Paper Summary: Eigenfaces for Recognition

Pham, Van Vung (R Number: R11562737)
Email: vung.pham@ttu.edu

***Abstract*—This work creates near-real-time system to track and recognize faces, it works by projecting the images into feature spaces using principle component analysis. A face is projected as linear combination of eigenfaces and to recognize faces we only have to compare the coefficients of the projection. This approach also provides a way to learn and recognize new faces in an unsupervised manner.**

1. Introduction

Human is good at both detecting and recognizing faces, therefore, computational models to do these are of high interest and demanded to be fast and accurate. The main idea of this work is to create a set of feature images ("face space") as principal components of the training images and so recognizing an image is done by projecting it into the face space and classifying an image is done by comparing the distance of it with the known faces in the face space. This work is relatively fast, simple, and insensitivity to small or smooth changes in the face images.

2. Background and related work

Early approaches were limited by the use of impoverished face features, recent parameterized feature models and multiscale matching faced several problems before they can be generally applicable, connectionist approaches tended to hide much of the pertinent information in the weights that maked it difficult to modify and evaluate.

3. The eigenface approach

This approch uses Principal Components Analysis (PCA) to find eigenvectors (eigenfaces) of the images. Any face can be represented as weighted sum of these eigenfaces.

The approach initialization includes:

1) Acquire an initial set of training images.
2) Use PCA to find M best eigenfaces (*face space*).
3) Project each known face images onto the "face space" and store the weights.

Following steps are used to recognize new faces:

1) Calculate a set of weights for the input image (by projecting it to each of the eigenfaces).
2) Determine if the image is a face by checking if it is close to the "face space".
3) If it is a face, classify the weight pattern as eighter a known or as unknown.
4) (Optional) Update the eigenfaces and/or weight patterns
5) (Optional) If the same unknown face is seen several times, calculate its characteristic weight pattern and incorporate into the known faces.

4. Calculating eigenfaces

Every face image $I(x, y)$ of size N by N is converted to N^2 vector. PCA is used to find vectors (eigenfaces) to best represent distribution of faces (*"face space"*).

Let $\Gamma_1, \Gamma_2, \Gamma_3, ..., \Gamma_M$ be the M training faces. The average face is $\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$ then compute the difference from average face $\Phi_i = \Gamma_i - \Psi$, then find M orthonormal vectors, u_n, which best describe the distribution of the data, k^{th} vector, u_k is chosen to maximize:

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^T\Phi_n)^2 \tag{1}$$

with respect to:

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1, & if l = k \\ 0, & otherwise \end{cases} \tag{2}$$

The vectors u_k and scalars λ_k are the eigenvectors and eigenvalues, respectively, of the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \tag{3}$$

Where $A = [\Phi_1\Phi_2...\Phi_M]$. C is N^2xN^2, leading to heavy computation. So, if $M < N^2$, the better approach is to solve the eigenvectors of the MxM matrix.

$$A^T A v_i = \mu_i v_i \tag{4}$$

Then premultiplying both sides by A, leadinng to

$$AA^T A v_i = \mu_i A v_i \tag{5}$$

Define MxM matrix $L = A^T A$, where $L_{mn} = \Phi_m^T\Phi_n$, then find the M eigenvectors, v_l of L. These are the eigenfaces u_i

$$u_l = \sum_{k=1}^{M} v_{lk}\Phi_k, \quad l = 1, ...M \tag{6}$$

5. Using eigenfaces to classify a face image

In practice, $M' < M$ is sufficient for identification. A new face (Γ) is transformed into face space using:

$$\omega_k = u_k^T(\Gamma - \Psi), \quad for\ k = 1, ..., M' \tag{7}$$

These make $\Omega^T = [\omega_1, \omega_2, ..., \omega_{M'}]$ can then be used to classify the input to face class k that minimizes:

$$\epsilon_k^2 = ||(\Omega - \Omega_k)||^2 \ (\Omega_k\ describes\ k^{th}\ face\ class) \tag{8}$$

Ω_i are calculated by averaging the results of the eigenface representation of each individual. A face is classified as class k when the minimum ϵ_k is smaller than a threshold θ_ϵ. Otherwise, the face is "unknown".

The distance ϵ between the image substracted by the mean ($\Phi = \Gamma - \Psi$) and its projection ($\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$) is:

$$\epsilon^2 = ||\Phi - \Phi_f||^2 \tag{9}$$

There are four cases: (1) near face space and near a face class, then classified (2) near face space but not near a known face class, then is unknown (3) distant from face space and near a face class, then is not a face (4) distant from face space and not near a known face class, then is not a face. As a summary, eigenfaces approach involves:

1) Collect a set of training face images (e.g., $M = 40$)
2) Calculate the (40x40) matrix L, find its M' (e.g., 10) best eigenvectors.
3) Use Eq. (6) to produce the (M') eigenfaces u_k.
4) For each known individual, calculate Ω_k by averaging the Ω [from Eq. (8)]. Choose a threshold θ_ϵ as the maximum allowable distance from any face class, and θ_ϵ as the maximum allowable distance from face space [using Eq. (9)].
5) For each new face image, calculate Ω, the distance ϵ_i to each known class, and the distance ϵ to face space. If the minimum distance $\epsilon_k < \theta_\epsilon$ and the distance $\epsilon < \theta_\epsilon$, classify the it as the class with vector Ω_k. If $\epsilon_k > \theta_\epsilon$ but $\epsilon < \theta_\epsilon$, then the image is "unknown", and optionally used to begin a new face class.
6) If the input is known, it could be added to the original set of familiar face images, and the eigenfaces may be recalculated.

6. Locating and detecting faces

Motion detecting and head tracking: In case of a single person moving in a static environment, we can use motion detection to find the moving part. Then the head is specified as the smaller motion blob above the bigger motion blob (the body). The algorithm can be summarized as 1) from video frames $I(x, y, t)$, 2) apply spatiotemporal filtering, 3) then apply thredholding, 4) apply motion analysis and 5) locate the head location (x, y).

Using "face space" to locate the face: The main idea is that the face image is closer to the face space comparing to non-face image. So at every location (x, y) of the image, calculate distance $\epsilon(x, y)$ between local subimage $I(x, y)$ centered at (x, y) and the face space to make the "face map". The darkest location in the face map (with low ϵ) is the potential place for the face. The direct application of Eq. (9) is expensive so we use another approach. To project subimage Γ onto face space, we subtract its from the mean as $\Phi = \Gamma - \Psi$. Let Φ_f is the projection of Φ in the face space, then the distance is calculated as:

$$\begin{aligned}\epsilon^2 &= ||\Phi - \Phi_f||^2 = (\Phi - \Phi_f)^T(\Phi - \Phi_f) \\ &= \Phi^T\Phi - \Phi^T\Phi_f\end{aligned} \tag{10}$$

Due to Φ_f^T and $(\Phi - \Phi_f)$ are perpendicular, their multiplication is $Zero$. In addition, $\Phi^T\Phi_f = \Phi_f^T\Phi_f$, $\Phi_f = \sum_{i=1}^{L} \omega_i u_i$, and eigenfaces are orthonormal, we have:

$$\Phi_f^T\Phi_f = \sum_{i=1}^{L} \omega_i^2 \tag{11}$$

and

$$\epsilon^2(x, y) = \Phi^T(x, y)\Phi(x, y) - \sum_{i=1}^{L} \omega_i^2(x, y) \tag{12}$$

also

$$\begin{aligned}\sum_{i=1}^{L} \omega_i^2(x, y) &= \sum_{i=1}^{L} \Phi^T(x, y)u_i \\ &= \sum_{i=1}^{L} [\Gamma(x, y) - \Psi]^T u_i \\ &= \sum_{i=1}^{L} [\Gamma^T(x, y)u_i - \Psi^T u_i] \\ &= \sum_{i=1}^{L} [I(x, y) \otimes u_i - \Psi^T u_i]\end{aligned} \tag{13}$$

Where the $\otimes$ is the correlation operator. In addition:

$$\begin{aligned}\Phi^T\Phi(x, y) &= [\Gamma(x, y) - \Psi]^T[\Gamma(x, y) - \Psi] \\ &= \Gamma^T(x, y)\Gamma(x, y) - 2\Psi^T\Gamma(x, y) + \Psi^T\Psi \\ &= \Gamma^T(x, y)\Gamma(x, y) - 2\Gamma(x, y) \otimes \Psi + \Psi^T\Psi\end{aligned} \tag{14}$$

So that

$$\begin{aligned}\epsilon^2(x, y) = \Gamma^T(x, y)\Gamma(x, y) - 2\Gamma(x, y) \otimes \Psi + \Psi^T\Psi \\ + \sum_{i=1}^{L} [\Gamma(x, y) \otimes u_i - \Psi \otimes u_i]\end{aligned} \tag{15}$$

Since Ψ and u_i are fixed, $\Psi^T\Psi$ and $\Psi \otimes u_i$ can be pre-calculated. In addition, all these calculation can be done using a simple Neural Network.

7. Learning to recognize new faces

This approach has the ability to learn and recognize new faces in an unsupervised manner. When a face is classified as "unknown", the computer store it and its pattern vector. If pattern vectors of "unknown" faces cluster in pattern space, there is a new face. They are checked for similarity (a threshold distance from the mean of the cluster), if passed, the average of the feature vectors is added to the database of known faces. Occasionally, the eigenfaces may be recalculated using these stored images.

8. Other issues

Eliminating the background: In practice, background can significantly effect the performance. To deal with this, the input face is multiplied by a two-dimensional Gaussian window centered on the face, thus diminishing the background and accentuating the middle of the face. Focusing on the center of the face is also having another advantage such as changing the hair style.

Scale (head size) and orientation invariance: The performance of this system decreases quickly as the head size/scale changes. The motion analysis gives an estimate of head size to rescale the face region to the eigenface size. Another approach is to use multiscale eigenfaces. There are also two methods used in case of head tilt. The first is to calculate the orientation of the motion blob of the head (less reliable as the shape tends be a circle). Second approach is based on the fact that faces are reasonably symmetric for frontal views, simple symmetry operators is used to estimate head orientation, this information is used to rerotate the image to align with the head with the eigenfaces.

Distribution in face space: The nearest-neighbor classification previously described assumes a Gaussian distribution in face space of an individual's feature vectors Ω. A better approach is to use Nonlinear networks described in [1] to learn face space distribution.

Multiple views: The approach to deal with other than full frontal views is to define a limited number of face classes for each known person corresponding to *characteristic views* e.g., frontal, side (at $+45^o or - 45^o$), right and left views.

9. Experiments with eigenfaces

A database of over 2500 faces digitized under controlled conditions. Sixteen subjects, three head orientations, three head sizes, and three lightning conditions. A six level Gaussian pyramid was constructed for each image, resulting in image resolution from 512x512 pixels down to 16x16 pixels.

First experiment checks the effects of varying lightning, size, and head orientation. Various groups of 16 images were selected and used as the training set. Within each training set there was one image of each person, all taken under the same conditions of lightning, size, and head orientation. All images in the database were then classified as being one of these sixteen individuals ($\theta_\epsilon = \infty$). Seven eigenfaces were used, and the achieved approximately 96% correct classification averaged over lightning variation, 85% over orientation variation, and 64% over size variation. So changing light condition causes relatively few errors, while size variation has high impact over performance.

Second experiment followed the same procedure, but the acceptance threshold θ_ϵ was also varied. At low values of θ_ϵ, only images that project very closely to the known face classes will be recognized, so that there will be few errors but many of the images will be rejeced as unknown. At high values of θ_ϵ most images will be classified, but there will be more errors. Adjusting θ_ϵ to achieve 100% accurate recognition boosted the unknown rate to 19%, while varying lighting, 39% for orientation, and 60% for size. Setting the unknown rate arbitrarily to 20% resulted in correct recognition rates of 100%, 94%, and 74% respectively.

Knowing these trade-offs would help the users set the appropriate configurations for corresponding situations.

10. Real-time recognition

This approach was applied to build a system to recognize faces near-real-time in a small and reasonably unstructured environment. Recognition occurs at rates of up to two or thee times per second.

11. Relationship to biology and neural nets

This work has several qualitative similarities with the current understanding of human face recognition. Gradual changes due to aging are easily handled by the occasional recalculation of the eigenfaces.

This approach can be implemented using simple parallel computing elements as in connectionist system or artificial neural network. A three-layer, fully connected linear network was used to implement significant part of the system.

12. Conclusion

This approach is based on information theory to do face recognition based on small set of image features that best approximates the set of known face images without the requirement of knowing intuitive notions of facial parts. This approach could be implemented using Neural network. It is relatively simple and applicable in near-real-time detection in a small scale reasonably unstructured environment. The experiments done also provide guides to user in order use this approach in different situations. This approach also provides a way to learn and recognize new faces in an unsupervised manner.

References

[1] M. K. Fleming and G. W. Cottrell. Categorization of faces using unsupervised feature extraction. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pp. 65–70. IEEE, 1990.