LoveFinderrz

DIVERSITY by EPITECH

I. Introduction

Salut à toi jeune entrepreneur !

Alors si aujourd'hui je me permets de te contacter, c'est pour une raison très simple, il faut que tu te poses les bonnes questions : Est-ce que tu préfères passer ta Saint-Valentin tout seul en boule dans ton coin, ou commencer très rapidement une méthode pour trouver l'amour ?

Moi je pense la question elle est vite répondue, alors soit tu me suis, sois tu sors du camp.

Aujourd'hui avec les applications de rencontres, on se fait embobiner facilement, c'est pourquoi TU vas pouvoir être le maître de ton destin et trouver l'amour en développant toi-même une IA qui trouveras ton âme sœur !

"Déjà vu, I've just been int this place before, higher on the street, and I know it's my time to go. Calling you and search is a mystery, Standing on my feet, it's so hard when I try to be me, woaaaaaaaah, Déjà vu"

II. Consignes

- Vous allez devoir compléter des fonctions sous Python avec la libraire OpenCV
- Si vous avez des questions, demandez à votre voisin de gauche, puis de droite. Coopérez !
- En cas de blocage, et après avoir tenté la consigne précédente, n'hésitez pas à demander à un Cobra, ils sont très gentils et sont là pour vous aider.
- Vous avez entièrement le droit d'utiliser internet comme vous le souhaitez pour vous renseigner, nous le conseillons fortement.
- Si vous n'avez jamais fait de Python ou de programmation auparavant et que vous souhaitez apprendre quelques notions de base avant de vous lancer : demandez à un Cobra de vous assister et rendez ici pour apprendre les bases.
- *N'hésitez pas à faire des bonus et ajouter/modifier des fonctionnalités à votre programme, la seule limite est votre imagination ! (Et 17h)*

- Amusez-vous ! Nous ne sommes pas là pour apprendre mais pour découvrir, vous avez le droit de ne pas réussir à compléter le sujet. 😊

III. Le commencement

Tous d'abord commencez par ouvrir Visual Code, puis ouvrez « IA.py ».

L'objectif de cette première partie est de récupérer le contenu de notre caméra via la librairie OpenCV, pour ensuite l'afficher sur notre écran.

Pour ce faire, nous allons suivre les étapes suivantes une par une :

(L'emplacement dans votre fichier ou il vous faudra réaliser ces étapes sont indiqués par des commentaires)

Tout d'abord, nous allons commencer par nous connecter à notre camera :

```
camera = Cv2.VideoCapture(0)
```

Désormais, l'on va pouvoir accéder à notre caméra grâce à la variable « camera » (nommage 100)

Ensuite, il va falloir lire l'image de notre caméra, bien évidement l'on traitera les images une par une, c'est pour cela que nous appelons cette fonction ci-dessous dans une boucle :

```
Ret, frame = camera.read()
```

Ici, notre variable « frame » contient l'image toute fraîche que notre caméra vient de lire.

Cette fonction est optionnelle, elle permet simplement retourner l'image afin d'avoir un effet miroir et de ne pas nous donner mal à la tête. (Essayez avec et sans pour voir la différence)

```
Frame = cv2.flip(frame, 1)
```

L'on finit par le plus important, afficher l'image de la caméra que l'on vient de lire :

```
cv.imshow("Nom_De_Votre_Fenetre", frame)
```

Désormais, notre programme est capable de se connecter à la caméra de votre, et une par une d'afficher chacune des images que la dites caméra lit !

IV. Les visages

Le retour de la caméra est apparu sur l'écran, super !

Cela dit, le chemin est encore long pour trouver l'âme sœur… Il va maintenant falloir détecter les visages que l'on voudra étudier par la suite.

Pour que notre IA puisse voir plus clairement les visages, l'on va appliquer un filtre sur notre image pour que celle-ci devienne grise :

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Ici l'on applique le filtre « cv2.COLOR_BGR2GRAY à notre image « frame », et l'on stocke le résultat de la modification dans une nouvelle variable « gray », car lorsque l'on voudra afficher le retour, l'on préféra garder l'image originelle pour avoir un retour en couleur 😉.

Maintenant l'on va récupérer la liste des visages grâce à une fonction déjà programmé dans la librairie OpenCV :

```
faces = face_cascade.detectMultiScale(gray)
```

Grâce à cette fonction, notre nouvelle variable faces contient une liste de contenant les cordonnées de chaque visage

Maintenant que nous avons la liste des visages détectés, l'on va essayer d'afficher des rectangles qui vont les encadrer pour vérifier que l'algorithme fonctionne bel et bien!

L'on commence par récupérer les coordonnées de chacun des visages dans une boucle:

```
for (x, y, width, height) in faces:
```

Si vous n'êtes pas familier avec la notion « for _ in _», n'hésitez pas à demander à un Cobra de vous expliquer ou à vous rendre sur ce lien.

Ensuite, on initialise la couleur de notre rectangle dans une variable sous le format (BLEU, VERT, ROUGE) :

```
color = (255, 0, 0)
```

Puis, l'on finit par dessiner le rectangle sur notre image « frame », en lui donnant les coordonnées du visage, pour le coin en haut à gauche, puis en lui donnant ces mêmes coordonnées auquels l'on a ajouté la taille du visage détecté pour le coin en bas à droite :

```
cv2.rectangle(frame, (x, y), (x + width, y height), color)
```

On dessine le ou les rectangles avec tous nos paramètres afin qu'ils soient aux bons endroits.

V. L'Ajout

Notre IA est capable de détecter un visage sur notre caméra, c'est un immense pas en avant !

Mais elle ne sait pas encore si ce visage correspond à votre âme sœur ou à un/e énième friendzone, pour cela l'on va devoir l'instruire.

Pour ce faire, il va d'abord falloir lui fournir des échantillons :

Récoltez sur internet des images (PNG et JPG) de célébrités populaires et impopulaires afin de construire votre base de données, et mettez celles qui vous plaisent dans le dossier « SoulMate » et celle qui vous plaisent moins dans le dossier « FriendZone ».

Maintenant que vous possédez une base de données assez fournie, il va falloir que notre IA analyse ces données afin de les reconnaître plus tard.

Commençons par changer de fichier, dirigez-vous dans le fichier « Learning.py »

Nous allons à nouveau devoir récupérer la liste des visages de chaque image ouverte comme réalisé précédemment :

```
faces = face_cascade.detectMultiScale(image_array)
```

Ne vous inquiétez pas, les Cobras se sont déjà occupés de coder l'algorithme qui ouvre les images une par une, cela prendrait beaucoup trop de temps, et nous vous aimons trop pour vous infliger cela. <3

Encore une fois, nous allons récupérer chacune des coordonnées des visages détectés dans chaque image :

```
for(x, y, w, h) in faces:
```

Et grâce à ces coordonnées, nous allons rajouter à une liste la partie de l'image en cours d'analyse la contenant uniquement le visage, ainsi que le « label » à une autre liste, nous indiquant à quelle catégorie appartient le visage analysé (FriendZone ou SoulMate) :

```
face_list.append(image_array[y:y+h, x:x+w])
label_list.append(current_id)
```

Si vous souhaitez comprendre « image_array[y :y+h , x :x+w], n'hésitez pas à demander à un Cobra de vous expliquer ou rendez-vous sur ce lien.

Désormais, il ne nous reste plus qu'à rajouter ces deux lignes de codes à la fin du fichier afin d'utiliser un algorithme déjà codé dans la libraire OpenCV pour entraîner notre IA :

```
recognizer.train(faces_images, np.array(images_labels))
recognizer.save ("Utils/recognizers/face-trainner.yml")
```

La fonction « train » sert à entrainer notre IA et « save » à sauvegarder ce qu'elle a appris, ou sa « mémoire » dans un fichier.

VI. L'Accomplissement

Maintenant que notre IA a en mémoire un échantillon de vos goûts, il va falloir lui laisser faire sa prédiction et s'exprimer.

Retournons à nouveau dans le fichier « IA.py » pour le grand final.

L'on commence par une dernière fois récupérer les coordonnées de chacun des visages que notre caméra voit :

```
for(x, y, w, h) in faces:
```

Ensuite grâce à coordonnées, l'on récupère la partie de l'image qui contient le visage :

```
face_gray = gray[y:y+h, x:x+w]
```

L'on utilise ici la version avec le filtre gris de l'image car c'est plus facile pour l'IA de faire ses comparaisons

Et enfin l'on demande à notre IA de donner son verdict, et de nous indiquer à quelle catégorie appartient le visage analysé :

```
labels_predicted, confiance = recognizer.predict(face_gray)
category = labels[labels_predicted]
```

Lors de la prédiction l'on se sert uniquement de la variable « labels_predicted », « confiance » permet juste à savoir à quel point l'IA est sûre d'elle dans sa prédiction.

Maintenant que l'on sait à quelle catégorie appartient la personne, il va falloir l'afficher en dessous de son visage, l'on commence par initialiser dans l'ordre la police, la couleur, et l'épaisseur du texte que l'on s'apprête à afficher :

```
font = cv2.FONT_HERSHEY_SIMPLEX
color = (255, 255, 255)
stroke = 2
```

Il ne reste plus qu'à écrire le texte en question sur l'image de la caméra avec les paramètres précédemment initialisés :

```
cv2.putText(frame, category, (x,y), font, 1, color, stroke,
cv2.LINE_AA)
```

Si vous voulez une explication sur les paramètres que nous n'avons pas mentionné précédemment, n'hésitez pas à demander à un Cobra.

VII. The End ?

Bravo !

Vous possédez une IA capable de reconnaître votre âme sœur (bien qu'elle ne soit basée que sur l'apparence).

Vous pouvez être fier de vous et aller vous vanter devant vos amis d'avoir non seulement trouvé un/e compagne/ons pour la Saint-Valentin, mais surtout d'avoir su réaliser une IA par vous-même !