E-Faces - Um classificador capaz de analisar imagens e classificá-las como faces ou não faces utilizando o método Eigenfaces

Éder Augusto Penharbel, Erdiane L. G. Wutzke,
Murilo dos S. Silva, Reinaldo A.C. Bianchi

Centro Universitário da FEI - UNIFEI
Departamento de Ciência da Computação
Av. Humberto de Alencar Castelo Branco, n.º 3972, São Bernardo do Campo,
São Paulo, Brasil - CEP 09850-901 - Fone: (11) 4353 2900 - Fax: (11) 4109 5994

epenharbel@gmail.com, {ewutzke,msilva,rbianchi}@fei.edu.br

Resumo

A classificação de imagens como face ou não face pode ser um passo inicial na implementação de um sistema de detecção de faces. Este artigo descreve a implementação de um classificador capaz de analisar uma imagem e classificá-la como face ou não face. O E-Faces foi implementado utilizando o método *eigenfaces*, um método que necessita de poucas imagens para o treinamento e foi capaz de alcançar uma taxa de 95% de classificações corretas.

Palavras-chave: Visão Computacional, Classificação de Imagens, PCA, *Eigenfaces*, Detecção Facial.

1 Introdução

Faces de pessoas são extremamente importantes nas iterações humanas. Uma grande parte das pessoas que conhecemos estão relacionadas a imagens de faces em nossa mente. Através dessa relação podemos identificar pessoas em contatos visuais, fotografias, vídeos, pinturas, etc.
Essa relação face/pessoa pode ser utilizada em diversas situações. Um exemplo é quando abrimos a porta de nossa casa para algum amigo ou quando somos reconhecidos em um ambiente.
É com base nessas situações que torna-se útil a implementação de sistemas computacionais de classificação, detecção e reconhecimento facial.
A classificação, detecção e reconhecimento facial vem atraindo a atenção de diversos pesquisadores [1] para que sistemas rápidos, confiáveis e robustos sejam criados.
A implementação desses sistemas computacionais pode ser utilizada no desenvolvimento de sistemas autônomos de segurança de ambientes, na criação de portas ”inteligentes”, na criação de sistemas que reconheçam seus usuários através de faces e muitas outras aplicações.
Este trabalho descreve a implementação de um classificador de imagens como faces ou não faces através do método *eigenfaces* [2]. O método *eigenfaces* é um método baseado na análise de componentes principais, ou *Principal Component Analysis - PCA*, e necessita de poucas imagens para criação de uma base de treinamento.
Nas seções seguintes serão descritos o PCA, o método *eigenfaces*, a implementação do sistema de classificação e os resultados obtidos.

2 Desenvolvimento

O classificador E-Faces é um sistema que aceita uma imagem como entrada e gera uma resposta que classifica a imagem como face ou não face.
O E-faces foi implementado através do método *eigenfaces* na linguagem C++, utilizando o compilador GNU G++ e a biblioteca GNU GSL. O E-Faces necessita de um banco de imagens de faces para extração das características. Esse banco de faces é criado e visualizado por uma interface feita em PHP e Bash Scripting. A seção seguinte explicará o PCA, fornecendo a base de entendimento do método *eigenfaces* com a finalidade de tornar claro o modo de funcionamento do classificador.

2.1 PCA

A *Principal Component Analysis*, análise de componentes principais ou expansão de Karhunen-Loève, talvez seja uma das maiores contribuições da álgebra linear e estatística aplicadas [3] [4]. É um método linear que pode ser aplicado na eliminação da redundância ou detecção de padrões em um conjunto de dados.
Quando visto pelo ponto de vista de transformada [5], o seu resultado é uma mudança de base, uma projeção em um novo espaço onde cada componente esteja livre de redundância e esteja expresso em ordem de variância ou contribuição ao conjunto de dados.
Na detecção de padrões pode-se empregar a distância euclidiana como critério de classificação. Na redução do conjunto utiliza-se as componentes que mais contribuem nessa variação do espaço, ou seja, as componentes cujos auto-vetores estejam relacionados com os maiores auto-valores da matriz de covariância do conjunto sendo analisado, desprezando os auto-vetores com baixos auto-valores associados.
O algoritmo do PCA é o seguinte:

1. Organize os dados em uma matriz D onde a primeira linha será formada pelas componentes da primeira amostra, a segunda a linha formada pelas componentes da segunda amostra e a N-ésima linha será formada pelas componentes da N-ésima amostra, como na matriz abaixo:

$$D = \begin{bmatrix} c1a1 & c2a1 & \dots & cMa1 \\ c1a2 & c2a2 & \dots & cMa2 \\ \vdots & \vdots & \ddots & \vdots \\ c1aN & c2aN & \dots & cMaN \end{bmatrix} \tag{1}$$

2. Crie um vetor média E, formado pelas médias de cada coluna:

$$E = \begin{Bmatrix} \mu_1 & \mu_2 & \dots & \mu_M \end{Bmatrix} \tag{2}$$

3. Subtraia de cada item de cada coluna M da matriz D a média μ_M correspondente a coluna da qual o item pertença

4. Calcule a matriz de covariância COVD:

$$COVD = \frac{1}{N-1} DD^T \tag{3}$$

5. Calcule os auto-vetores e auto-valores da matriz COVD gerando um vetor AVAL e uma matriz AVET.

6. Ordene os auto-vetores na matriz AVET de auto-vetores pela ordem crescente dos auto-valores no vetor AVAL correspondentes.

2.2 Eigenfaces

O método *eigenfaces* é um método baseado em aparência segundo [1] que busca as componentes principais de uma distribuição facial, ou os auto-vetores da matriz de covariância de um conjunto de imagens de faces.
O nome *eigenfaces* é atribuído aos auto-vetores (*eigenvectors*) da matriz de covariância das imagens das faces do banco de faces de treinamento por possuírem aspectos de faces.
Seu funcionamento é similar ao funcionamento do PCA, entretanto é utilizada uma leve otimização para reduzir a matriz de covariância, reduzindo o processamento necessário para fazer o cálculo de seus auto-vetores e auto-valores.
Este método possibilita a classificação de imagens a partir do cálculo da distância euclidiana entre a imagem sendo analisada e a imagem sendo analisada projetada no novo espaço. Se o valor da distância euclidiana estiver dentro de uma distância limite, a imagem sendo analisada é considerada face, caso contrário é considerada como não face.
O método *eigenfaces* é interessante por possibilitar, além da classificação, a reconstrução e a compactação de imagens de faces.

2.2.1 Geração das eigenfaces

1. Tendo em mãos um conjunto de M faces,

$$\Gamma = \begin{pmatrix} \Gamma_1 & \Gamma_2 & \dots & \Gamma_M \end{pmatrix}$$

   onde Γ_i é cada face do conjunto

2. Calcule a face média Ψ

$$\Psi = \frac{\sum_{i=1}^{M} \Gamma_i}{M}$$

3. Crie uma matriz com as faces de treino com os pixeis dispostos em linhas e as M faces do conjunto de treinamento dispostas em colunas.

4. Subtraia a imagem média Ψ de cada imagem de Γ, gerando uma nova matriz

$$\Phi = \Gamma - \Psi$$

5. Sendo M menor que a dimensionalidade (largura * altura das imagens de treinamento) das imagens em Φ, calcule a matriz de covariância

$$C = \Phi^T \Phi$$

6. Calcule os auto-vetores ν e auto-valores λ da matriz C

7. Crie a matriz de transformação

$$\mu = \nu\Phi,$$

onde a matriz μ conterá M - 1 auto-vetores significativos. Entretanto ainda é possível realizar a eliminação de alguns desses M - 1 auto-vetores pela ordem de importância de seus correspondentes auto-valores λ, gerando M' auto-vetores escolhidos.

8. Para finalizar, normalize os M' vetores da matriz μ.

A Figura 1 ilustra o conjunto de treinamento, a imagem média e as 41 *eigenfaces* obtidas de 42 imagens de faces.

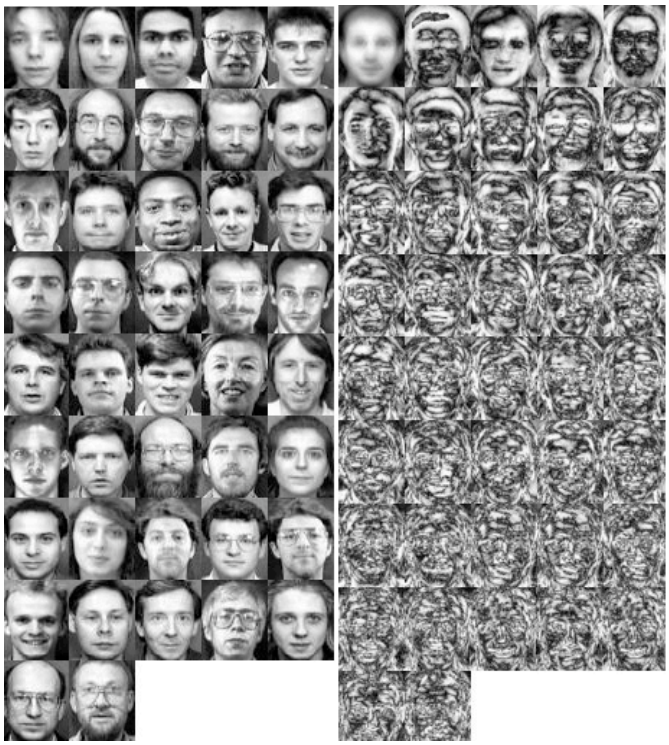

Figura 1: À esquerda: banco de faces para criação das *eigenfaces*. À direita: primeira imagem é a imagem média, as restantes são as *eigenfaces*.

2.2.2 Reconstrução de faces

A reconstrução de faces é feita da seguinte forma:

1. Subtraia a face média Ψ da face Γ a ser reconstruída

$$\Phi_{nova} = \Gamma - \Psi$$

2. Faça a transformação da Φ_{nova}, projetando-a no espaço de faces

$$\omega = \mu^T \Phi_{nova}$$

3. Faça a transformação inversa

$$\Omega = \mu\omega$$

A Figura 2 ilustra a reconstrução de uma imagem.

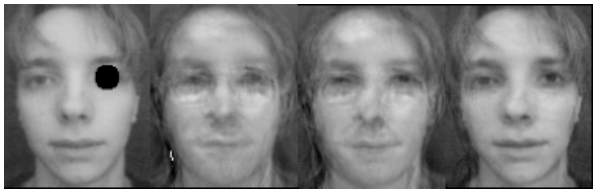

Figura 2: Da esquerda para direita: imagem a ser reconstruída, imagem reconstruída com 20 componentes, imagem reconstruída com 30 componentes e imagem reconstruída com 40 componentes.

2.2.3 Classificação de faces

A Classificação de faces é feita da seguinte forma:

1. Subtraia a face média Ψ da imagem Γ a ser classificada

$$\Phi_{nova} = \Gamma - \Psi$$

2. Faça a transformação da Φ_{nova}, projetando-a no espaço de faces

$$\omega = \mu^T \Phi_{nova}$$

3. Faça a transformação inversa

$$\Omega = \mu\omega$$

4. Calcule a distância euclidiana

$$\epsilon = \|\Phi_{nova} - \Omega\|$$

Se o valor ϵ estiver abaixo de um certo limite θ, considere a imagem Γ como sendo uma imagem de face, caso contrário considere a imagem Γ como sendo uma imagem de não face.

2.3 Interface

Como já dito anteriormente o classificador é um conjunto de programas em C++ que são manipulados por uma interface web em PHP e Bash Scripting. Essa interface permite montar um conjunto de treinamento, gerar e visualizar as *eigenfaces*, classificar imagens e fazer a detecção em uma imagem. Um screenshot da interface desenvolvida pode ser visualizada nas figura 3.

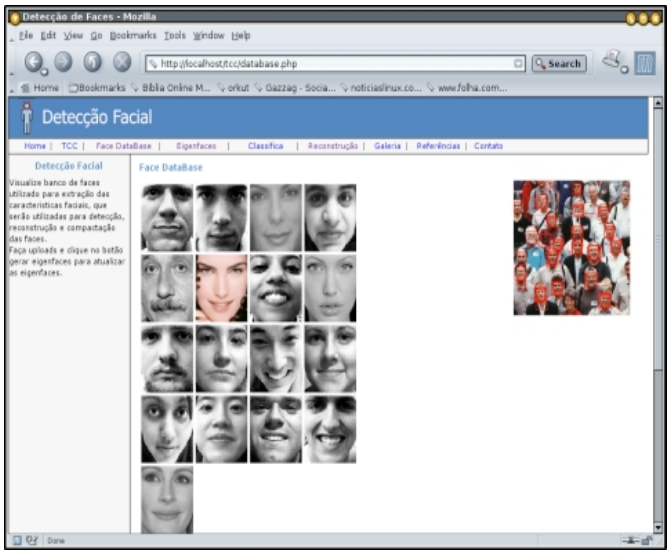

Figura 3: Interface para criação do banco de faces.

3 Resultados

Nossos experimentos nos mostraram que o método foi capaz de classificar corretamente 95% das faces presentes em um banco de imagens de 2429 faces, gerando 5% de detecções falso-negativo.
Foi feita também a avaliação em um banco de imagens de não faces para determinação da quantidade de erros falso-positivo, onde foram corretamente classificadas 93% das não faces presentes em um banco de imagens de 4548 não faces, gerando 7% de detecções falso-positivos.
Por fim, foi feito um teste onde o método foi aplicado exaustivamente para busca de faces frontais em imagens. Esse teste foi realizado da seguinte forma:

1. Faça x = 0 e y = 0.
2. Retire uma sub-janela no ponto (x,y) com a mesma dimensão das faces apresentadas ao classificador durante a etapa de treino.
3. Apresente a sub-janela ao classificador.
4. Se a sub-janela foi classificada como face então marque aquela posição como sendo uma face.
5. Incremente x.
6. Se x maior do que a largura da imagem então faça x = 0 e incremente y.
7. Se y maior do que a altura da imagem então faça y = 0 e reduza a imagem em 20%.
8. Se a dimensão da imagem for maior ou igual a dimensão da sub-janela volte ao passo 1, senão finalize a busca.

A Figura 4 exibe o resultado do teste feito em duas imagens utilizando esse método.

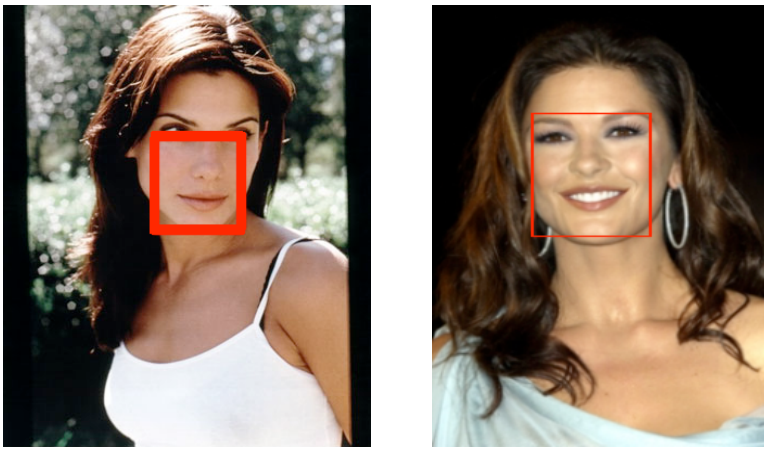

Figura 4: Resultado do teste em duas imagens.

4 Conclusão

Neste trabalho foi possível visualizar a maior fragilidade do método: a determinação do valor θ para o limite da distância euclidiana, pois essa distância determina quando uma imagem é face ou não face. A escolha de um valor θ incorreto pode fazer com que a taxa de erro seja muito elevada resultando na ineficiência do método.
Também foi possível concluir empiricamente que este é um método que apresenta uma boa relação entre o tamanho do conjunto de faces de treinamento com os resultados obtidos.

Referências

[1] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[2] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1991.

[3] Lindsay Smith. A tutorial on principal component analysis. 2002.

[4] Jon Shlens. A tutorial on principal component analysis. 2003.

[5] Rafael C. Gonzalez and Richard E. Woods. *Processamento de Imagens Digitais*. Edgard Blücher, 2000.