Résumé OpenCV

Utilisation OPENCV:

OpenCV (Open Source Computer Vision) est une bibliothèque open source de vision par ordinateur et de traitement d'images. Elle a été initialement développée par Intel en 1999 pour fonctionner sous Windows. Depuis, elle a été portée sur de nombreuses plateformes, notamment Linux, Mac OS et Android. OpenCV est écrite en C++ mais dispose également de bindings pour Python.

Lire une image avec OpenCV

Pour lire une image avec OpenCV, on peut utiliser la fonction cv2.imread(). Cette fonction prend en paramètre le chemin vers le fichier image et le mode de lecture (cv2.IMREAD_COLOR pour une image couleur, cv2.IMREAD_GRAYSCALE pour une image en niveaux de gris ou cv2.IMREAD_UNCHANGED pour lire l'image telle quelle avec tous les canaux).

```
import cv2

# Charger une image en couleur
img_color = cv2.imread('image.jpg', cv2.IMREAD_COLOR)

# Charger une image en niveaux de gris
img_gray = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE).
```

Lire une vidéo avec OpenCV

Pour lire une vidéo avec OpenCV, on peut utiliser la classe cv2.VideoCapture(). Cette classe prend en paramètre le chemin vers le fichier vidéo ou l'identifiant de la caméra (0 pour la caméra par défaut, 1 pour la seconde caméra, etc.). Ensuite, on peut utiliser la méthode read() pour lire les images de la vidéo une par une.

```python
import cv2

# Ouvrir une vidéo
cap = cv2.VideoCapture('video.mp4')

# Boucle de lecture des images de la vidéo
while cap.isOpened():
    ret, frame = cap.read()

    # Vérifier si la lecture a réussi
    if ret:
        # Traiter l'image ici

        # Afficher l'image
        cv2.imshow('frame', frame)

        # Attendre l'appui sur une touche et quitter si c'est la touche 'q'
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    else:
        break

# Libérer les ressources
cap.release()
cv2.destroyAllWindows()
```

Détection d'objets avec OpenCV

OpenCV propose plusieurs algorithmes de détection d'objets, tels que la détection de visages, la détection de personnes ou la détection de voitures. Ces algorithmes sont basés sur des classifieurs en cascade, qui sont des ensembles de caractéristiques qui permettent de distinguer les objets d'intérêt des autres.

Pour utiliser un algorithme de détection d'objets avec OpenCV, on peut utiliser la classe cv2.CascadeClassifier(). Cette classe prend en paramètre le chemin vers le fichier XML qui contient les informations sur le classifieur en cascade. Ensuite, on peut utiliser la méthode detectMultiScale() pour détecter les objets dans une image. Cette méthode prend en paramètre l'image à analyser, le facteur de réduction de l'image (plus le facteur est petit, plus la détection sera précise mais plus le temps de calcul

sera élevé) et le nombre minimal de voisins pour chaque rectangle détecté (plus ce nombre est élevé, moins il y aura de faux positifs mais plus il y aura de faux négatifs).

```python
import cv2
# Charger la cascade de visage pré-entrainée
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Charger l'image à analyser
img = cv2.imread('test.jpg')

# Convertir l'image en niveaux de gris
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Détecter les visages dans l'image
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# Dessiner des rectangles autour des visages détectés
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Afficher l'image avec les rectangles autour des visages détectés
cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Cet exemple utilise la cascade de visage pré-entrainée incluse dans OpenCV (haarcascade_frontalface_default.xml) pour détecter les visages dans une image. La fonction detectMultiScale() est utilisée pour détecter les visages à différentes échelles dans l'image en utilisant un facteur d'échelle et un nombre minimum de voisins. Ensuite, des rectangles sont dessinés autour des visages détectés.

Vous trouverez tout les exemples ici :

https://github.com/khalillakhdhar/opencv

Manipulation des images:

La détection des contours

La détection des contours est une technique importante en traitement d'images qui permet de trouver les limites entre les différentes régions d'une image. La détection de contours peut être utilisée pour de nombreuses applications telles que la détection d'objets, la reconnaissance de formes et la segmentation d'images.

OpenCV propose plusieurs algorithmes pour la détection de contours, le plus commun étant l'algorithme de Canny. Voici un exemple de code pour détecter les contours d'une image en utilisant l'algorithme de Canny :

```
import cv2
import numpy as np

# Chargement de l'image
img = cv2.imread('image.jpg')

# Conversion de l'image en niveau de gris
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Détection des contours avec l'algorithme de Canny
edges = cv2.Canny(gray, 100, 200)

# Affichage des contours détectés
cv2.imshow('Contours', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

La conversion d'images en niveaux de gris

La conversion d'images en niveaux de gris est une opération courante en traitement d'images, qui consiste à transformer une image couleur en une image en noir et blanc. La conversion en niveaux de gris peut être utile pour simplifier le traitement d'une image et réduire la quantité de données à traiter.

Voici un exemple de code pour convertir une image en niveaux de gris en utilisant OpenCV :

```python
import cv2

# Chargement de l'image
img = cv2.imread('image.jpg')

# Conversion de l'image en niveau de gris
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Affichage de l'image en niveau de gris
cv2.imshow('Image en niveau de gris', gray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

La soustraction de fond

La soustraction de fond est une technique de traitement d'images qui permet de soustraire le fond statique d'une image pour ne garder que les objets en mouvement. Cette technique est souvent utilisée pour la détection de mouvement dans une vidéo.

OpenCV propose une fonction pour la soustraction de fond, appelée cv2.createBackgroundSubtractorMOG2(). Voici un exemple de code pour soustraire le fond d'une vidéo :

```python
import cv2

# Initialisation de la soustraction de fond
backgroundSubtractor = cv2.createBackgroundSubtractorMOG2()

# Chargement de la vidéo
cap = cv2.VideoCapture('video.mp4')

while True:
    # Lecture d'une image de la vidéo
    ret, frame = cap.read()

    if ret:
        # Soustraction du fond de l'image
        fgMask = backgroundSubtractor.apply(frame)

        # Affichage de l'image soustraite
        cv2.imshow('Image soustraite', fgMask)

        # Sortie si la touche 'q' est pressée
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
```

```
        break

# Libération des ressources
cap.release()
cv2.destroyAllWindows()
```

La détection de mouvement :

Pour la détection de mouvement, il faut d'abord utiliser la soustraction de fond pour obtenir un masque de premier plan. Ensuite, il est possible d'utiliser les fonctions de détection de contours d'OpenCV pour trouver les contours des objets en mouvement.

Voici un exemple de code pour détecter les contours des objets en mouvement :

```
import cv2

cap = cv2.VideoCapture(0)

background_subtractor = cv2.createBackgroundSubtractorMOG2()

while True:
    ret, frame = cap.read()

    foreground_mask = background_subtractor.apply(frame)

    contours, hierarchy = cv2.findContours(foreground_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    cv2.drawContours(frame, contours, -1, (0, 255, 0), 2)

    cv2.imshow('Contours', frame)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Ce code utilise BackgroundSubtractorMOG2 pour soustraire le fond de chaque image et détecter les mouvements. Ensuite, il utilise la fonction cv2.findContours() pour détecter les contours des objets en mouvement. Les contours sont ensuite dessinés sur l'image originale à l'aide de la fonction cv2.drawContours().

Amélioration de qualité:

Introduction

L'amélioration de la qualité des images est un domaine important dans le traitement d'images. Elle consiste à améliorer la qualité visuelle des images en supprimant le bruit, en augmentant la netteté, en ajustant la luminosité, etc. Dans ce cours, nous allons explorer quelques techniques d'amélioration de la qualité des images à l'aide de la bibliothèque OpenCV.

Conversion en niveaux de gris

La conversion d'une image en niveaux de gris est la première étape pour améliorer la qualité d'une image. Elle permet de convertir une image en couleur en une image en niveaux de gris, ce qui facilite le traitement ultérieur. Pour cela, nous allons utiliser la fonction cvtColor de la bibliothèque OpenCV :

```
import cv2

# Charger l'image en couleur
image = cv2.imread("image.jpg")

# Convertir l'image en niveaux de gris
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Suppression de bruit

Le bruit dans les images est généralement causé par des facteurs tels que l'éclairage insuffisant, la compression d'image ou le matériel de capture. La suppression du bruit peut améliorer la qualité visuelle de l'image. Il existe plusieurs techniques pour supprimer le bruit dans une image, mais dans ce cours, nous allons nous concentrer sur la suppression de bruit gaussien avec la fonction GaussianBlur :

```
import cv2

# Charger l'image en niveaux de gris
gray_image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)

# Supprimer le bruit gaussien avec un noyau de 5x5
blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)
```

Soustraction des fond:

La soustraction de fond (ou background subtraction en anglais) est une technique de traitement d'image qui consiste à séparer l'arrière-plan d'une scène de ses objets en mouvement. Cette technique est souvent utilisée dans les systèmes de vidéosurveillance pour détecter les objets en mouvement et les suivre.

Le principe de la soustraction de fond est de prendre une image de référence (l'arrière-plan) et de soustraire chaque nouvelle image de la vidéo à cette image de référence pour ne garder que les différences (les objets en mouvement). Ces différences peuvent ensuite être utilisées pour détecter les objets en mouvement.

OpenCV dispose de plusieurs fonctions pour effectuer la soustraction de fond, dont la fonction createBackgroundSubtractorMOG2. Cette fonction crée un modèle de mélange de gaussiennes (MOG2) pour l'arrière-plan et effectue la soustraction de fond en utilisant ce modèle.

Voici un exemple de code qui utilise la fonction createBackgroundSubtractorMOG2 pour effectuer la soustraction de fond :

```python
import cv2

# Charger la vidéo
cap = cv2.VideoCapture("video.mp4")

# Créer un objet BackgroundSubtractorMOG2
fgbg = cv2.createBackgroundSubtractorMOG2()

while True:
    # Lire une frame de la vidéo
    ret, frame = cap.read()

    # Appliquer la soustraction de fond
    fgmask = fgbg.apply(frame)

    # Afficher le résultat
    cv2.imshow('Soustraction de fond', fgmask)

    # Attendre une touche pour quitter
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Libérer les ressources
cap.release()
cv2.destroyAllWindows()
```

La fonction absdiff d'OpenCV est utilisée pour calculer la différence absolue entre deux images. Elle prend deux images en entrée et renvoie une image de différence absolue.

Voici un exemple simple d'utilisation de la fonction absdiff

```python
import cv2

# Ouvrir la vidéo
video_capture = cv2.VideoCapture("video.mp4")

# Récupérer la première image pour l'utiliser comme fond
_, background = video_capture.read()
background = cv2.cvtColor(background, cv2.COLOR_BGR2GRAY)
background = cv2.GaussianBlur(background, (21, 21), 0)

while True:
    # Récupérer une image de la vidéo
    _, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    # Calculer la différence entre l'image actuelle et l'image de fond
    diff = cv2.absdiff(background, gray)
    threshold = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)[1]
    threshold = cv2.dilate(threshold, None, iterations=2)

    # Trouver les contours des objets mouvants
    contours, _ = cv2.findContours(threshold.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Dessiner les contours sur l'image originale
    for contour in contours:
        if cv2.contourArea(contour) < 5000:
            continue
        (x, y, w, h) = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Afficher l'image résultante
    cv2.imshow("Video", frame)

    # Quitter le programme si on appuie sur la touche 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Libérer les ressources
video_capture.release()
cv2.destroyAllWindows()
```

Ce code utilise la fonction absdiff pour calculer la différence entre l'image actuelle et l'image de fond, puis utilise la fonction findContours pour détecter les contours des objets en mouvement. Les contours sont ensuite dessinés sur l'image originale à l'aide de la fonction rectangle.

Binarisation et dessin des contours

drawContours et threshold d'OpenCV :

La fonction drawContours() permet de dessiner les contours trouvés sur une image. Elle prend en entrée les points qui définissent les contours, l'index du contour à dessiner et la couleur et l'épaisseur de la ligne à dessiner.

La fonction threshold(), quant à elle, est utilisée pour binariser une image. Elle convertit une image en niveaux de gris en une image binaire en appliquant un seuil à chaque pixel. Les pixels qui dépassent le seuil sont définis comme blancs, tandis que ceux qui sont en dessous sont définis comme noirs.

Voici un exemple d'utilisation de ces deux fonctions :

```
import cv2

# Charger les deux images
img1 = cv2.imread('image1.jpg')
img2 = cv2.imread('image2.jpg')

# Convertir les images en niveaux de gris
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# Calculer la différence absolue
diff = cv2.absdiff(gray1, gray2)

# Afficher l'image de différence
cv2.imshow('Différence absolue', diff)
cv2.waitKey(0)
cv2.destroyAllWindows()
import cv2
import numpy as np

# Charger l'image
img = cv2.imread('image.jpg')

# Convertir l'image en niveaux de gris
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
# Appliquer un seuillage pour binariser l'image
ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Trouver les contours dans l'image binarisée
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

# Dessiner les contours sur l'image d'origine
cv2.drawContours(img, contours, -1, (0, 255, 0), 3)

# Afficher l'image avec les contours dessinés
cv2.imshow('Contours', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ici, nous avons chargé une image, converti en niveaux de gris, appliqué un seuillage pour binariser l'image, puis trouvé les contours dans l'image binarisée. Enfin, nous avons dessiné les contours sur l'image d'origine et affiché le résultat final.

Notez que nous avons utilisé la fonction cv2.RETR_TREE pour récupérer tous les contours et leurs hiérarchies, et la méthode cv2.CHAIN_APPROX_SIMPLE pour simplifier les contours en ne gardant que leurs points extrêmes.

Détection automatique des contours :

La détection des contours est l'un des aspects les plus fondamentaux du traitement d'image. Les contours peuvent être utilisés pour identifier des objets dans une image, pour suivre des objets en mouvement ou pour segmenter une image. OpenCV fournit plusieurs fonctions pour détecter les contours dans une image, la plus utilisée étant la fonction findContours.

La fonction findContours prend une image en entrée et renvoie une liste de contours. Les contours sont représentés par des points qui définissent les limites d'un objet dans l'image. La fonction findContours prend également en compte les paramètres tels que la résolution de l'image, la méthode de détection des contours et le type de chaîne de contour.

Voici un exemple de code pour détecter les contours dans une image :

```
import cv2

# Charger l'image en niveaux de gris
image = cv2.imread("image.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Appliquer un filtre de seuillage pour obtenir une image binaire
_, thresh = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)

# Trouver les contours dans l'image
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

# Dessiner les contours sur l'image originale
cv2.drawContours(image, contours, -1, (0, 255, 0), 2)

# Afficher l'image avec les contours dessinés
cv2.imshow("Contours", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exercices:

1. Écrivez un programme qui convertit une image en couleur en une image en niveaux de gris à l'aide de la fonction cvtColor d'OpenCV en supprimant la bruit.
2. Ecrivez un programme qui détecte les différences entre 2 photos légèrement différente de votre choix et qui affiche la photo de différences mse (supprimer la bruit avant)
3. Améliorez votre code Exercice 2 par une soustraction de fond
4. Écrivez un programme qui détecte les contours dans une image à l'aide de la fonction findContours d'OpenCV. Vous pouvez ensuite dessiner les contours détectés sur l'image d'origine à l'aide de la fonction drawContours.
5. Détecter les visage à partir d'une vidéo : exemple

https://www.pexels.com/video/video-of-woman-being-examined-8090200/