Polytechnique Montréal

INF6803-Traitements vidéos et applications

TP3 - Reconnaissance de visages par EigenFaces

Auteurs:
Desmet Laurent 1865564,
Zins Pierre 1863527

Avril, 2017

Contents

1 Présentation de notre algorithme

Pour ce TP, nous avons voulu implémenter la méthode de reconnaissance de visages par *EigenFaces*. Cette méthode est composée de 2 phases différentes. La première consiste en un apprentissage, à partir d'un ensemble de données d'entraînement par une Analyse des Composantes Principales (apprentissage non supervisé). L'idée générale est d'extraire les "features" essentielles des images dans le but de les compresser et garder seulement les informations "importantes". La seconde phase aura pour objectif de déterminer si un nouveau visage est connu ou non. On pourra également savoir si une photo ne contient pas de visage.
Pour les données d'entraînement, nous nous sommes basé sur le dataset `"ORL Database of faces"` de l'Université de Cambridge. Il est composé de 40 personnes avec 10 images pour chacune d'entre elles. Les images (92x112 pixels) sont en noir et blanc, et bien centrées autour du visage des personnes. Pour la seconde phase, nous utilisons de nouvelles images, qui ne faisaient pas partie de l'ensemble d'apprentissage. Certaines images présentent des personnes faisant partie de l'ensemble d'entraînement et d'autres non. Nous avons donc utilisé 9 images de chacune des personnes pour l'entraînement (9x40=360 images) et 1 image pour le test (1x40=40 images). Nous avons ensuite ajouté d'autres images pour le test : des visages inconnus et des photos ne contenant pas de visage.

Voici une rapide description des étapes importantes de la méthode d'entraînement et de test :

Entraînement

1. On calcul l'image moyenne des images d'entraînement.
2. On retire l'image moyenne à chaque image et elles sont transformées sous la forme d'un vecteur (1034x1).
3. On rassemble tous les vecteurs dans une matrice (1034xN).
4. On calcul ensuite la matrice de covariance de la matrice obtenue à l'étape précédente et on détermine les vecteurs propres.
5. On conserve uniquement les M vecteurs associés aux valeurs propres les plus grandes. Ces M vecteurs vont définir notre nouvel espace *"Face Space"*.
6. Pour chaque image d'entraînement, nous allons calculer des poids *wi*, qui correspondent à la projection du visage dans le *"Face Space"*.

Test

1. Pour chaque image test, nous retirons l'image moyenne (obtenue dans la phase d'entraînement) et nous la transformons sous forme de vecteur (1034x1).
2. Nous projetons chaque image, dans le *"Face Space"*. Cela correspond à obtenir les poids *wi*.
3. Pour faire l'identification, nous comparons simplement les poids *wi* de l'image test avec les poids *wi* de chaque image d'entraînement. La comparaison se fait par le calcul de la distance Euclidienne entre les 2 vecteurs de poids.

Nous avons également implémenté la partie permettant de reconstruire une image d'entraînement après avoir appliqué le PCA. Cette partie n'était pas nécessaire pour faire l'identification des visages, mais était intéressante pour voir l'impact du nombre de vecteurs propres conservés sur la quantité d'information perdue. De plus cela permettait de s'assurer du fonctionnement de notre PCA.

2 Implémentation

Dans cette section, nous allons présenter quelques points de notre implémentation.

Comparaison des poids *wi* Lors de l'identification d'un visage test, nous faisons la comparaison des poids *wi* de l'image test avec tous les poids *wi* des images d'entraînement. Dans l'article présentant la méthode **EigenFace** [1], les auteurs proposaient de calculer pour chaque **personne** de l'ensemble d'entraînement un vecteur de poids moyen. C'est à dire qu'à la place d'avoir 9 vecteurs de poids pour les 9 images de la même personne, nous n'aurions qu'un seul vecteur de poids moyen. Cela permet ensuite de limiter le nombre de comparaisons des poids lors de la phase d'identification d'un visage. Cependant, après avoir essayé cette méthode, nous avons pu voir que l'identification était moins bonne. De plus, nous avons seulement 9 images pour chaque personne d'entraînement, donc le nombre de comparaison restera limité dans notre cas. Ainsi, nous avons conservé 9 vecteurs de poids pour les 9 images d'entraînement de chaque personne, plutôt qu'un vecteur de poids moyen.

Nombre de vecteurs propres conservés Pour la procédure PCA, il est important de définir le nombre de vecteurs propres que l'on conserve. Pour cela, nous définissons un seuil qui représente le pourcentage : quantité d'information conservée par rapport à la quantité d'information totale. En fixant, ce seuil à 1, nous gardons

tous les vecteurs propres et nous conservons donc la totalité de l'information. Après la reconstruction, nous obtenons l'image initiale. Dans l'article présentant **Eigen-Face** [1], les auteur proposent un seuil de 0.95. Cela revient à conserver les 177 (sur 360) vecteurs propres contenant le plus d'information. De plus, en visualisant la reconstruction des images, il est intéressant de voir l'influence du nombre de vecteurs propres conservés par rapport à la qualité de la reconstruction. Cela est présenté par la suite dans la section *Expériences*.

Identification Pour l'identification d'un visage, nous calculons donc des distances entre les poids de l'image test avec ceux des images d'entraînement. Cependant, il est important de pouvoir détecter si un visage n'est pas connu (personne non présente dans la base d'entraînement) et si une image ne présente pas de visage. Pour cela, nous utilisons 2 seuils sur la distance entre les poids.

- `DISTANCE_FACE_THRESHOLD` : si la plus faible distance est supérieure à ce seuil, le visage est inconnu.
- `DISTANCE_SPACE_THRESHOLD` : si la plus faible distance est supérieure à ce seuil, l'image ne présente pas de visage.

Le seuil `DISTANCE_SPACE_THRESHOLD` sera plus grand que `DISTANCE_FACE_THRESHOLD` . Une des difficultés est de fixer une valeur à ces seuils, pour que chaque image test soit bien classifiée : visage correspondant, visage inconnu, non visage. Pour cela, nous avons fait évoluer notre base de test. Tout d'abord en ajoutant des visage non connus. Pour ces visages, les distances obtenues étaient pus importantes que pour les visages connus. Cela nous a permis de fixer le premier seuil (`DISTANCE_FACE_THRESHOLD`). Puis nous avons ajouté d'autres images qui ne correspondent pas à des visages. Les distances obtenues étaient encore plus importantes, et cela nous a permis de fixé le second seuil (`DISTANCE_SPACE_THRESHOLD`). Enfin, nous avons refait les tests avec de nouvelles images (visages inconnus et non visage) qui n'avaient pas été utilisées pour fixer les valeurs des seuils. Cependant, nous avons utilisé peu d'images pour cela, donc il est probable qu'avec de nouvelles images, certaines soient mal classifiées. Fixer des valeurs correctes pour ces seuils est difficile car cela dépend également du nombre de vecteurs propres conservés, puisque les valeurs des distances changent.

3 Expériences

Dans un premier temps, nous avons regardé notre taux d'identification avec des photos de personnes connues (qui étaient dans la base d'entraînement). Nous avons

donc utilisé 40 nouvelles photos (1 pour chaque personne), et nous avons fait varier le seuil correspondant au nombre de vecteurs propres conservés.

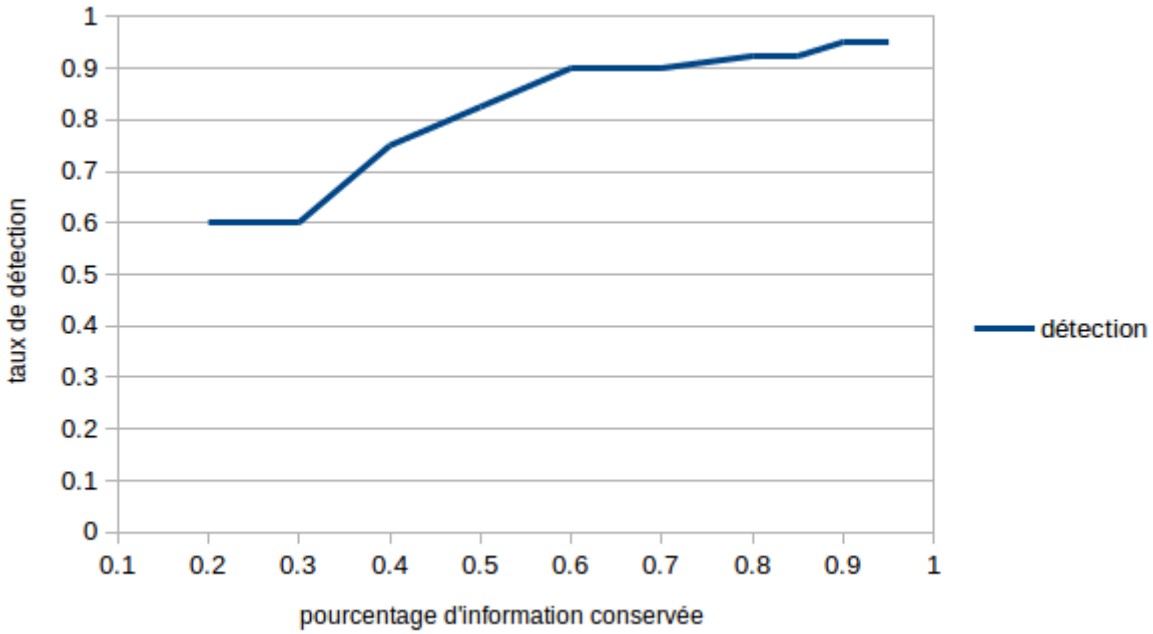

Figure 1: taux de détection en fonction de la quantité d'information conservée

On peut remarquer que notre détection dépasse 90% quand l'on garde 60% de l'information. Cela correspond à identifier correctement 36 images parmi les 40 en ne conserver que 11 vecteurs propres parmi les 9x40=360 de départ. La quantité d'information dans notre base d'entraînement est donc fortement réduite de cette manière, tout en conservant un taux d'identification correct. Nous avons également visualisé le nombre de vecteurs propres gardés en fonction du taux d'information.

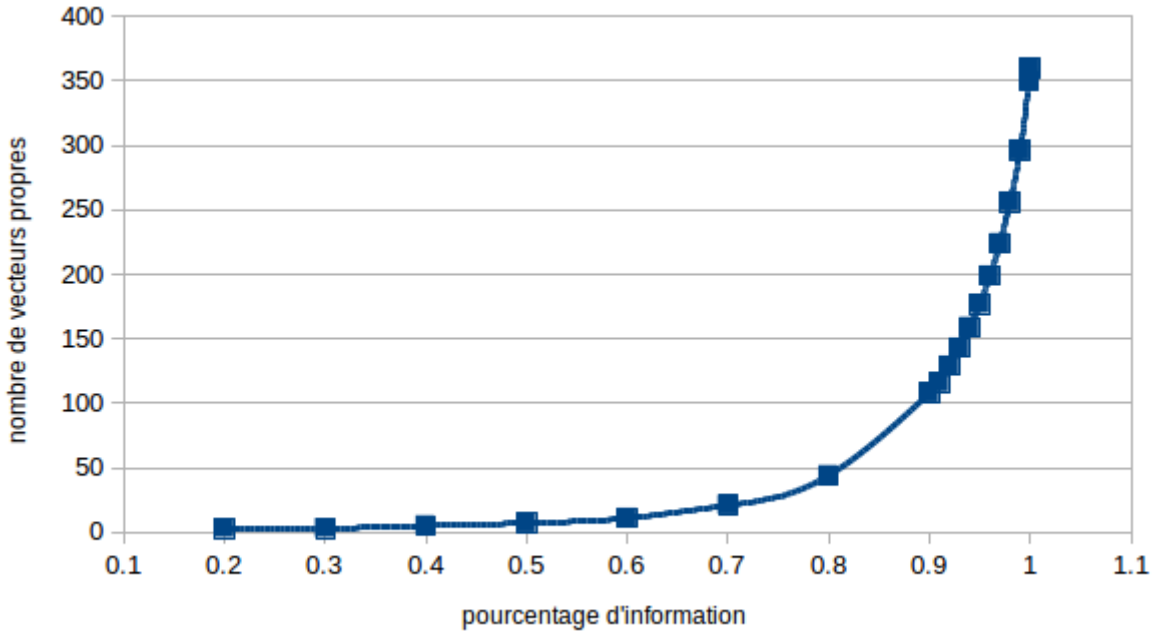

Figure 2: Nombre de vecteurs propres en fonction de la quantité d'information conservée.

Dans un second temps, nous avons visualisé la reconstruction d'images pour voir l'effet de la quantité d'information conservée par rapport à la qualité de l'image reconstruite.

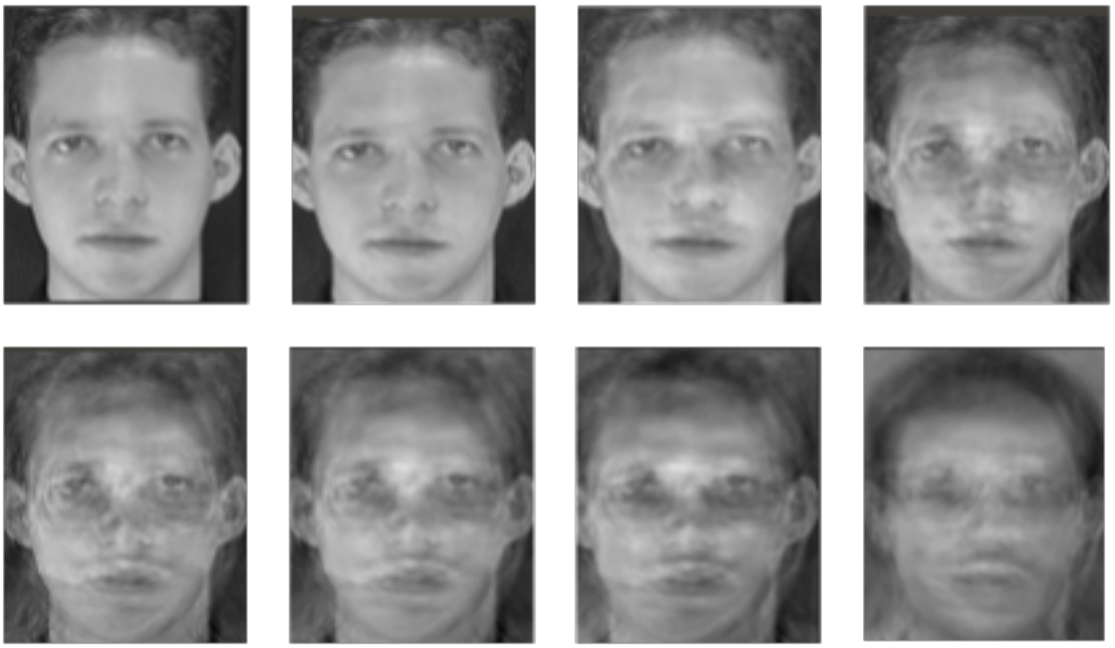

Figure 3: Visages reconstruits

Voici les paramètres utilisés pour la reconstruction :

- visage 1 : taux : 1 - 360 vecteurs propres
- visage 2 : taux : 0.999 - 350 vecteurs propres
- visage 3 : taux : 0.99 - 296 vecteurs propres
- visage 4 : taux : 0.98 - 256 vecteurs propres
- visage 5 : taux : 0.97 - 224 vecteurs propres
- visage 6 : taux : 0.96 - 199 vecteurs propres
- visage 7 : taux : 0.95 - 177 vecteurs propres
- visage 8 : taux : 0.9 - 108 vecteurs propres

Pour l'identification, nous avons vu qu'avec un seuil de 0.6 (11 vecteurs propres parmi 360), nous parvenons à dépasser 90% d'identification. Au contraire, au niveau de la reconstruction, la qualité des visages diminue très rapidement. Visuellement, il

est possible de reconnaître la personne jusqu'à un seuil entre 0.98 et 0.96 (256 - 199 vecteurs propres). Dès 0.9 (108 vecteurs propres), il n'est plus vraiment possible de reconnaître le visage.

4 Difficultés rencontrées

La principale difficulté de ce TP fût de trouver un dataset correct. Cet inconvénient majeur consiste à trouver plusieurs images d'une personne prise dans des conditions identiques et dont l'image est centrée sur un visage neutre. Ici nous utilisons un dataset qui nous semble correct bien que certains visages soient désaxés.
Le principe de base est également simple mais l'implémentation fût parfois plus compliquée, notamment au niveau des dimensions des matrices et du type de données que ces dernières contenaient. Ainsi, la partie la plus longue à coder n'était pas la décomposition en vecteurs propres mais bien la recomposition des images et la détection.
De plus, on peut éventuellent penser que notre algorithme aurait put être optimiser avec un meilleur dataset.
Enfin, nous avons essayé de faire une reconstruction par pixel, c'est à dire qu'il n'y avait plus 360 vecteurs propres mais bien un nombre égal à la taille de l'image. Ainsi, on pouvait voire une dégradation progressive de l'image. Cela consistait à rajouter un bruit poivre et sel dans l'image mais dont la fréquence augmentait significativement avec le nombre de vecteurs propres que l'on enlevait. Remarquons qu'ici cela ne nous donnait pas de meilleurs résultats, bien que la reconstruction soit assez robuste, mais pas aussi efficace pour la détection. Néanmoins, mous sommes dans un cas particulier où le nombre d'images est inférieure au nombre de pixels, mais dans la situation inverse, cette méthode peut-être envisager sérieusement car elle en résulterait à une diminution de mémoire significative.

5 Conclusion

Ce TP nous aura permis d'expérimenter une méthode d'apprentissage non supervisé par Analyse des Composantes Principales. Cette méthode permet d'extraire les features importantes des images et de les résumer au sein de vecteur : les vecteurs propres. En projetant une nouvelle image dans cet espace, il est alors simple d'observer une corrélation entre l'image test et une appartenant au set d'apprentissage. En outre, nous avons constaté que, dans notre exemple lié à un dataset bien précis, on pouvait conserver seulement 75% de l'énergie -ou information- pour acquérir des

résultats corrects de détection. Cela représente à peine plus de 10 vecteurs propres sur les 360 disponible originellement ! Cette méthode, en plus des avantages cités ci-dessus, permet de réaliser une importante compression des données.
Malheureusement, cet algorithme possède de nombreux de défauts. Déjà, il n'est pas généralisable dans le sens où il peut reconnaitre seulement une quantité finie de visage. Dans le cas où le nombre de visage augmentent sensiblement -par exemple, par un dataset fourni par la police- cela serait beaucoup trop lourds à traiter. De plus, le dataset doit être sélectionner sur des visages neutres, sans aucun artifice et centré sur le visage. Il faut également un nombre important d'image pour espérer une détection efficace du visage.
Ainsi, bien que nous estimons nos résultats satisfaisant, on pourrait éventuellement améliorer l'algorithme, notamment en y incorporant un pre-processing qui permettrait d'optimiser le dataset. Nous le savons, nous obtenons un dataset non idéal et nous pourrions tout à fait envisager d'appliquer un algorithme qui améliorerait les images du dataset pour rendre l'algorithme *EigenFaces* plus efficace. Par exemple, on pourrait éventuellement penser à un algorithme qui détecte et centre le visage d'une photo, ou bien qui ré-axe l'orientation des visages par reconstruction. De plus, nous ne disposons pas toujours de 10 voire 15 photos d'une même personne, et une pratique courante dans les algorithmes d'apprentissage est l'augmentation du dataset par transformations géométriques. Cela consiste à y appliquer des transformations (légères rotations, translations, ...) à partir d'une même image pour augmenter la taille du dataset et le rendre ainsi plus robuste. Ce type d'algorithme -qui reste du pre-processing- peut être également envisager ici.

Références

[1] A. Pentland M. Turk. Eigenfaces for recognition, journal of cognitive neuroscience.