Accès sécurisé par reconnaissance faciale par CNN

HAI927I - Projet Image - CR4

NGUYEN Thi-Christine - RICHARD Pierre

24/10/2024

Sommaire

1 Introduction

Pour cette semaine nous nous sommes penchés que la recherche de data pour notre base de données puis on a commencé à chercher comment détecter les visages dans une image.

2 La base de données

Nous avons commencé à récupérer des images de célébrité et avons ajouté des images personnelles (personnes de la classe, famille...). Cette base de données sera complétée au fur et à mesure.

Emma Watson

Daniel Radcliffe

Rupert Grint

Bonnie Wright

Figure 1: Exemple d'images dans notre base de données

3 Détection de visage

Pour la détection de visage, nous avons utilisé l'algorithme d'Haar Cascade. Cet algorithme est présent dans openCV, cependant, nous avons dû faire plusieurs traitements pour pouvoir l'utiliser.
En effet, pour utiliser cet algorithme par openCV, il nous a fallu convertir nos images en niveaux de gris. Nous avons donc utilisé une fonction openCV permettant de transformer nos images en image en niveau de gris avant d'appliquer l'algorithme d'Haar.

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

Après cette conversion, nous faisons appel à la Haar Cascade. Nous avons d'abord initialisé le classifieur de Haar Cascade qui est un modèle pré-entraîné pour détecter les visages. Le fichier xml est fourni par openCV.

```
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
```

Nous sommes ensuite passés à la détection de visage. Nous avons utilisé la méthode `detectMultiScale`, qui renvoie une liste de rectangles contenant les coordonnées des visages détectés.

$$\text{faces} = [[x_1, y_1, w_1, h_1], [x_2, y_2, w_2, h_2]]$$

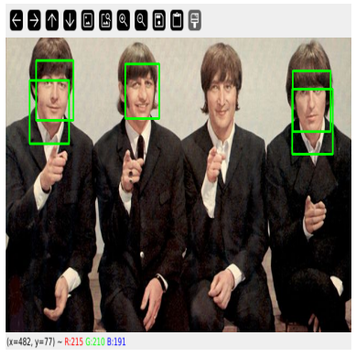

Figure 2: Première version de détection

Nous pouvons voir deux soucis, tout d'abord tous les visages ne sont pas reconnus. Pour cela, nous avons augmenté la précision en modifiant les paramètres de la fonction `detectMultiScale` mais cela peut ajouter des artéfacts. La fonction `detectMultiScale` peut prendre en paramètre un nombre minimum de voisin et un scale factor.

- Le nombre de voisin : indique combien de voisin doivent être présents pour q'un rectangle soit retenu. Si plusieurs détections se chevauchent et que le nombre de voisin est supérieur à ce seuil alors les détections en double seront fusionnées.
- scale factor : Donne la taille de l'échelle et de l'image

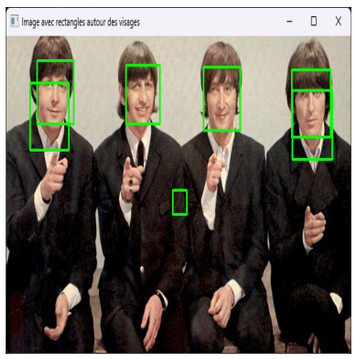

Figure 3: Détection de tous les visages

Ensuite, nous pouvons remarquer que plusieurs rectangles représentant le même visage étaient retournés. En effet, la méthode d'Haar décale itérativement sa zone d'analyse. Nous avons donc récupéré ces informations et avons fusionné les rectangles représentant le même visage.

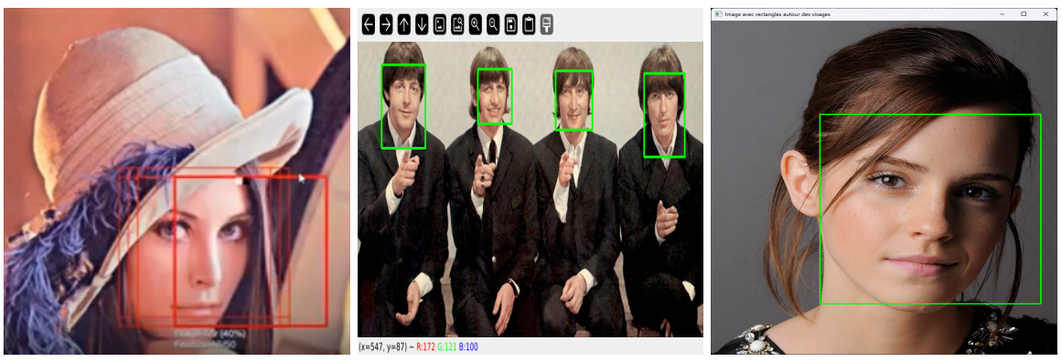

Figure 4: Fusion des rectangles

4 Portage mobile

Nous avons trouvé une méthode afin de porter l'application python sur android à l'aide de deux outils, buildozer (le compilateur) et Kivy (la mise en forme).

Cependant, nous rencontrons quelques problèmes concernant la caméra. Ainsi, nous essayons d'avoir une caméra dans le bon sens. Ensuite, nous verrons comment faire le lien entre les méthodes mises en place et l'application.

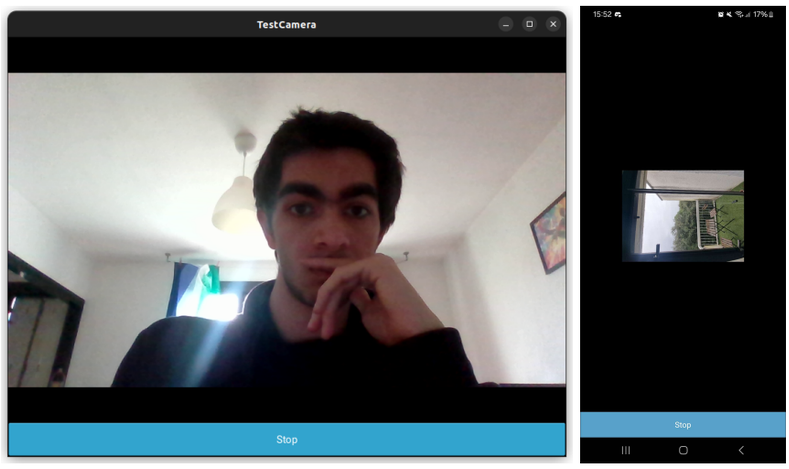

Figure 5: Application Kivy

5 Conlusion

Cette semaine, nous avons donc réussi à récupérer les visages présents dans une image, sous forme de tableau de rectangle. De plus, nous avons trouvé un moyen de porter l'application sur un téléphone.
Pour la semaine prochaine, nous voulons avoir une méthode IA mise en place ainsi qu'une méthode sans IA mise en place.
Si nous avons le temps, nous aimerons avancer sur le portage sur Android.