette approche est de filmer les objets d'intérêt devant un fond d'une couleur uniforme afin d'extraire facilement le premier-plan en utilisant des techniques de « chroma keying ».

Les suppositions

Les suppositions seront un peu différentes de la première approche. En effet, nous n'avons plus besoin d'une caméra fixe ni d'une vidéo, une seule image suffit. En revanche, nous supposons que le fond est d'une couleur uniforme et distincte des couleurs des objets d'intérêt.

Extraction du premier-plan par chroma keying

Le chroma keying est une technique qui consiste à remplacer une couleur spécifique dans une image par une autre image ou vidéo.

Nous avons vu que les images sont représentées dans l'espace de couleur RGB (Rouge, Vert, Bleu), pour le chroma keying, il est souvent plus pratique d'utiliser l'espace de couleur HSV [4] (Hue, Saturation, Value).

Pour chaque pixel de l'image, nous allons vérifier si sa couleur correspond à la couleur de fond (par exemple, le vert dans le cas d'un fond vert). Si c'est le cas, nous remplaçons ce pixel par un pixel de l'image à **incruster**.

Comment caractériser une couleur ?

Pour rappel, l'espace de couleur HSV est une représentation alternative des couleurs qui peut être plus adaptée pour certaines applications, comme ici.

HSV signifie :

- **Hue** (Teinte) : Représente la couleur
- **Saturation** (Saturation) : Représente l'intensité de la couleur
- **Value** (Valeur) : Représente la luminosité de la couleur

Problème trivial ?

Bien que l'idée de base soit extrémement simple, en pratique, il peut être difficile de caractériser précisément une couleur.

Dans cette partie, vous êtes invités à prendre note des résultats afin de comparer vos résultats avec ceux de vos voisin·e·s afin de discuter des différences potentielles.

A vous !

1. Prenez de quoi noter vos résultats

4. La documentation d'OpenCV expliquant comment changer d'espace de couleur

2. Allez sur https://colorpicker.me/ et définissez l'intervalle des valeurs **H**ue, **S**aturation et **V**alue qui correspondent à des couleurs que vous considérez comme vertes.
3. Allez sur https://ismy.blue/ et indiquez, pour chaque couleur affichée, si vous la considérez comme bleue ou verte. A la fin de l'expérience, notez la valeur de **H**ue correspondant à votre limite entre bleu et vert ainsi que le pourcentage
4. Comparez vos résultats avec ceux de vos voisin·e·s

Il est intéressant de noter que beaucoup de choses peuvent influencer la perception d'une couleur, notamment l'éclairage, les écrans sur laquelle elle est affichée mais également notre propre perception. De plus, culturellement certaines couleurs sont définies différemment.

La librairie OpenCV

La librairie `OpenCV` (**Open** Source **C**omputer **V**ision Library) est une bibliothèque open-source de traitement d'images et de vision par ordinateur très populaire, de nombreux algorithmes issus de la recherche y sont implémentés.

Installation

Pour installer la librairie `OpenCV` pour Python, vous pouvez utiliser le gestionnaire de paquets `pip`.

Ouvrez votre terminal ou invite de commande et exécutez la commande suivante :

```
pip install opencv-python
```

Des différences majeures entre OpenCV et les autres librairies

Chaque librairie de traitement d'images a ses propres conventions et particularités. Nous avions déjà vu les différences entres `PIL` (Pillow) et `NumPy` pour le système de coordonnées.

Avec `OpenCV`, il y a une différence majeure à prendre en compte concernant l'ordre des canaux de couleur. Alors que la plupart des librairies utilisent l'ordre RGB (Rouge, Vert, Bleu) pour représenter les couleurs, `OpenCV` utilise l'ordre BGR (Bleu, Vert, Rouge). Cela signifie que lorsque vous chargez une image avec `OpenCV`, les canaux de couleur sont inversés par rapport à ce que vous pourriez attendre si vous êtes habitué à d'autres librairies.

Accès aux flux de la caméra d'un ordinateur

Le code ci-dessous est à votre disposition sur Moodle :

```python
import cv2

video = cv2.VideoCapture(0) # 0 pour la première webcam connectée, 1 pour la deuxième, etc.
while True:
    ret, frame = video.read()
    if not ret:
        break
    cv2.imshow('Frame', frame)
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
```

Extraire les images d'une vidéo

Le code ci-dessous est à votre disposition sur Moodle :

```python
import cv2
import os
# Chargement de la vidéo
video = cv2.VideoCapture('cars.mp4')
# Choisir d'extraire les images ou non
extract_images = True
compteur = 0
# Création du dossier pour stocker les images extraites s'il n'existe pas
if extract_images:
    output_folder = 'images'
```

```
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
while True:
    ret, frame = video.read()
    if not ret:
        break
    cv2.imshow('Frame', frame)
    if extract_images:
        # Sauvegarde de l'image extraite
        cv2.imwrite(os.path.join(output_folder, f'frame_{compteur:04d}.jpg'), frame)
        compteur += 1
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
```

Charger les images d'un dossier

Le code ci-dessous est à votre disposition sur Moodle :

```
import cv2
import os
folder = 'images'
for image in sorted(os.listdir(folder)): # Suppose que les images sont nommées de manière à être triées correctement
    image_path = os.path.join(folder, image)
    frame = cv2.imread(image_path)
    if frame is None:
        continue
    cv2.imshow('Frame', frame)
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
```

Changer d'espace de couleur

Le code ci-dessous est à votre disposition sur Moodle :

```
import cv2

bgr_image = cv2.imread('people.png')
hsv_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2HSV)
```

Tester l'intervalle et utiliser un masque

Le code ci-dessous est à votre disposition sur Moodle :

```
import cv2
import numpy as np

x_min = 100
y_min = 50
z_min = 50

x_max = 200
y_max = 150
z_max = 150

seuil_bas = (x_min, y_min, z_min)
```

```
seuil_haut = (x_max, y_max, z_max)

image = cv2.imread('gradient2.png')
mask = cv2.inRange(image, seuil_bas, seuil_haut)

# Mettre à noir les pixels en dehors de la plage de seuils
image[np.where(mask == 0)] = [0, 0, 0]

cv2.imshow('image', image)
cv2.waitKey(0)
```