PROJET ING1
Reconnaissance faciale par Analyse en Composantes Principales

Clément Bertails
Fabien Darrigrand
Maxime Longuet
Grégoire Sibille
Russelle Silatcha

18 mars 2022

Table des matières

1 Le projet

En première année de cycle ingénieur, nous avons comme projet d'écrire un programme en java qui permet de faire de la reconnaissance faciale. Nous avons donc décidé d'utiliser une base de données qui contiendra un certain nombre d'images de plusieurs visages. Cette première base de données sera une base de référence. Nous aurons aussi la possibilité de l'agrémenter en envoyant au programme une image calibrée. De plus, nous utiliserons quelques images "test" afin de tester notre programme java.

La reconnaissance faciale en elle même est basée sur un vecteur caractéristique qui est propre à chaque visage. En effet, ces caractéristiques sont supposées invariantes pour un individu donné, mais différentes d'une personne à l'autre.

Ce projet est découpé en plusieurs parties. Premièrement nous allons créer une base de données normalisée contenant les images de référence du projet. Nous allons ensuite faire une Analyse en Composante Principale de chaque image et ainsi, nous en déduirons le vecteur caractéristique de chacune. Par la suite, via une projection dans le sous-espace des eigenfaces nous allons analyser à nouveau l'image et effectuer ainsi une identification.

Avant de réaliser le code source du projet nous devions effectuer une analyse en composantes principales et expliquer comment cela fonctionne. Nous devions aussi écrire un diagramme de modélisation du projet (UML) qui nous permettra d'avoir une représentation des différentes classes/objets nécessaires à la réalisation du projet, soit une réflexion sur la structure globale du programme.

2 La base de données (BDD)

2.1 Les photos normalisées

Nous avons choisi de prendre des photos avec d'autres groupes pour avoir une base commune. Ces photos sont en 1920x1080 pixels (redimensionnables plus tard). Elles sont essentiellement composées de tons de gris et sont orientées verticalement. Le fond des images est monotone afin d'éviter de perturber l'identification.

Nous avons décidé de prendre 4 photos par personne, 3 d'entre elles sont utilisées comme photos de références, et la quatrième est uniquement destinée aux tests de développement et d'application de notre programme.

Afin de faciliter la création de cette base de données, nous l'avons réalisée en collaboration avec d'autres groupes pour se la partager. Notre programme sera donc effectué avec des images de nos camarades.

2.2 Exemples

Voici quelques exemples de photos de référence :

Et ici quelques exemples de photos de test :

FIGURE 1 – Photos

3 L'Analyse en Composantes Principales (ACP)

3.1 Le concept de l'ACP

Revenons d'abord à notre source d'information : les images. Les photos qui composent notre base de données sont équivalentes à des matrices dont chaque composante est un vecteur composé d'un ton de gris. Une image de 1920*1080 pixels correspondrait donc à 2'073'600 points dont la couleur est définie, et avec cela, nous devons être capables de reconnaître la personne. Bien évidemment, ces valeurs sont trop grandes et nous adapterons les tailles après quelques tests, afin d'éviter des calculs interminable mais aussi en maximisant les informations fournies par chaque image. On doit donc utiliser une méthode pour déterminer lesquels sont les plus importants pour arriver à un résultat : c'est le rôle de l'ACP. L'ACP permet d'étudier les échantillons en regroupant ceux ayant des caractéristiques similaires, et c'est ce qu'on veut vu que l'on a accès à une grande quantité d'échantillons qui vont nous permettre de repérer des composantes principales intéressantes pour distinguer les visages.

3.2 La réduction de la dimension

Imaginons un espace de données de grande dimension et plein de points dans cet espace.

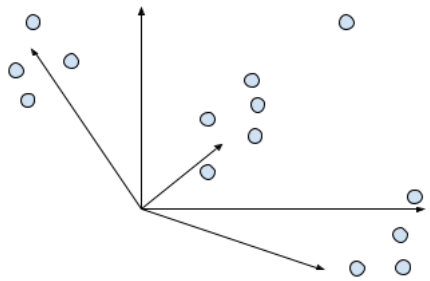


FIGURE 2 – Exemple d'un espace de dimension 5

L'ACP consiste à identifier les meilleurs façons de projeter ces valeurs sur un espace plus simple pour y voir les principales variations du nuage de points (notamment en ignorant les directions selon lesquelles les variations sont négligeables).

Ainsi, on peut passer d'un espace de grande dimension (D) à un hyperplan de dimension (d) beaucoup plus petit (D » d).

Le principe consiste en l'identification des directions de plus grande variations du nuage de points (les composantes principales) puis ensuite effectuer une projection orthogonale sur l'hyperplan engendré par ces composantes principales. Ainsi on garde seulement C composantes principales parmi toutes celles de base.

On a alors une nouvelle représentation des données qui est alors la réduction de la dimension.

3.3 L'ACP dans la reconnaissance faciale

L'utilisation de l'ACP en reconnaissance faciale sert à obtenir la variation d'une banque d'image afin d'apprendre à comparer des visages machinalement. Étant donné les nombreuses variables d'un visage (distance entre sourcils, longueur nez, etc.) il est nécessaire de capturer les composantes principales de la banque et de l'exprimer en vecteurs propres. La méthode utilisée lors de notre projet de reconnaissance faciale est la méthode eigenfaces.

3.4 La méthode eigenfaces

3.4.1 Images de références

Les images de références (qui servent pour l'apprentissage) sont représentées sous forme de matrices de dimensions l*h (largeur * hauteur de l'image en pixels) puis transformées en vecteurs de l*h lignes.
Ici, I_i représente la i-ème image de la banque de données (c'est une matrice de pixels).

$$I_i = \begin{vmatrix} a_{1,1} & a_{1,2} & . & . & . & a_{1,N} \\ a_{2,1} & a_{2,2} & . & . & . & a_{2,N} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ a_{N,1} & a_{N,2} & . & . & . & a_{N,N} \end{vmatrix}_{N \times N} \longrightarrow \begin{vmatrix} a_{1,1} \\ . \\ . \\ . \\ a_{1,N} \\ . \\ . \\ . \\ a_{N,N} \end{vmatrix}_{N^2 \times 1} = \Gamma_i \tag{1}$$

3.4.2 Calcul du visage moyen

On obtient le visage moyen à partir des N visages de référence.

$$\Psi = \frac{1}{N} \times \sum_{i=1}^{N} \Gamma_i \tag{2}$$

Le visage moyen peut être ainsi vu comme le centre de gravité de la banque d'image, ou encore le visage type, par "défaut". Tout écart à ce visage est unique et peut donc servir à identifier quelqu'un.

3.4.3 Recherche des visages propres

En soustrayant le visage moyen aux visages de référence on a le visage propre. Ce visage propre correspond aux caractéristiques qui différencient cette personne des autres et ne contient pas l'information moyenne contenue dans chaque visage (elle est inutile pour nous car commune à tout le monde).

$$\begin{array}{lcl} \Phi_i & = & \Gamma_i - \Psi \\ \Gamma_i & : & \text{vecteur image de référence} \\ \Phi_i & : & \text{vecteur des caractéristiques propres de l'image } \Gamma_i \end{array} \tag{3}$$

3.4.4 Calcul de la matrice de covariance

A ce niveau, la base des données est faite. Le rassemblement dans une matrice donne :

$$A = [\Phi_1 \Phi_2 ... \Phi_N] \tag{4}$$

On obtient maintenant la covariance en multipliant A par sa transposée

$$C = \sum_{i=1}^{N} \phi_i \phi_i^T = AA^T \tag{5}$$

On peut maintenant se rendre compte que même en prenant de petites images de taille N^2 (100x100 soit 10'000 pixels), on se retrouverait donc avec une matrice de dimension $N^2 \times N^2$. Il serait irraisonné de faire ces calculs même avec un ordinateur. Nous utiliserons donc une méthode alternative.

3.4.5 Calcul des vecteurs propres

Dans le cas où la résolution *(R)* est grandement supérieure au nombre d'images on peut simplifier les calculs en passant par une matrice L de taille R*R. Il est possible de retrouver les vecteurs propres de C en multipliant ceux de L par la matrice A.

Avec e_i les vecteurs propres de $C = AA^T$ associés aux valeurs propres λ_i, et v_i les vecteurs propres de $L = A^T A$ associés aux valeurs propres μ_i on a :

$$Ce_i = \lambda_i e_i \qquad Lv_i = \mu_i v_i \tag{6}$$

Soit

$$\begin{cases} e_i &= Av_i \\ \lambda_i &= \mu_i \end{cases} \tag{7}$$

On peut constater que les valeurs propres sont égales et que les vecteurs propres sont liés.

3.4.6 Calcul des poids associés à chaque visage propre

Ordonnance des vecteurs propres selon leurs valeurs correspondantes. Plus la valeur propre est élevée et plus les informations contenues dans le vecteur propre sont importantes.

$$\begin{aligned} \Phi_i &= \textstyle\sum_{j=1}^{K} w_j e_j \\ w_j &= e_j^T \Phi_i \\ e_j &: \text{Visage Propre (eigenfaces)} \end{aligned} \tag{8}$$

Dans cette étape, nous utilisons les vecteurs propres que nous avons obtenus à l'étape précédente. Nous prenons les visages d'entraînement normalisés (visage - visage moyen) Γ_i et représentons chaque vecteur de visage dans la combinaison linéaire des K meilleurs vecteurs propres.

En concaténant les vecteurs propres e_i on peut définir la matrice E. Ainsi, on peut construire la matrice G par :

$$G = E^T \times A \tag{9}$$

3.4.7 Identification d'une personne

L'identification d'une personne consiste à trouver l'image qui lui ressemble le plus parmi les images de référence. Pour cela il faut retirer à l'image à trouver les caractéristiques moyennes puis on la projette sur l'espace vectoriel relatif à la matrice de projection G afin de trouver la plus petite distance entre deux projections (car elles se ressemblent le plus). Le résultat de l'identification est donc la personne associée à l'image de la base de données qui ressemble le plus à l'image présentée au système.

Mesure de distance Calcul de la distance euclidienne entre la projection de l'image test et la projection. Pour qu'un visage test soit reconnu, il faut que la distance avec le plus proche voisin (ou les K plus proches) soit inférieure à un seuil donné.

Le but est de suivre les étapes effectuées précédemment, nous projetons le vecteur normalisé dans l'espace propre pour obtenir la combinaison linéaire d'eigenfaces.

$$\Phi = \sum_{j=1}^{K} w_j e_j \tag{10}$$

où

$$\Phi = \Gamma - \Psi \tag{11}$$

A partir de la projection ci-dessus, nous générons le vecteur :

$$\Omega = E^T \times \Phi \tag{12}$$

Puis on recherche la distance minimal entre Ω et les vecteurs G_i de G :

$$d_{min}(\Omega, G_i) = min \mid\mid \Omega - G_i \mid\mid \tag{13}$$

Distance de Minkowski Pour effectuer une mesure de similarité, il faut passer par les distances Euclidiennes définies par la distance de Minkowski d'ordre p dans un espace euclidien. Si l'on considère deux vecteurs V = (v1, v2, ..., vn) et W = (w1, w2, ...,wn), la distance de Minkowski d'ordre p noté Lp est définie par :

$$L_p = (\sum_{i=1}^{n}(|v_i - w_i|)^p)^{1/p} \tag{14}$$

Distance de Mahalanobis L'espace de Mahalanobis est un espace où la variance selon chaque dimension est égale à 1. Obtenu à partir des images Im en divisant chaque vecteur propre par son écart-type correspondant. Relations entre les vecteurs propres de l'image Im u et v avec deux vecteurs r et s :

$$r_i = \frac{u_i}{\sigma_i} = \frac{u_i}{\sqrt{\lambda_i}} \text{ et } s_i = \frac{v_i}{\sigma_i} = \frac{v_i}{\sqrt{\lambda_i}} \tag{15}$$

où λ_i les valeurs propres associées aux vecteurs u et v et σ_i l'écart-type.

La distance de Mahalanobis L2 (MahL2) est identique à la distance euclidienne mais représentée dans l'espace de Mahalanobis. Elle est définie par :

$$Mah_{L2}(u, v) = \sqrt{\sum_{i=1}^{N} |r_i - s_i|^2} \tag{16}$$

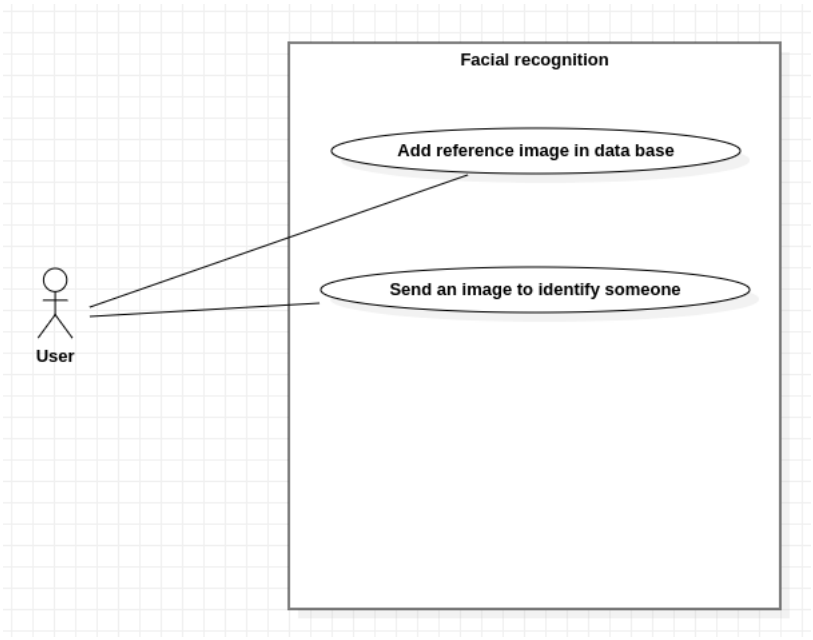
où u et v deux vecteurs propres de projections respectives r et s sur l'espace de Mahalanobis.

4 Diagramme de Modélisation (UML)

Pour la partie modélisation du projet, nous avons réaliser trois diagrammes différents.

4.1 Diagramme de cas d'utilisation

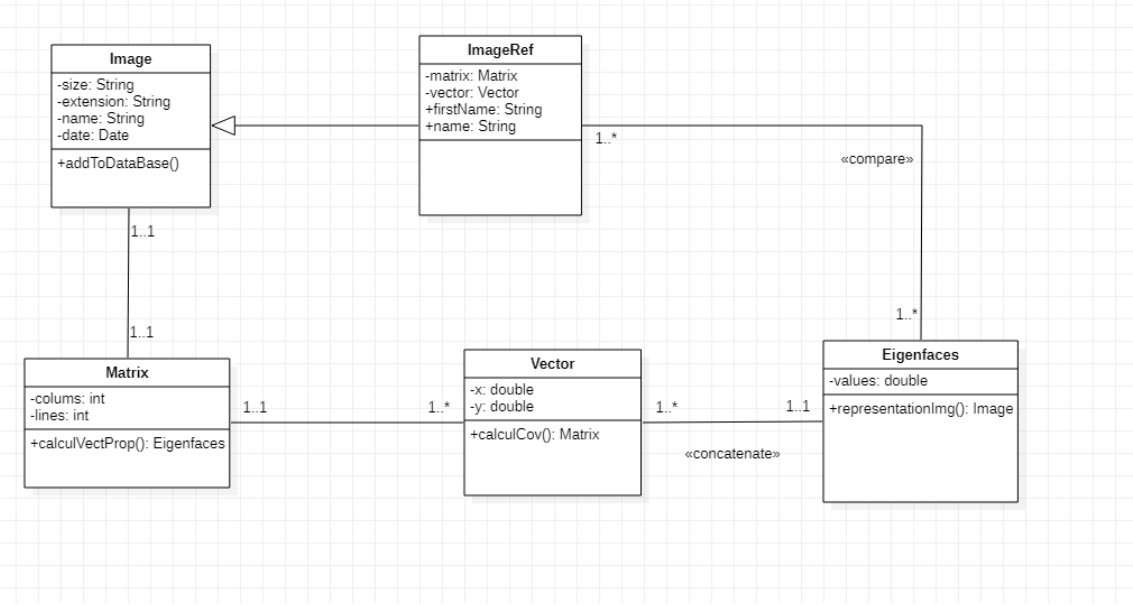
Premièrement nous avons effectué le diagramme de cas d'utilisation.



Le diagramme de cas d'utilisation est assez vite résumé. En effet, le programme aura deux fonctionnalités. La première est la possibilité d'ajouter une image de référence dans la base de données, tout en vérifiant qu'elle est bien normalisée. La seconde est d'appliquer la reconnaissance faciale sur une image test. L'utilisateur pourra choisir via une interface graphique l'option voulue.

4.2 Diagramme de classe

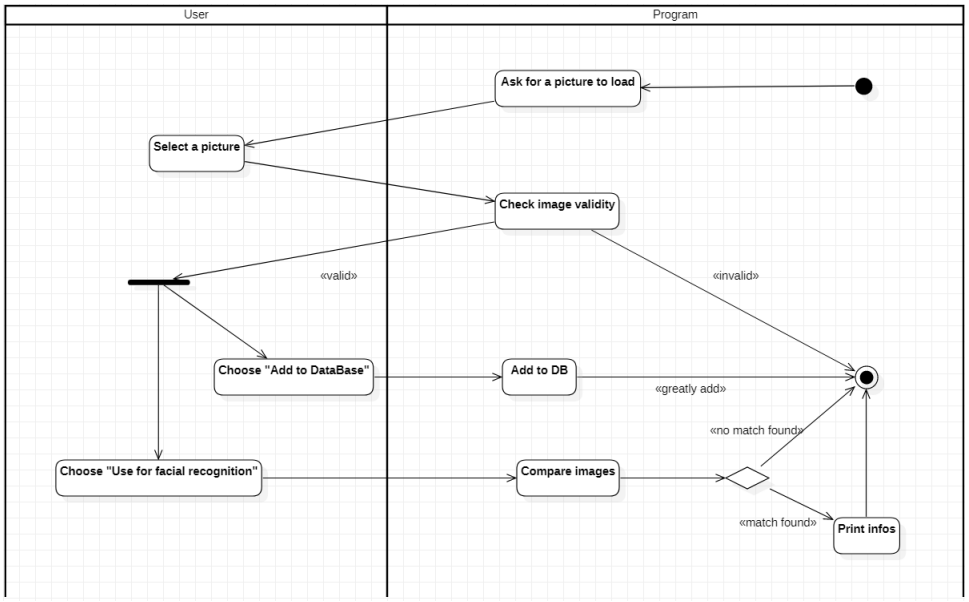
Ensuite, nous avons effectué le diagramme de classe du projet.



Dans notre diagramme de classe nous avons tout d'abord une classe mère Image qui contient tous les informations sur une image puis nous avons une classe fille ImageRef qui contient les informations traitées des images test.
Ensuite nous avons créé une classe Matrice, qui à partir des pixels de l'image qu'on veut comparer avec les ImageRef, doit pouvoir calculer la matrice de covariance.
Dès lors qu'on a obtenu la matrice de covariance de l'image, on obtient les vecteurs propres d'où la création de la classe Vecteur. Puis on peut déduire les valeurs propres.
Après avoir fait tout ce traitement il nous reste à comparer ces informations avec celles de ImageRef.

4.3 Diagramme d'activité

Pour finir, nous avons effectué le diagramme d'activité du projet.



Le diagramme d'activité est assez simple lui aussi, nous commençons donc dès que le programme est lancé. Ce dernier attend le chargement d'une image. Dès que l'utilisateur à importé l'image souhaitée, le programme demande parmi les fonctionnalités "Ajouter à la Base de Données" et "Appliquer la reconnaissances faciale". L'utilisateur choisit alors l'option voulue, enfin l'image est vérifiée. Si l'image n'est pas correctement normalisée, alors une erreur s'affiche et le programme se finit, sinon selon le choix de l'utilisateur l'ajout dans la base de données prend effet ou bien, dans le second cas, la reconnaissance faciale débute. Le programme se termine après avoir exécuté l'action demandée.

5 Nous et le projet

5.1 Répartitions des tâches

Bien que nous devons commencer à repartir les tâches, nous avons pensé important de rappeler que nous avons du mal à estimer le travail à effectuer et donc les durées associées. C'est la raison pour laquelle, au lieu d'attribuer un domaine du projet à chacun (comme BDD, ACP etc) nous avons préféré assigner des rôles un peu plus généraux. Ainsi Grégoire devra garder un oeil sur l'avancement du projet et des livrables et sera en charge de nous rappeler si on reste trop longtemps sur une partie. En même temps, Clément et Maxime se chargeront de tester régulièrement les fonctionnalités afin de détecter les erreurs, essayer de les comprendre et les corriger s'ils le peuvent. Fabien et Russelle se chargeront principalement d'implémenter les différentes fonctionnalités dans le code source du programme. Bien évidemment, tous sont en charge du développement et de l'avancée régulière du projet, y compris Grégoire, Clément et Maxime.

5.2 Avis

Nous avons abordé ce projet avec beaucoup d'intérêt. En effet, il s'agit là de notre premier projet en Java et le sujet abordé nous attire. Nous avons fait des réunions en présentiel, dans les locaux de l'école, ainsi que des réunions en distanciel, via Teams ou encore Discord. Nous avons décidé, dans un premier temps, de produire la base de données en commun avec plusieurs autres groupes de travail. Ensuite nous nous sommes réunis pour étudier l'Analyse en Composantes Principales appliquée à la reconnaissance faciale. Nous avons enfin réalisé les différents diagrammes présents ci-dessus.

Références

[1] *Face Recognition Using Eigenfaces (PCA Algorithm)* : article sur l'utilisation de l'ACP, GeekForGeeks,
21 septembre 2021

[2] *RECONNAISSANCE FACIALE PAR METHODE ACP HYBRIDE* : mémoire de fin d'études sur la reconnaissance faciale,
RANDRIAMAHANDRY Vonjinirina Eric
mai 2016

[3] *Reconnaissance Faciale par Eigenfaces* : article sur la reconnaissance faciale par ACP, Silanus.fr,
5 février 2017