Miniprojet1_Junhui LUO

December 14, 2020

Étape 1 : Détection d'un seul visage et tracking post détection Feedback de course: cela entraînera la perte de la cible lors d'un déplacement rapide. Cela peut être dû à une faible configuration de l'ordinateur. Le programme ne peut pas re-suivre après avoir perdu la cible et ne peut que redémarrer le programme.

Importer CV2 et le remplacer par cv sera plus pratique.

[1]:
```
#detect+tracing
import cv2 as cv
```

Le classificateur est intégré dans opencv et peut être appelé directement à l'aide de la fonction cv.data.haarcascades pour éviter de référencer la totalité de l'emplacement cible.

[2]:
```
face_cascade = cv.CascadeClassifier(cv.data.haarcascades +␣
↪"haarcascade_frontalface_default.xml")
```

Initialisez d'abord le tracker. Ici, nous avons adopté TrackerMOSSE. Le tracker est rapide, mais pas aussi précis que CSRT et KCF. C'est suffisant pour notre miniprojet.

[3]:
```
def initialize():
    tracker = cv.TrackerMOSSE_create()
    return tracker
```

La logique de cette étape est: 1. Lisez la caméra. 2. Détecte le fonctionnement de l'appareil photo et, s'il est normal, effectue la reconnaissance faciale. 3. Loop le tracker pour suivre le premier visage qui a été lu. 4. Si on entre q, le processus se termine. (La signification de chaque fonction et algorithme spécifique est expliquée après chaque fonction.)

[7]:
```
def main():
    tracker = initialize()
    #Allume l'appareil photo. Les pixels de la caméra avant étant trop faibles,
    #utilisez ici (1) pour allumer la caméra arrière.
    cap = cv.VideoCapture(1)

    #Démarre le premier loop, la valeur par défaut est true.
    #Entre dans le loop de detect et démarre la détection de visage.
    Detect = True
    while Detect:
        #Lise la caméra.
        ret, frame = cap.read()
```

```
        #ret signifie que la caméra peut être lue
        if ret:
            #Effectuer un traitement en niveaux de gris sur l'image, frame␣
↪représente chaque image lue.
            #color_bgr2gray représente bgr en niveaux de gris.
            #Le traitement Grayscale permet d'accélérer la détection des␣
↪visages et de réduire le décalage.
            #Il est généralement utilisé dans la détection de visage
            gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
            #Perform face detection, 1.05 Ce paramètre représente un multiple␣
↪de l'échelle de détection.
            #3 signifie que 3 visages sont détectés dans une plage et peuvent␣
↪être identifiés comme des visages.
            #Plus le paramètre est élevée et plus la difficulté de détection␣
↪est élevée.
            faces = face_cascade.detectMultiScale(gray, 1.05, 3)

            #len (), le nombre de faces peut être obtenu grâce à cette fonction.
↪
            #En d'autres termes, lorsque la première face est lue, l'étape␣
↪suivante peut être effectuée.
            if(len(faces)>0):
                #x, y est le point du coin supérieur gauche et w, h est la␣
↪longueur du cadre.
                for x,y,w,h in faces:
                    # 0 signifie le premier visage
                    x,y,w,h = faces[0]
                    #Initialisez le tracker, récupérez frame de la␣
↪caméra(premier frame de face)
                    #et utilisez (x, y, w, h) comme référence.
                    tracker.init(frame, (x,y,w,h))
                    #Après la lecture, utilise detect = false pour quitter la␣
↪boucle.
                    Detect = False
        #Si la lecture de la caméra échoue, quittez et indiquez la raison.
        else:
            print("cannot grab frame, streaming end ?")
            break

    #Second loop: mettre à jour le tracker et augmenter la trame.
    while True:
        ret, frame = cap.read()
        #Donner au ROI la gamme de visages mise à jour.
        ret, roi = tracker.update(frame)
        if ret:
            #Lisez les données de plage de visage (x, y, w, h) en ROI.
```

```
            #int signifie que le résultat est affiché sous forme d'entier.
            x,y,w,h = [int(i) for i in roi]
            #Dessiner un cadre de visage: cadre (image), coordonnées en haut à␣
↪gauche,
            #coordonnées en bas à droite, couleur du cadre RBG, largeur du␣
↪cadre.
            cv.rectangle(frame, (x,y), (x+w, y+h), (0, 0, 255), 2)
            #print(x,y,w,h) dans cette étape, on peut utiliser print pour␣
↪vérifier l'état d'exécution du programme.
            #Il est également plus facile de trouver des problèmes dans le␣
↪programme.

        #Afficher les résultats de la caméra et du suivi du visage dans la␣
↪fenêtre.
        cv.imshow("result", frame)
        #Monitor les actions du clavier toutes les 1 millisecondes. Si vous␣
↪entrez q, vous quittez la boucle.
        if cv.waitKey(1) == ord('q'):
            break

    #Fermer l'appareil photo et fermer toutes les fenêtres.
    cap.release()
    cv.destroyAllWindows()
```

Exécuter le main() défini

[8]:
```
main()
```

```
192 286 72 72
192 287 72 72
192 287 72 72
192 286 72 72
192 286 72 72
191 285 72 72
191 285 72 72
191 285 72 72
191 285 72 72
191 285 72 72
192 285 72 72
192 286 72 72
192 286 72 72
192 286 72 72
192 286 72 72
191 287 72 72
191 287 72 72
192 287 72 72
192 287 72 72
192 287 72 72
```