Projet Python

Objectif : Réalisation d'un code en python pour contrôler les lecteurs multimédia grâce à diffèrent geste de la main.

Explication du code source

La réalisation du code source consiste en 10 étapes que nous allons tenter d'expliquer dans les lignes qui suivent :

---------- ----------

Étape - 1 - Importer des bibliothèques et capturer une caméra

Étape - 2 -Convertir les images en hsv

Étape - 3 - Suivre la main sur la base de la couleur

Étape - 4 -Créer un masque sur la base de la couleur et filtrer la couleur réelle

Étape - 5 - Inversez la valeur du pixel, puis améliorez le résultat pour une meilleure sortie

Étape - 6 - Trouver les contours d'un objet coloré spécifique

Étape - 7 - Trouvez le contour de la zone maximale et dessinez-le sur le flux en direct

Étape - 8 - Trouver la détection de convexité pour compter les valeurs et appliquer la méthode Cosin

Étape - 9 - Liez les gestes de la main avec les touches du clavier.

Étape - 10 – Affichage des fenêtres de sortie

---------- ----------

TANNOUCHE Chaimaa

ESSAOURI Hafsa

SIDIBE Mamadou

Étape - 1 - Importer des bibliothèques et capturer une caméra :

Nous aurons besoin pour la réalisation de ce projet de la bibliothèque OpenCV, pyautogui, numpy et math :

-La bibliothèque **NumPy** permet d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres.

-OpenCV (pour Open Computer Vision) est une **bibliothèque** graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

-**PyAutoGUI** est un module Python d'automatisation d'interface graphique multiplateforme pour les êtres humains. Utilisé pour contrôler par programme la souris et le clavier, il permet à vos scripts Python de contrôler la souris et le clavier pour automatiser les interactions avec d'autres applications.

-**Math** est une bibliothèque python nous permettant d'avoir accès a plusieur fonction mathématique.

```python
# Step -1
import cv2
import numpy as np
import math
import pyautogui as p
import time as t
```

Étape - 2 -Convertir les images en hsv

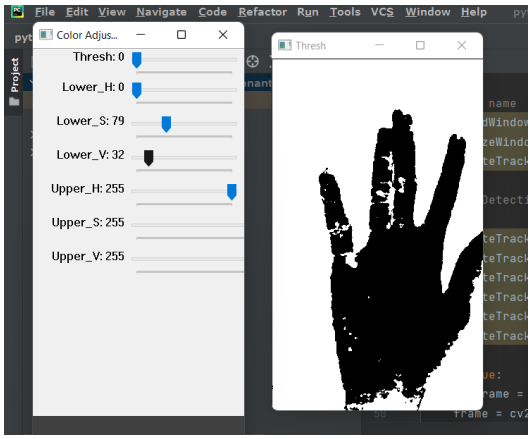
Le modèle TSV pour Teinte Saturation Valeur (en anglais HSV pour Hue Saturation Value ou HSB pour Hue Saturation Brightness), est un système de gestion des couleurs en informatique.

Nous allons procéder a sa configuration et lui attribuer des valeur par défauts :

Ajustement des paramètres po

```python
# window name
cv2.namedWindow("Color Adjustments", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Color Adjustments", (300, 300))
cv2.createTrackbar("Thresh", "Color Adjustments", 0, 255, nothing)

# COlor Detection Track

cv2.createTrackbar("Lower_H", "Color Adjustments", 0, 255, nothing)
cv2.createTrackbar("Lower_S", "Color Adjustments", 0, 255, nothing)
cv2.createTrackbar("Lower_V", "Color Adjustments", 0, 255, nothing)
cv2.createTrackbar("Upper_H", "Color Adjustments", 255, 255, nothing)
cv2.createTrackbar("Upper_S", "Color Adjustments", 255, 255, nothing)
cv2.createTrackbar("Upper_V", "Color Adjustments", 255, 255, nothing)

while True:
    _, frame = cap.read()
    frame = cv2.flip(frame, 2)
    frame = cv2.resize(frame, (600, 500))
    # Get hand data from the rectangle sub window
    cv2.rectangle(frame, (0, 1), (300, 500), (255, 0, 0), 0)
    crop_image = frame[1:500, 0:300]
```

Ajustement des paramètres pour la gestion des couleurs :



Étape – 2/3 -Détection de la main et Suivre la main sur la base de la couleur :

```
# Step -2
hsv = cv2.cvtColor(crop_image, cv2.COLOR_BGR2HSV)
# detecting hand
l_h = cv2.getTrackbarPos("Lower_H", "Color Adjustments")
l_s = cv2.getTrackbarPos("Lower_S", "Color Adjustments")
l_v = cv2.getTrackbarPos("Lower_V", "Color Adjustments")

u_h = cv2.getTrackbarPos("Upper_H", "Color Adjustments")
u_s = cv2.getTrackbarPos("Upper_S", "Color Adjustments")
u_v = cv2.getTrackbarPos("Upper_V", "Color Adjustments")
# Step -3 - Follow hand with color
lower_bound = np.array([l_h, l_s, l_v])
upper_bound = np.array([u_h, u_s, u_v])
```

Étape - 4 -Créer un masque sur la base de la couleur et filtrer la couleur réelle

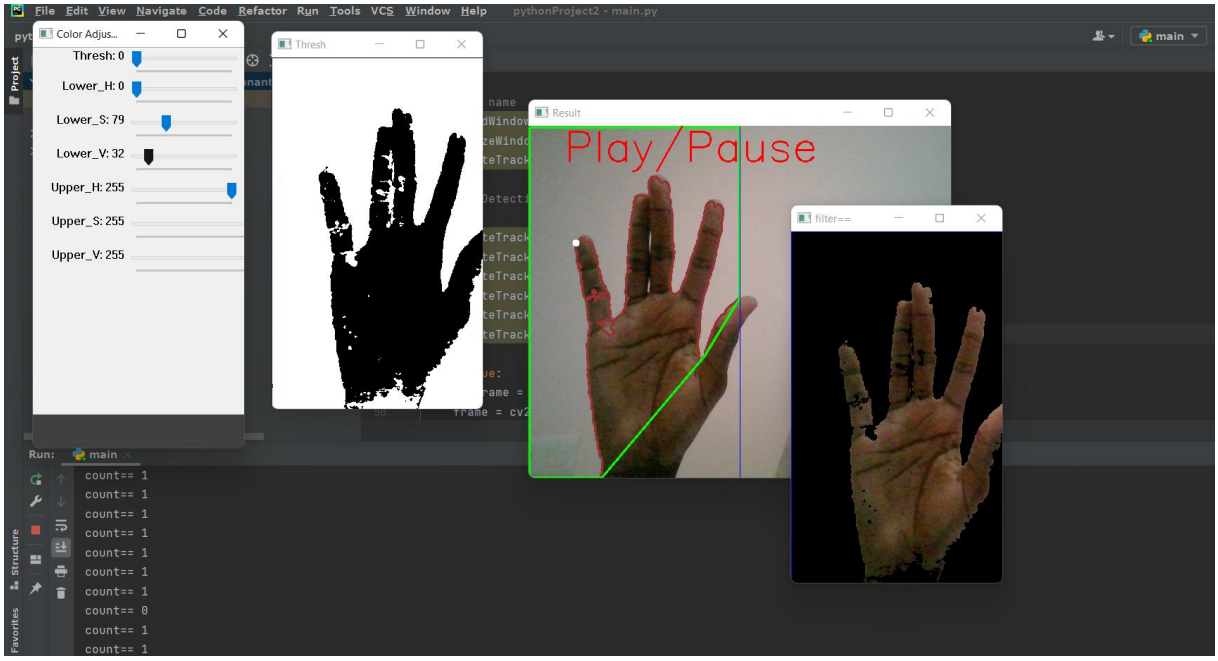
Filtrer la couleur avec OpenCV : La segmentation des couleurs ou le filtrage des couleurs est largement utilisé dans OpenCV pour identifier des objets/régions spécifiques ayant une couleur spécifique.

Pour identifier une région d'une couleur spécifique, nous plaçons le seuil et on crée un masque pour séparer les différentes couleurs.

L'espace colorimétrique HSV est beaucoup plus utile à cette fin car les couleurs dans l'espace HSV sont beaucoup plus localisées et peuvent donc être facilement séparées.

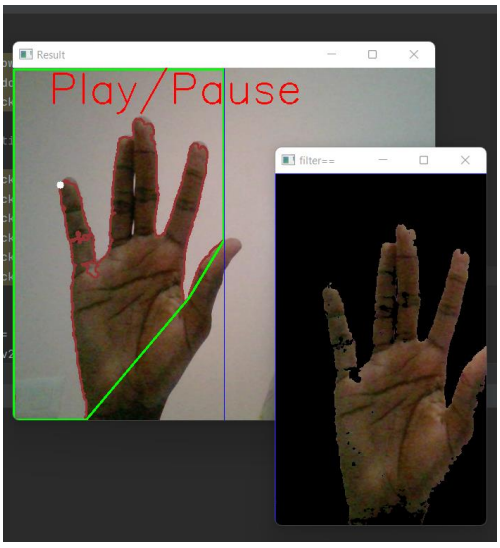
Le filtrage des couleurs a de nombreuses applications et utilise des cas tels que la cryptographie, l'analyse infrarouge, la conservation des aliments périssables, etc.

Dans de tels cas, les concepts de traitement d'images peuvent être utilisés pour découvrir ou extraire des régions d'une couleur particulière (dans notre cas ,extrait une main d'une region).




```
# Step - 4
# Creating Mask
mask = cv2.inRange(hsv, lower_bound, upper_bound)
# filter mask with image
filtr = cv2.bitwise_and(crop_image, crop_image, mask=mask)
```

Résultat du filtrage :



Étape - 5 - Inversez la valeur du pixel, puis améliorez le résultat pour une meilleure sortie


```
# Step - 5
mask1 = cv2.bitwise_not(mask)
m_g = cv2.getTrackbarPos("Thresh", "Color Adjustments")  # obtenir la valeur de la barre de suivi
ret, thresh = cv2.threshold(mask1, m_g, 255, cv2.THRESH_BINARY)
dilata = cv2.dilate(thresh, (3, 3), iterations=6)
```

Étape - 6 - Trouver les contours d'un objet coloré spécifique(main)

Représente les contours en Rouge.

Étape - 7 - Trouvez le contour de la zone maximale et dessinez-le sur le flux en direct

Représente les contours en Vert.


```
# Step -6
# findcontour(img,contour_retrival_mode,method)
cnts, hier = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

```python
# Step -7
# Find contour with maximum area
cm = max(cnts, key=lambda x: cv2.contourArea(x))
# print("C==",cnts)
epsilon = 0.0005 * cv2.arcLength(cm, True)
data = cv2.approxPolyDP(cm, epsilon, True)

hull = cv2.convexHull(cm)

cv2.drawContours(crop_image, [cm], -1, (50, 50, 150), 2)
cv2.drawContours(crop_image, [hull], -1, (0, 255, 0), 2)
```

Étape - 8 - Trouver la détection de convexité pour compter les valeurs et appliquer la méthode Cosin

Analyse d'image et calcul des angle grâce au quel nous allons attribue différentes valeurs aux différentes positions de la main (angle des doigts) a laide notamment des point blanc placé comme repère sur la main

Et count nous retourne en tant réel la valeur déduit par l'angle de la main chaque signe (positionnement des doigts) de la main correspondante a un chiffre diffèrent (dans notre exemple nous avons pris 5 signes diffèrent)

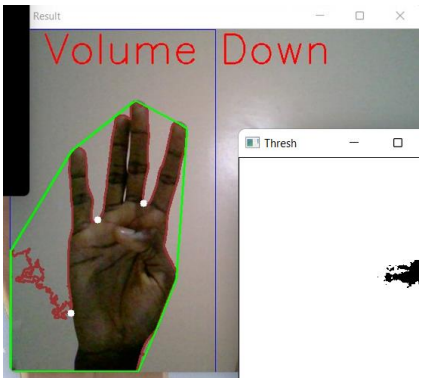
main

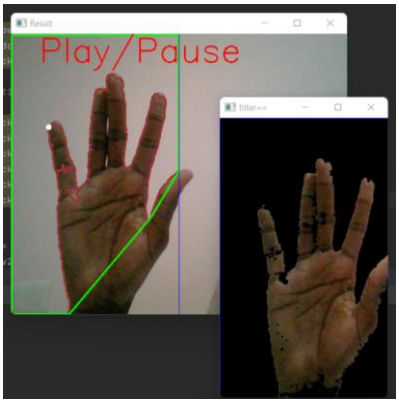
```
count== 3
count== 3
count== 3
count== 2
count== 3
count== 2
count== 2
count== 3
count== 2
count== 5
count== 3
count== 2
count== 3
count== 3
count== 2
count== 2
count== 3
count== 4
count== 0
count== 1
count== 0
count== 0
count== 0
count== 0
count== 0
count== 0
count== 1
count== 0
```

```python
# Step - 8
# Find convexity defects
hull = cv2.convexHull(cm, returnPoints=False)
defects = cv2.convexityDefects(cm, hull)
count_defects = 0
# print("Area==",cv2.contourArea(hull) - cv2.contourArea(cm))
for i in range(defects.shape[0]):
    s, e, f, d = defects[i, 0]

    start = tuple(cm[s][0])
    end = tuple(cm[e][0])
    far = tuple(cm[f][0])
    # Cosin Rule
    a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)
    b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
    c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
    angle = (math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) * 180) / 3.14
    # print(angle)
    # if angle <= 50 draw a circle at the far point
    if angle <= 50:
        count_defects += 1
        cv2.circle(crop_image, far, 5, [255, 255, 255], -1)

print("count==", count_defects)
```

Étape - 9 - Liez les gestes de la main avec les touches du clavier.

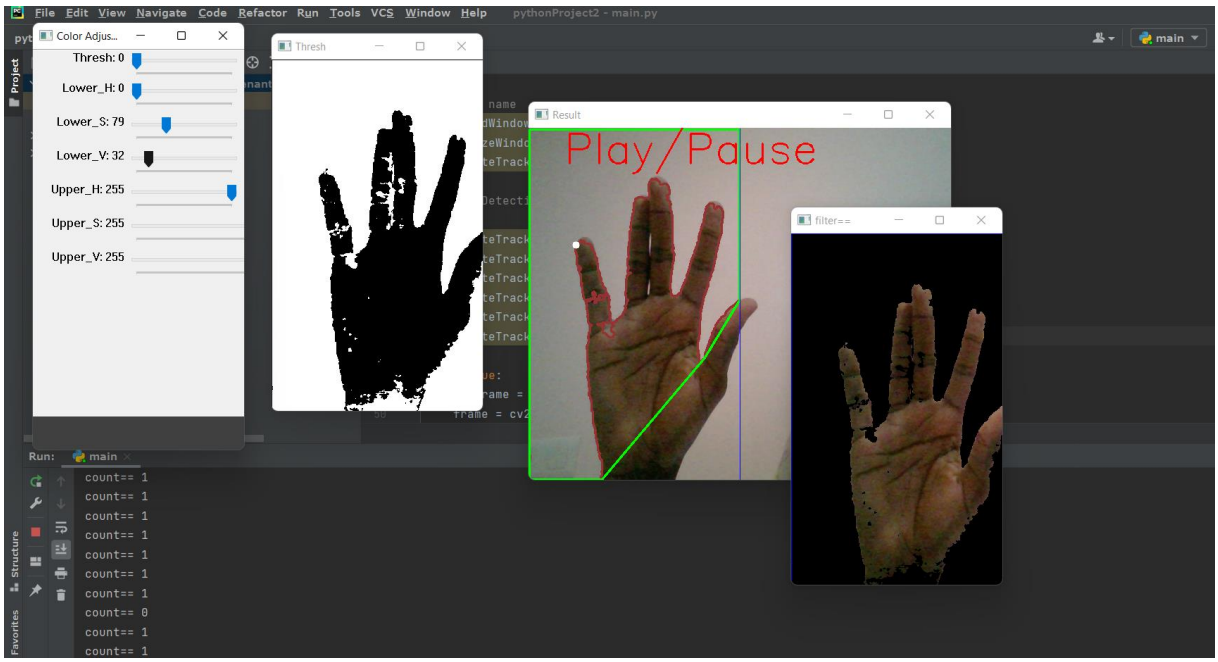
Assignation de chaque geste une touche du clavier pour permettre lorsque qu'on se trouve dans un lecteur multimédia d'effectuer différentes actions






```python
# Step - 9
# Print number of fingers
if count_defects == 0:

    cv2.putText(frame, " ", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2)
elif count_defects == 1:

    p.press("space")
    cv2.putText(frame, "Play/Pause", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2)
elif count_defects == 2:
    p.press("up")

    cv2.putText(frame, "Volume UP", (5, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2)
elif count_defects == 3:
    p.press("down")

    cv2.putText(frame, "Volume Down", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2)
elif count_defects == 4:
    p.press("right")

    cv2.putText(frame, "Forward", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2)
else:
    pass
```

Étape - 10 – Affichage des fenêtres de sortie




```
    # step -10
    cv2.imshow("Thresh", thresh)
    # cv2.imshow("mask==",mask)
    cv2.imshow("filter==", filtr)
    cv2.imshow("Result", frame)

    key = cv2.waitKey(25) & 0xFF
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

FI N