Compte Rendu Semaine 4

Accès sécurisé par reconnaissance faciale

Josua Philippot - Félix Yriarte
Master 2 IMAGINE

Octobre 2021 - Décembre 2021

Table des matières

1 Contexte

Nous avons décidé d'axer notre travail sur trois grands points, avant de nous intéresser aux méthodes de reconnaissance faciale utilisant des réseaux de neurones convolutifs :

- L'amélioration de notre méthode par LBPH développée l'an dernier :
  - Meilleure méthode de comparaison d'histogrammes
  - Intégration MLBPH comme vu dans la littérature
  - Prétraitements appliqués sur nos images afin de réduire le fond, ou bruit
- La mise en place d'un système de reconnaissance faciale par eigenfaces

2 Amélioration LBPH : classificateur en cascades de Haar

Dans l'optique d'améliorer nos résultats, notamment lors de notre utilisation de LBPH, nous avons décidé d'utiliser une implémentation du classificateur en cascade de Haar.
Son principe est le suivant :

- On constitue une base de données comportant des images "positives" (contenant l'objet de la détection, dans notre cas des visages) et des "négatives" (sans visages).
- On extrait les caractéristiques pseudo-Haar[1]
- On entraîne le classificateur sur notre base de données.
- On détecte les objets désirés sur une ou plusieurs images.

Étant donné que l'utilisation de ce classificateur nécessite une grande base de donnée pour l'entraîner, et les connaissances suffisantes pour entraîner un réseau neuronal, nous avons décidé dans un premier temps de nous pencher sur les implémentations de Haar déjà existantes pour gagner du temps. Ainsi nous avons choisi d'utiliser celle proposée par la librairie OpenCV [2].
La librairie nous met à disposition ce classificateur qui utilise sur un réseau de neurones pré-entraîné sur une large base de visage, et qui est déjà capable de détecter les zones d'intérêt faciales très rapide. De plus, elle met également à disposition les méthodes d'entraînement des dits réseaux pour que l'utilisateur puisse entraîner lui-même son propre réseau.
Nous avons réussi à prendre le pendre en main et à l'utiliser pour obtenir la zone dans laquelle se trouve le visage dans une image que l'on passe en entrée du programme. Avec cette information, il nous sera possible à l'avenir de traiter les images uniquement dans les zones pertinentes, comme le visage uniquement, ou même récupérer seulement certaines zones d'intérêt plus restreintes comme les yeux, le nez et la bouche.

3 Eigenfaces

Dans la méthode par eigenfaces, on tente de récupérer les *composantes principales* d'un ensemble d'image d'entrainement. On cherche en fait à définir une base (dans l'espace des visages, ou *face space*) tel qu'il soit possible de représenter n'importe quelle image de visage comme une combinaison linéaire d'eigenfaces (les éléments de cette base, qui sont des images).

Pour ce faire, il est nécessaire de calculer les vecteurs propres de la matrice de covariance de notre base initiale d'images.

Pour la base AT&T (400 images niveaux de gris de visages, provenant de 40 personnes, en résolution 92x112), les eigenfaces obtenues sont présentées ici :

Elles sont ordonnées par valeur propre décroissante ; ce qui signifie que les eigenfaces présentées sont de *moins en moins représentatives de la base d'images*. Pour une base d'images d'entraînement de N images, on obtiendra *au plus* N vecteurs propres (et eigenfaces, donc).

Une fois cette base constituée, il est nécessaire de projeter les images de visages à analyser sur les différentes eigenfaces (de la même manière que l'on projeterait un vecteur sur un autre ; par produit scalaire !). On obtient donc un coefficient flottant, représentant la similarité entre le visage analysé et l'eigenface considérée. Après projection sur un certain nombre d'eigenfaces, on obtient donc un *code*, constitué de N flottants, qui représente un visage. Avec une bonne base d'entrainement, il est même possible de reconstruire une image de visage en n'utilisant que les eigenfaces et leur coefficient associé (tant qu'un assez grand nombre d'eigenfaces est considéré).

Pour *reconnaître un visage* par eigenfaces, une distance entre des combinaisons linéaires d'une même base d'eigenfaces est effectuée. Par exemple, si le code associé à un visage dans un face space $(Im_i)_{i\in[1,N]}$ donné est

$\alpha_1 Im_1 + \alpha_2 Im_2 + ... + \alpha_N Im_N = \sum \alpha_i Im_i$ et $\sum \alpha'_i Im_i$.

Une distance entre ces deux combinaisons linéaires est $\sum \alpha_i - \alpha'_i$ Il est donc possible de mesurer la similarité entre différentes images de visages, d'où l'interêt de la méthode pour de la reconnaissance faciale.

4 Perspectives

Nous pouvons donc avancer sur différents points :

— Mise en place de courbe ROC (expérimentations approfondies sur la base [3] intégrale) afin de trouver de bons paramètres (seuils) pour nos méthodes classiques

— Mise en place d'une base de eigenfaces et d'une distance entre combinaisons linéaires d'eigenfaces

— Se renseigner sur les CNNs pour la reconnaissance faciale et en débuter l'implémentation.

5 Nos sources

L'intégralité du travail que nous allons produire se trouve sur ce dépôt Github :
https://github.com/JPhilippot/FaceRecognition

Références

[1] Wikipedia. *Caractéristiques pseudo-Haar*. https://fr.wikipedia.org/wiki/Caract%C3%A9ristiques_pseudo-Haar, 2018.

[2] OpenCV. *Cascade Classifier*. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html, 2021.

[3] Kasikrit Damkliang. *AT&T Database of Faces*. kaggle.com https://www.kaggle.com/kasikrit/att-database-of-faces, 2021.