Accès sécurisé par reconnaissance faciale par CNN

HAI927I - Projet Image - CR5

NGUYEN Thi-Christine - RICHARD Pierre

17/11/2024

Sommaire

1 Introduction

Cette semaine, nous avons mis en place une méthode de reconnaissance faciale classique sans réseau de neurones.
Nous avons également eu la base de données des membres de la classe et nous les avons traitées.

2 LBPH

La méthode de LBPH ou Local Binary Pattern Histogram est une méthode qui consiste à seuiller les valeurs au alentour du pixel. Ces voisins seuillés permettront de définir la valeur binaire du pixel. Ainsi, grâce à cette nouvelle valeur, on sait comment varie la lumière par rapport à ses voisins.

Ensuite, on divise l'image en plusieurs blocks et on en sort des histogrammes. Nous les concaténons ensuite pour avoir un grand histogramme et ce sont ces éléments que nous comparons.

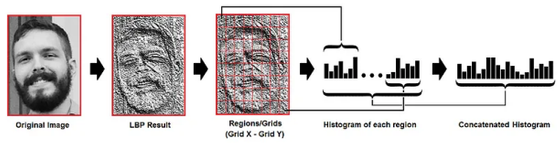

Figure 1: LBPH

Nous avons pu mettre en place cette méthode et nous l'avons testé sur la base de données ORL, car bien que nous ayons une base de données avec plusieurs membres du master, nous n'avions pas encore le prétraitement nécessaire lorsque nous avons développé cette méthode.

Figure 2: ORL Database

Ainsi nous avons les résultats suivants pour la base de données ORL qui possède 40 classes de 10 images:

train data	train data per class	test data	right predictions	wrong prediction	right prediction percent
360	9	40	38	2	95
320	8	80	77	3	96
280	7	120	113	7	94
240	6	160	151	9	94
200	5	200	184	16	92
160	4	240	219	21	91
120	3	280	238	42	85
80	2	320	260	60	81
40	1	360	246	114	68

Table 1: LBPH with 8*8 blocks

On voit ici que plus, on entraîne l'algorithme, meilleur sont les résultats, mis à part pour la première ligne de notre tableau. Cela est le cas, car nous ne testons pas sur beaucoup de données, ainsi la grandeur des derniers résultats est moins fiable.

3 Comparatif avec VGGface

train data	train data per class	test data	LBPH right percent	VGGface right percent
360	9	40	95	95
320	8	80	96	90
280	7	120	94	90
240	6	160	94	92
200	5	200	92	88
160	4	240	91	82
120	3	280	85	80
80	2	320	81	78
40	1	360	68	61

Table 2: LBPH and VGGface comparaison

On remarque avec ces résultats que VGG face est moins efficace que LBPH. Ces résultats sont peut-être biaisé à cause de la base de données ORL. Il faudra recomparer les deux méthodes avec notre nouvelle base de données.

4 Base de données

Nous avons désormais la base de données récoltée par nos camarades Renault et Benjamin. Nous allons utiliser cette base de données pour entraîner notre modèle CNN. Cette base de données comprend environ 80 photos.

4.1 Découpage des images

Pour utiliser notre CNN (qui sera détaillée à la section suivante), nous avons besoin de données. Soit des images de taille 224 x 224 pixels représentant des visages de personnes. En premier temps comme expliqué à sur le rapport de la semaine précédente, nous avons utilisé Haar cascade. Malheureusement, nous nous sommes rendu compte du manque de précision sur la détection des visages. Nous avons donc décidé d'utiliser Dlib shape predictor.

4.1.1 Haar Cascade

En utilisant Haar cascade sur toutes les images récolté par nos camarades. Nous avons rapidement réalisé que si le visage des personnes n'était pas plus ou moins en face de la caméra, il était très difficile pour l'algorithme de reconnaître les visages.
En effet, on s'est retrouvé avec beaucoup d'artéfacts. Une solution aurait été de jouer avec les paramètres (tels que le nombre de voisin) mais cela reste trop spécifique à chaque image.

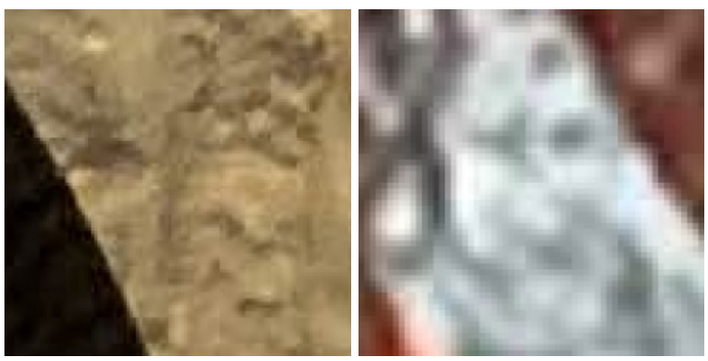

Figure 3: Mauvaises détection

Sur les 80 photos présentes dans la base de données, l'algorithme a réussi à détecter 57 "visages" dont 19 sont des artéfacts. Donner de telles images à notre CNN pour l'entraîner n'est pas pertinent. Nous somme donc passer sur la méthode Dlib shape predictor.

4.1.2 Dlib Shape Predictor

Dlib Shape Predictor est une méthode plus robuste pour détecter les visages, même lorsque ceux-ci ne sont pas parfaitement alignés. Contrairement à Haar Cascade, qui détecte simplement un rectangle englobant le visage, Dlib Shape Predictor utilise un réseau pour localiser les points clés du visage (landmarks). Ces points comprennent les yeux, le nez, la bouche, et les contours du visage.

1. **Détection des visages** : La détection initiale du visage est réalisée avec le détecteur de visages intégré de Dlib, qui utilise un algorithme basé sur des Histogrammes de Gradients Orientés (HOG).

2. **Extraction des points clés (landmarks)** : Une fois le visage détecté, le Shape Predictor extrait 68 points de repère du visage.

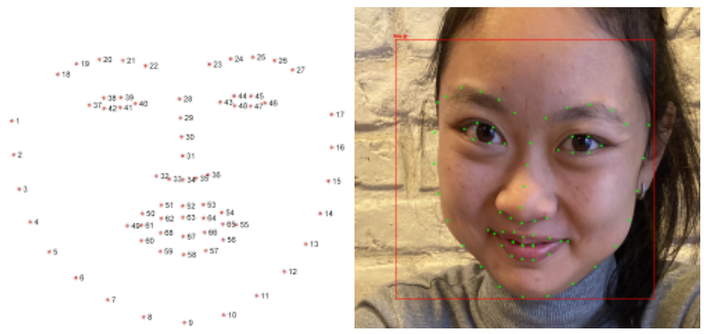

Figure 4: 68 landmarks

Grâce à l'utilisation de cette méthode, on a à présent 68 visages détectés sans aucun artéfact. On voit donc une différence notable sur la détection des visages. Cependant, on peut également remarquer que le temps de récupération et d'analyse des visages est un peu long avec Dlib Shape Predictor. Temps pour générer la base de données :

- Dlib Shape Predictor : 64.91 secondes.
- Haar Cascade : 42.18 secondes.

5 Le CNN

Après avoir récupéré un bon nombre d'images de taille 224 x 224, nous allons les utiliser pour entraîner notre modèle. Mais tout d'abord, nous avons construit notre modèle sous la forme de VGG. En effet, nous avons été contraints d'utiliser cette architecture car nous souhaitions utiliser VGG-Face (qui est un modèle pré-entrainé pour les poids).

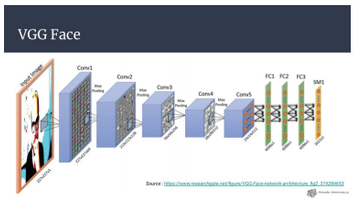

Figure 5: Schéma d'architecture de VGG Face

Pour notre CNN, nous avons créé une fonction permettant de créer un bloc de couches de convolution suivi d'une couche de MaxPooling (réduit la dimension spatiale des données).
Suite à cela nous avons suivi le modèle de VGG-Face (voir figure au-dessus). Nous avons donc 5 blocs convolutionnels, 3 couches fully-connected et la couche finale pour l'extraction et l'aplatissement de vecteur.

Nous allons envoyer dans notre CNN toutes nos images précédemment récoltée et allons stocker les vecteurs récupérés dans un tableau. Ce tableau de vecteur va nous permettre de savoir qui est la personne sur l'image.

Nom : Thi
Vecteur 1 : [2.4976823 2.5114067 1.4116911 ... 0. 3.9164333 2.8119905]
Vecteur 2 : [2.2570574 3.5280151 1.1167387 ... 0. 3.9918454 4.3108644]
Vecteur 3 : [2.9510415 0. 0. ... 0. 4.7658415 4.0085306]
Vecteur 4 : [3.208823 0.99648213 1.5239264 ... 0.07882105 1.9355844 0.03370737]

6 Trouver le vecteur le plus proche

Actuellement pour chercher le vecteur le plus proche entre l'image en entrée et les images dans la base de données, nous utilisons un calcul d'angle entre les vecteurs.
Nous avons des résultats plutôt satisfaisants mais pour plus de précision, nous testerons différentes manières (utilisation de SVM, distance...).

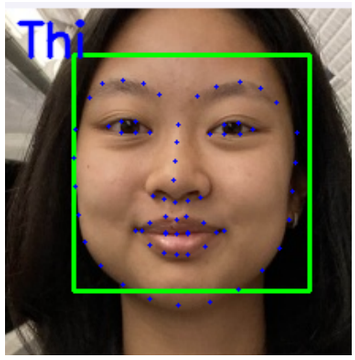

Figure 6: Reconnaissance

7 Ressource

Understanding Face Recognition Using LBPH Algorithm
Face Recognition: Understanding LBPH Algorithm
ORL dataset
HOG
Dlib Shape Predictor