Reconnaissance faciale par Analyse en Composantes Principales

Abade Maxime, Lasgleizes David, Legrand Amandine,
Morelli Candice, Ribas Justine

Projet de semestre 2 ING1

06/05/2022

ING1 - GI - Groupe 10

Professeurs encadrants : Nisrine Fortin et Muhammad Ehsan

Table des matières

1 Introduction

[3] À l'heure où la question de la surveillance mondiale est sur toutes les lèvres, nous sommes au cœur de ce qui anime de nombreux débats depuis des décennies. En effet, les gouvernements se sont dotés d'une surveillance mondiale. Leur cause ; la sécurité mondiale, principalement dans les grandes villes comme Londres.

Les dernières technologies connues visant spécifiquement une reconnaissance faciale efficace, utilisent des bases de *deep learning* et d'Intelligence Artificielle pour avoir un enregistrement des visages, et pour continuer à évoluer, en devenant de plus en plus précis. Dans le cadre de ce projet, nous n'utiliserons pas ce système d'apprentissage en profondeur, mais nous utiliserons la méthode d'*Analyse en Composantes Principales*.

Lors de ce projet, notre objectif est de créer un modèle étant capable de reconnaître un visage appartenant à notre base d'apprentissage. Nous avons donc constué notre propre base d'images en noir et blanc pour développer et perfectionner notre modèle qui sera codé en Java.

2 Analyse en Composantes Principales

2.1 Le concept de l'ACP

[4] Apparue au *XXe* siècle, l'*Analyse en Composantes Principales* est une technique d'analyse statistique. Elle est utilisée pour l'étude d'échantillons à plusieurs caractéristiques et est donc parfaitement adaptée à l'analyse multidimensionnelle. Un visage est très complexe dans sa quasi-unicité et dans la variété des expressions faciales qu'il peut prendre. Ce faisant, la méthode de l'*Analyse en Composantes Principales* se prête parfaitement à une étude d'un problème multidimensionnel. L'*Analyse en Composantes Principales* est une méthode de réduction de la dimension qui est souvent utilisée pour transformer un grand ensemble de variables en un ensemble plus petit qui contient encore la plupart des renseignements dans le grand ensemble

Les informations étant en général corrélées, on ne peut pas les dissocier, on doit les étudier comme un ensemble de données.

2.2 Le principe de la réduction de la dimension

[1] Traiter numériquement des images demande de considérer chaque pixel de ces dernières. Pour une image 100 x 100 pixels, il y a donc 10 000 données à traiter. Si nous considérons une dizaine d'images, cela fait 100 000 données. Notre objectif dans ce projet requiert de comparer les images les unes aux autres. Le nombre de traitements serait alors gigantesque et ne sera pas supporté par les machines actuelles. C'est pour cela que nous avons besoin de réduire la dimension de nos données.

L'analyse en composante principale est une méthode bien connue de réduction de dimension. Elle consiste à transformer des variables corrélées aux nouvelles variables décorrélées les unes des autres. Autrement dit, il s'agit de résumer l'information contenue dans un ensemble de données en un certain nombre de variables synthétiques : les composantes principales.

Dans le cadre de la reconnaissance faciale dans ce projet, nous allons appliquer les calculs de l'ACP sur la matrice des vecteurs centrés juxtaposés. Ensuite, nous projetterons le résultat selon un nombre K de caractéristiques qui, selon nous, contiennent les informations nécessaires à la reconstruction d'une image.

2.3 Application du concept de l'ACP

[2] L'application du concept de l'ACP pour la reconnaissance faciale est la suivante. Le but est de rechercher les composantes principales de chaque visage que nous allons représenter sous la forme de vecteurs propres.

Pour commencer, chaque image sous forme de matrice va être transformée en vecteur, comme suit :

$$A = \begin{pmatrix} a & d \\ b & e \\ c & f \end{pmatrix} \longrightarrow vect(A) = \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}$$

Une fois toutes les matrices sous la forme de vecteurs nous les rassemblons dans une matrice τ de la forme suivante :

$$\tau = \begin{pmatrix} a_{1,1} & b_{1,1} & \cdots & z_{1,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & b_{n,1} & \cdots & z_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,m} & b_{1,m} & \cdots & z_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,m} & b_{n,m} & \cdots & z_{n,m} \end{pmatrix}$$ où a, b, ..., z representent des images différentes

Il faut ensuite calculer l'image moyenne ainsi :

$$\Psi = \frac{1}{N} \sum_{i=1}^{N} \tau_i \text{ , avec N représentant le nombre d'images d'apprentissage}$$

Nous pouvons ensuite soustraire l'image moyenne à chaque image :

$$\phi_i = \tau_i - \Psi \qquad \text{avec i=1 ... N}$$

Par la suite, nous rassemblons tous les vecteurs des images centrées dans une matrice A sur laquelle nous appliquons l'ACP. Il nous faut trouver la matrice de covariance C en faisant le produit de A par sa transposée A^T. Or, le resultat de cette multiplication nous donnerait une matrice beaucoup trop grande qui serait très compliquée à traiter. Nous allons donc d'abord multiplier A^T et A, nous retournant une matrice C' plus petite. À partir de cette matrice, nous devons déterminer les valeurs propres et vecteurs propres.

$$\begin{array}{cc} A^T A v_i = \lambda_i v_i & \text{\# définition de la valeur propre} \\ A A^T A v_i = \lambda_i A v_i & \text{\# multiplication par A} \\ A A^t u_i = \lambda_i A v_i & \text{\# } u_i = A v_i \\ C u_i = \lambda_i u_i & \text{\# } C = A A^T \end{array}$$

Avec λ_i : valeurs propres de C', v_i : vecteurs propres de C' et u_i : vecteurs propres de la matrice C

Ainsi, nous avons obtenu les valeurs propres λ_i et les vecteurs propres $u_i = Av_i$ de la matrice de covariance C. Il faut maintenant trier les valeurs propres par ordre décroissant et ranger de manière correspondante les vecteurs propres. Une fois triés, nous allons retenir les k^e premieres valeurs et vecteurs propres afin de retenir au moins 80% de l'information. Les vecteurs propres retenus sont mis dans une matrice colonne E. Nous pouvons finalement construire la matrice G qui représente le poids de chaque visage propre pour chaque image, en projetant la matrice A sur la transposée de E.

$$G = E^T A$$

Maintenant la phase d'apprentissage est terminée. À partir des matrices E et G, nous pouvons déterminer si une image appartient à notre base de données.
Pour cela, comme pour l'apprentissage, nous mettons notre image I sous forme de vecteur. Dans un premier temps nous, allons calculer le vecteur de poids ω de l'image P :

$$\omega = E^T P$$

Ensuite, il nous faut calculer la distance (on choisit ici la distance quadratique) entre ω et chacun des vecteurs de la matrice G : $d(\omega, g_i) = \left|\omega - g_i\right|_2$

Le visage reconnu sera donc celui pour lequel la distance entre son poids et ω est la plus petite. Il est aussi important de définir un seuil de distance au-delà duquel un visage sera considéré comme non reconnu afin d'éviter une correspondance même si le visage demandé n'est tout simplement pas dans la base de donnée.

3 Analyse UML

Afin de conceptualiser notre programme et d'affiner notre compréhension, nous avons dans un premier temps réalisé des diagrammes UML.

3.1 Diagramme de cas d'utilisation

Lorsque nous exécutons notre programme, l'utilisateur peut réaliser deux actions. Il peut choisir une image de notre base de test pour que le modèle tente de reconnaître le visage. Il peut également visualiser l'évolution de l'erreur d'une image lors de sa reconstruction. Cela donne ce diagramme de cas d'utilisation :

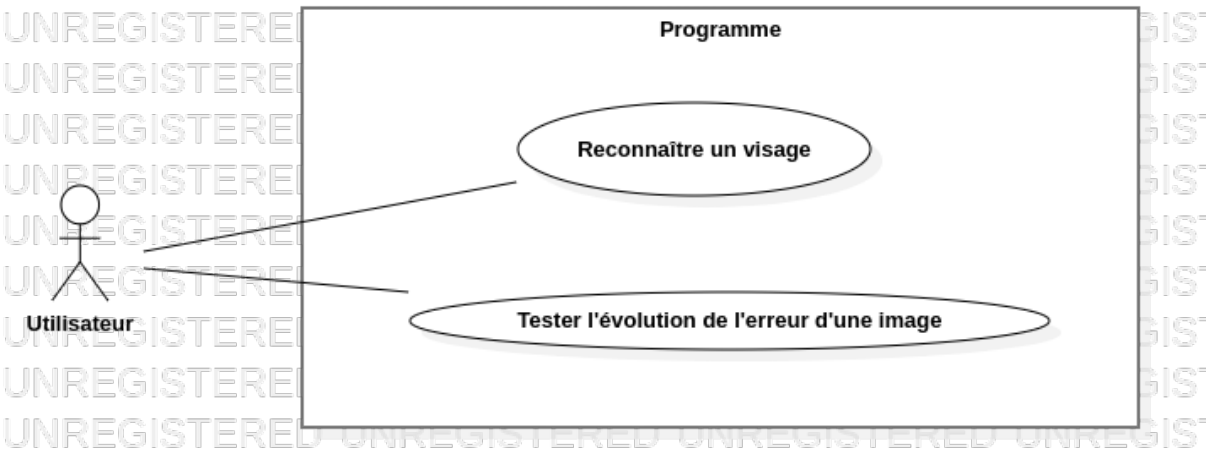


FIGURE 1 – Diagramme de cas d'utilisation

3.2 Diagramme de classes

Pour le diagramme de classes, nous avons décidé d'avoir trois classes instanciées qui seront :

— **ElementEnregistreMap** : assococie un nom, un prénom et une expression selon un id.
— **ImageBase** : image correspondant à un visage, qui peut être sous forme de matrice ou de vecteur.
— **Base** : contient les 57 images de notre base et permet d'effectuer tous les cacluls d'ACP et de projections.

Voici le diagramme de classes de notre projet :

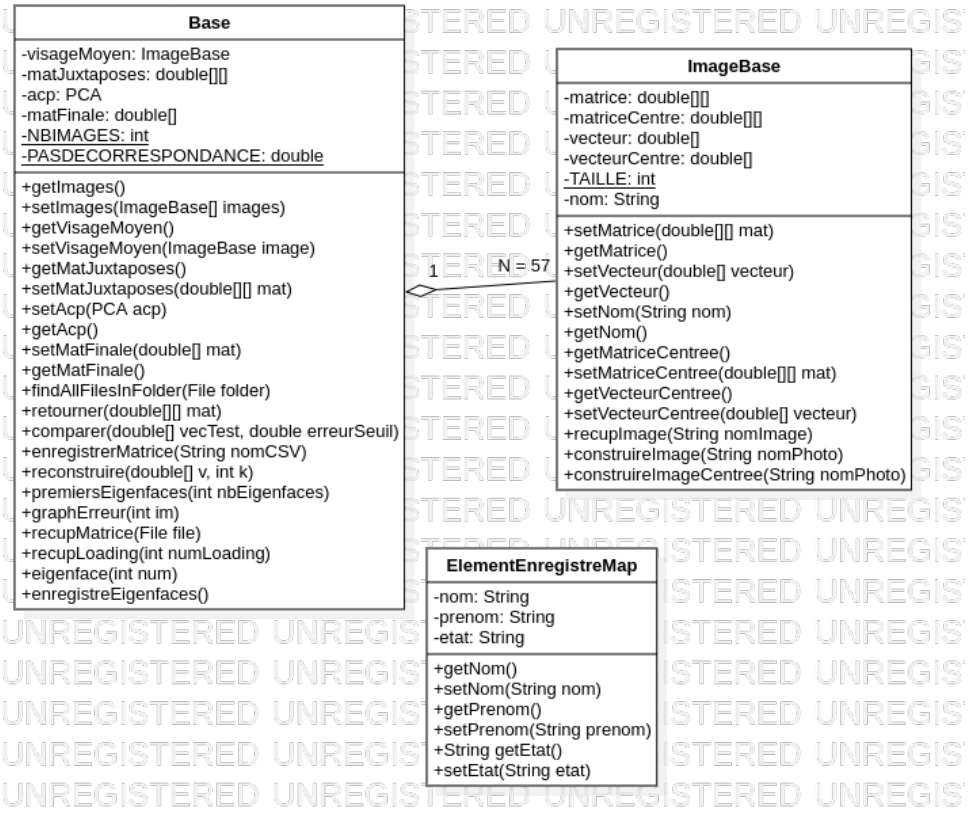


FIGURE 2 – Diagramme de classes du programme

4 Base d'images

4.1 Constitution de la base d'image

Pour constituer notre base d'image, nous nous sommes réunis avec les groupes 4, 16 et 19 pour prendre des photos de nos visages. Nous avons essayé de faire quatre expressions différentes, trois pour la base d'apprentissage et une pour la base de tests. Ainsi, nous avions constitué une base de 36 images prises dans les mêmes conditions d'éclairage et de cadrage. En voici deux exemples :

(a) Photo de David Lasgleizes (b) Photo de Justine Ribas

FIGURE 3 – Photos de la base d'images de deux personnes du Groupe 10

Afin d'enrichir notre base d'image, nous avons décidé d'ajouter les images du Groupe 5. Les photos sont prises sous un angle et une luminosité différente. Ce groupe nous a aussi suggéré l'idée de glisser des images d'inconnus dans la base de tests. Ainsi, nous pourrions tester des visages qui n'appartiennent pas à notre base d'apprentissage et voir si notre modèle nous donne le bon résultat. Il y a alors 3 images de visages inconnus dans la nouvelle base de tests.

Notre base constitue donc **74 images** au total, 57 images d'aprentissage et 17 images de tests.

4.2 Organisation de la base d'image

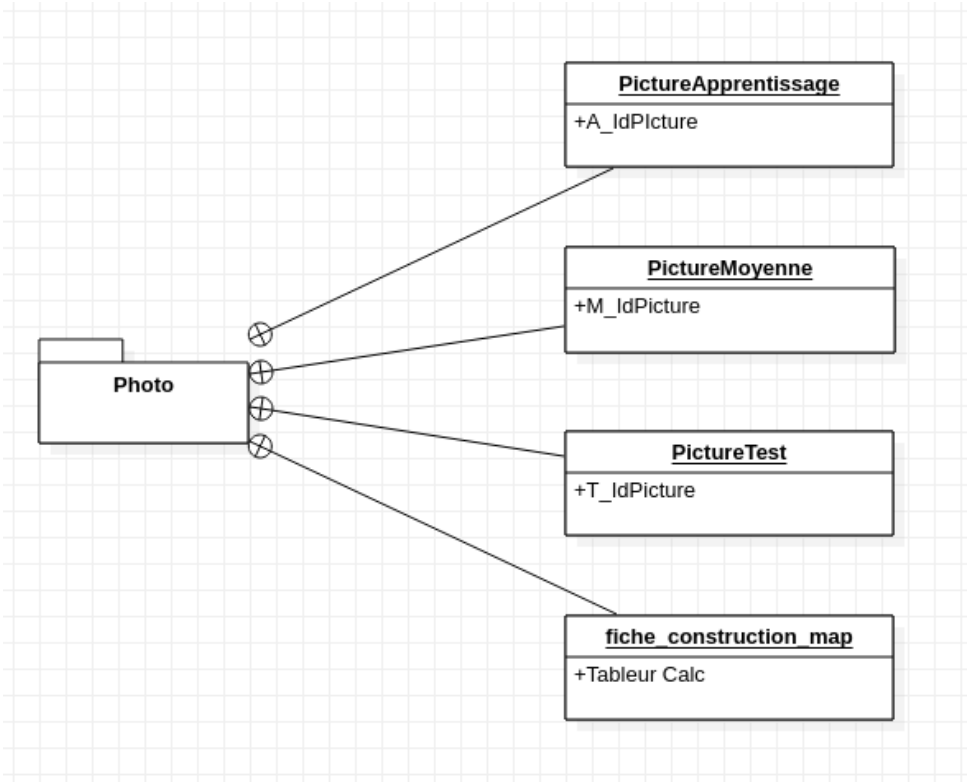


FIGURE 4 – Shématisation de la base d'image

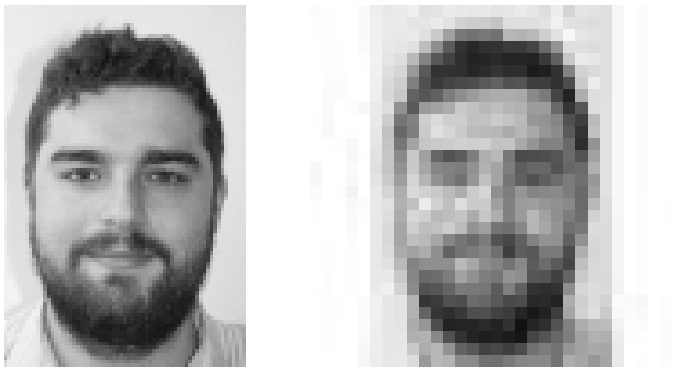
Notre base de données va se baser sur le schéma ci-dessus. On va retrouver dans ce dossier toutes les images relatives au projet.

La nomenclature de chaque image va se faire de la façon suivante : le premier caractère du nom représente le type d'image dont il s'agit (A, M et T, respectivement pour apprentissage, moyen et test). Après l'underscore on retrouve l'id spécifique de l'image. Cet id permettra, de retrouver notamment le nom et le prénom de la personne à qui le portrait appartient, ou bien l'émotion qu'il transmet dans la-dite image.

La correspondance entre l'id et le nom sera gérée grâce à un fichier csv que l'on va exploiter en tant que map dans java. En effet, nous avons fait le choix d'utiliser une map pour manipuler les données des images. Pour chaque image, la clé représente son id, la valeur est une classe qui nous permet de stocker les trois informations : nom, prénom et état.

4.3 Choix d'implémentation

En ce qui concerne notre base d'images, nous avons dû faire plusieurs choix. Tout d'abord, nous avons réduit la taille des images à 30x30 pixels. C'est le juste-milieu que nous avons trouvé entre le temps de calcul convenable et la reconnaissance du visage par l'utilisateur.



(a) Photo en 100x100 pixels (b) Photo en 30x30 pixels

FIGURE 5 – Photos de David Lasgleizes

(a) Photo en 100x100 pixels (b) Photo en 30x30 pixels

FIGURE 6 – Photos de Justine Ribas

5 Description du programme

5.1 Application de l'ACP dans le programme

Image moyenne

Dans notre programme, nous récupérons toutes les images de la base d'apprentissage. Pour chacune d'entre elles, nous avons une matrice et un vecteur colonne qui représentent chaque pixel par une valeur entre 0 et 1. À partir de chacun de ces vecteurs, nous calculons l'image moyenne.

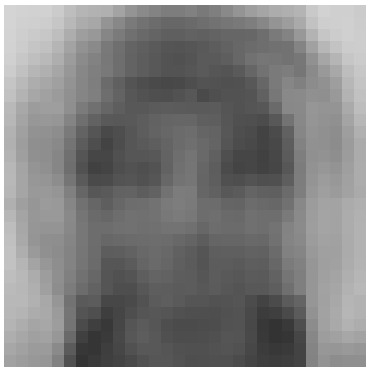
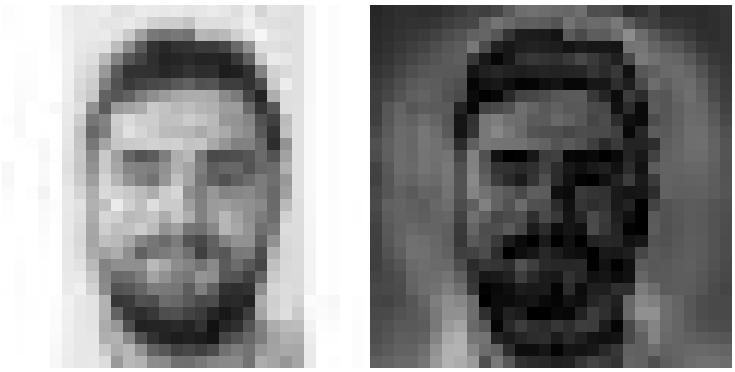


FIGURE 7 – Image moyenne de notre base d'image

Images centrées

Grâce à l'image moyenne, nous avons ensuite centré toutes les images de notre base. À partir des vecteurs centrés de chaque image, nous construisons une matrice qui juxtapose ces vecteurs centrés.



(a) Image originale (b) Image centrée

FIGURE 8 – Image originale et image centrée de David Lasgleizes

Matrice des vecteurs projetés

Nous appliquons ensuite l'ACP sur la matrice composée des vecteurs centrés juxtaposés. Nous avons décidé de projeter le résultat de l'ACP en gardant 12 caractéristiques sur 900.

Pour faire les calculs de l'ACP, nous avons décidé d'importer la librairie **smile** dans laquelle de nombreuses méthodes, utiles à nos calculs, étaient déjà implémentées.

Le résultat de l'application de l'ACP sur notre matrice des vecteurs juxtaposés nous donne une nouvelle matrice, la matrice des vecteurs projetés dans notre base d'ACP. Dans notre code, on la nomme la matrice finale G. Pour éviter de faire toujours les mêmes calculs à chaque exécution du programme, nous avons décidé de stocker les valeurs de cette matrice dans un fichier CSV. Nous avons réalisé la lecture de l'écriture dans les fichiers CSV grâce à la librairie **opencsv**.

Reconnaissance d'une image

Pour tester si une personne d'une nouvelle image appartient bien à notre base de test et pour la reconnaître, nous transformons tout d'abord cette image en vecteur, la centrons puis la projetons. Ensuite, nous comparons chaque vecteur de notre base avec ce nouveau vecteur. Nous calculons la distance entre deux vecteurs en calculant l'erreur quadratique entre les coordonnées puis en faisant la moyenne. Si cette moyenne est supérieure à **0.1**, l'image n'est pas reconnue.

Reconstruction d'une image

Pour reconstruire une image à partir d'un vecteur projeté, nous avons effectué le calcul suivant :
Soit k le nombre de caractéristiques retenues.
Soit V le vecteur d'une image projeté dans la base (de taille k).
Soit U la matrice des eigenfaces de V (de taille 900 x k).
Soit VR le vecteur reconstruit à partir de sa projection (de taille 900).
Alors : $VR = U \times V + M$

Calcul de l'évolution de l'erreur de reconstruction

Finalement, nous avons calculé l'évolution de l'erreur de reconstruction en fonction de nombre de caractéristique retenues. Pour ce faire, pour une image choisie, nous l'avons projetée puis reconstruite 18 fois en gardant de 1 à 851 caractéristiques. À chaque fois, nous avons comparé l'image non projetée et l'image reconstruite pour en calculer l'erreur. Enfin, nous avons affiché ces valeurs d'erreur dans un graphique.

5.2 Interface utilisateur

Nous avons développé l'interface graphique avec la technologie **JavaFX** pour laquelle nous avons également dû importer une librairie. Elle s'organise en quatre parties dans l'ordre du déroulé du processus. En voici un apperçu global :

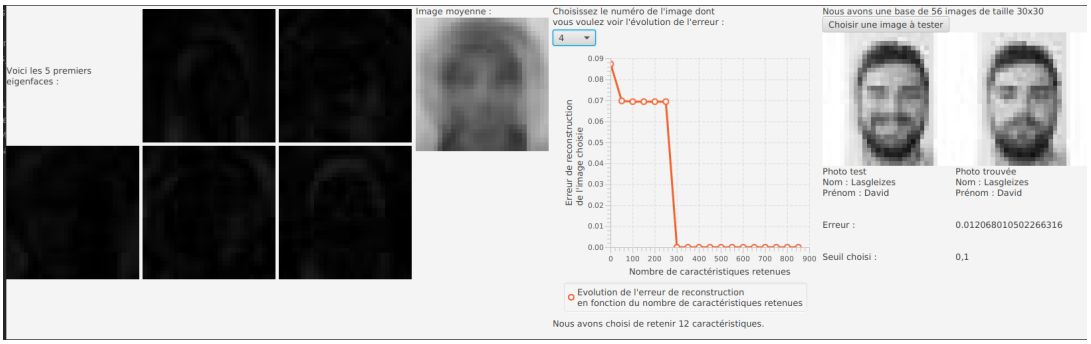


FIGURE 9 – Représentation de notre interphace graphique

Dans la première partie, nous affichons les **cinq premiers eigenfaces** que nous avons obtenus :

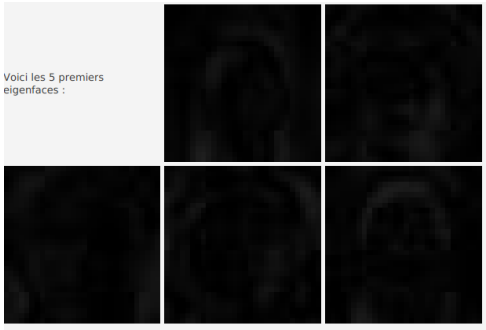


FIGURE 10 – Affichage des cinq premiers eigenfaces sur notre interphace graphique

Dans la deuxième partie, nous affichons notre **image moyenne**.

FIGURE 11 – Affichage de l'image moyenne sur notre interphace graphique

Dans la troisième partie, l'utilisateur peut choisir le numéro de l'image pour laquelle il souhaite afficher le **graphique d'évolution de l'erreur lors de sa reconstruction**.

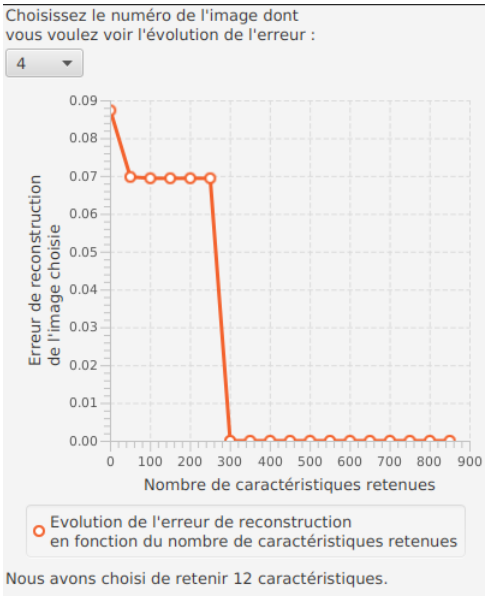
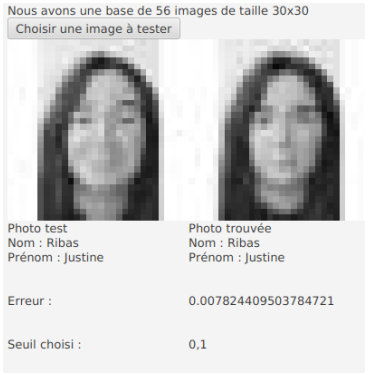
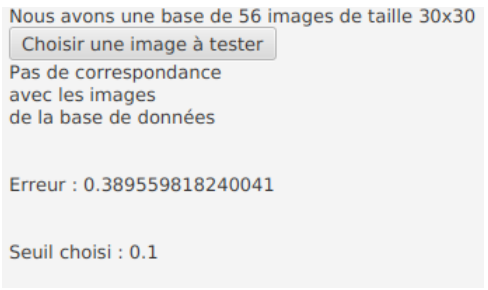


FIGURE 12 – Affichage du graphique d'évolution de l'erreur lors de la reconstruction de la 4ème image, sur notre interphace graphique

Dans la quatrième partie, l'utilisateur peut sélectionner une image dans le dossier "Photo" afin de la faire **tester** par notre programme. L'image la plus ressemblante sera affichée. Dans le cas où le visage n'a pas été reconnu, le programme l'affiche également.



(a) Test de l'image T_020.jpg de Justine Ribas



(b) Test de l'image T_074.jpg d'un visage inconnu

FIGURE 13 – Affichage deu test d'une image sur notre interphace graphique

6 Commentaires sur les résultats obtenus

6.1 Fiabilité du modèle

Globalement notre solution est fonctionnelle. Lorsque nous sélectionnons une image de test, le programme reconnaît bien le bon visage en affichant l'image de la base d'apprentissage la plus ressemblante (images T_004.jpg à T_071.jpg).

Dans le cas contraire, lorsque nous testons une image de la base test dont la personne n'apparaît pas dans la base d'apprentissage, le programme indique bien qu'il ne reconnaît pas la personne (T_072.jpg, T_073.jpg et T_074.jpg).

Cependant, il y a deux images de la base test pour lesquelles le programme ne reconnaît pas les visages : l'image T_036.jpg et T_043.jpg.

Finalement, on peut dire que notre modèle est fiable sur **88,2%** de nos images tests.

Si l'on se penche sur nos résultats intermédiaires comme notre image moyenne ou encore nos eigenfaces, on peut également voir qu'ils sont cohérents.

Enfin, lorsqu'on affiche le graphe de l'évolution de l'erreur lors de la reconstruction de l'image, les valeurs sont également cohérentes : plus le nombre de caractéristiques retenues augmente, plus l'erreur de reconstruction diminue. Nous pouvons voir que pour chaque image, il y d'abord un pallier d'erreur à environ 0.07 pour K allant de 20 à 250. A partir de 300 caractéristiques retenues, l'erreur devient très proche de 0.

6.2 Points positifs et négatifs du programme

Notre solution est plutôt bonne, car le modèle donne de bon résultat. De plus, nous avons développer une interface utilisateur intuitive et facile à utiliser. Cependant, nous avons quelques pistes pour améliorer notre programme :

- Le temps d'exécution du programme prend encore quelques secondes. Nous avons essayé de sauvegarder dans des fichiers CSV un maximum d'information. Néamoins, ayant utilisé une librairie pour les calculs d'ACP, nous ne pouvons pas tout sauvegarder et sommes obligés de recalculer certaines données.
- À cause du temps d'exéction, une nouvelle fois, nous avons été contraints de réduire les images à 30 x 30 pixels. Visuellement, ce n'est pas idéal pour l'utilisateur. Nous aurions aimé pouvoir traiter des images de 100 x 100 pixels.
- Afin d'améliorer la fiabilité, nous trouverons intéressant d'ajouter beaucoup plus d'images à notre base et de potentiellement revoir le seuil d'acceptabilité.

7 Conclusion

7.1 Organisation au sein de l'équipe

Pour l'organisation du projet, nous avons décidé lors de notre première réunion, d'appliquer une méthode vue en cours de Relations Humaines et Communication. Chacun s'est exprimé sur son rythme de travail, ses forces et ses faiblesses. Nous avons constaté que certains ont plus de mal à garder un rythme régulier de travail. Ainsi, pour garder une motivation et émulation collective, nous avons décidé de nous réunir une fois par semaine au minimum.

Pour faire un bilan sur cette décision, nous avons plutôt bien réussi à tenir le rythme d'une réunion par semaine, excepté les semaines où nous avions beaucoup de travaux dans d'autres matières. En ce qui concerne la répartition des tâches nous avons essayé de sectoriser ces dernières :

— **Maxime Abade** s'est occupé de tout l'aspect mathématique, il nous a guidés dans notre compréhension de l'ACP.

— **David Lasgleizes** a créé toute la base d'images. Il a réfléchi aux problèmes d'organisation et a développé tout le code permettant de créer et de récupérer les données de la map.

— **Justine Ribas** a réalisé le traitement des images. Elle a transformé les images en matrices de valeurs entre 0 et 1. Elles s'est aussi occupé de l'opération inverse qui consiste à rematérialiser nos images résultats comme l'image moyenne ou les images centrées. Pour finir, elle s'est occupée de l'écriture et de la lecture dans les fichiers CSV.

— **Amandine Legrand** s'est occupée d'implémenter toutes les notions mathématiques dans le programme. Elle s'est occupée de toutes les fonctions visant à modifier les matrices associées aux images, d'intégrer les fonctions relatives à l'ACP. Elle s'est aussi occupée de la projection des images et de la gestion de l'erreur. Enfin, elle a contribué à la finalisation de l'interface graphique.

— **Candice Morelli** s'est occupée dans un premier temps de comprendre l'aspect mathématique du projet. Puis par la suite, elle s'est occupée de l'interface graphique. Elle a organisé les différentes composantes graphiques au mieux pour s'adapter à l'utilisateur.

Pour ce projet, nous nous sommes familiarisés avec l'outils GitLab. Au début du projet, nous nous sommes formés ensemble pour être sûrs que chacun puisse utiliser cet outil correctement. Nous n'avons donc pas eu de problème particulier à utiliser GitLab. Cela nous a même beaucoup aidé et nous a facilité le travail en équipe.

7.2 Conclusions personnelles

Voici nos conclusions personnelles ainsi que nos ressentis et impressions sur ce projet :

— **Maxime Abade**
J'ai bien aimé le thème du projet. Je l'ai trouvé cohérent par rapport au contexte actuel. Le fait de mêler une grande approche mathématique à de la programmation informatique nous montre que le code sans réelle réflexion n'apporte rien. De plus, l'encadrement de la part du corps professoral était plus qu'agréable. Il en va de même pour la cohésion du groupe, ainsi que de son rythme, et implication dans ce travail collectif. Ce projet m'aura motivé à approfondir mes connaissances, à effectuer un véritable travail de recherche et de compréhension.

— **David Lasgleizes**
J'appréhendais ce projet notamment sur le point de vue du nombre de personnes des groupes. Cependant, après la réalisation du premier rendu, j'ai compris la nécessité de ce nombre. Le sujet quant à lui était très enrichissant, cela m'a permis de comprendre que les sujets qui nous ont été présentés en ce début d'année sont à approfondir.

— **Justine Ribas**
J'ai globalement aimé ce projet. J'ai trouvé que le sujet de la reconnaissance faciale était intrigant et accrocheur. Techniquement parlant, il n'y a pas eu de difficultés particulières, le plus dur était la compréhension du sujet et de l'ACP. Grâce à ce projet, j'ai pu mettre en application concrète des notions vues en cours de Java et cela m'a parfois aidé à combler des lacunes. Enfin, ce projet m'a également apporté une nouvelle expérience de travail en équipe. En effet, il est assez rare d'être aussi nombreux pour un projet scolaire. Nous avons donc appris à répartir et à sectoriser les tâches, même si ce n'était pas toujours facile de partager équitablement.

— **Amandine Legrand**
Même si le projet a demandé beaucoup de travail personnel, j'ai bien aimé codé le programme en Java et voir le résultat devenant de plus en plus concret. La partie programmation n'a pas posé problème, si ce n'est la compréhension liée à la librairie importée Smile. Pour les mathématiques, l'ACP a été plus dure à assimiler, mais nous y sommes arrivés grâce aux explications de nos professeurs. Concernant le travail en équipe, il s'est plutôt bien passé car, il y avait une bonne entente dans le groupe. Nous aurions par contre dû mieux répartir les tâches au début, pour que certains ne se retrouvent pas avec beaucoup plus de travail que d'autres.

— **Candice Morelli**
Lors de la première présentation du projet, je le trouvais intéressant, mais j'avais quelques difficultés sur la compréhension de la partie mathématique. Puis, après une bonne lecture du projet et des explications de nos professeurs encadrant, je l'ai bien compris. Le projet m'a permis de mieux comprendre l'utilité du principe de l'ACP vue au premier semestre. J'ai pu également réaliser la partie que je préfère réaliser dans un projet, c'est-à-dire l'interface graphique. Le projet s'est bien déroulé grâce à une bonne organisation, communication et entente au sein du groupe.

Bibliographie

[1] Yohann C. Qu'est-ce que l'analyse en composantes principales ? https://datascientest.com/acp, 2021.

[2] RANDRIAMAHANDRY Vonjinirina Eric. Reconnaissance faciale par methode acp hybride. http://biblio.univ-antananarivo.mg/pdfs/RandriamahandryVonjinirinaE_ESPA_MAST_2016.pdf, 2014 /2015.

[3] Frank Hersey. Analyse en composantes principales. https://www.biometricupdate.com/202202/live-facial-recognition-resumes-for-london-met-as-law-enforcement-seeks-wider-us 2022.

[4] Wikipédia. Analyse en composantes principales. https://fr.m.wikipedia.org/wiki/Analyse_en_composantes_principales.