FIAP

MBA em Inteligência Artificial & Machine Learning

Tecnologia de Processamento de Imagens

Prof. Msc. Michel Pereira Fernandes

Projeto Final de Avaliação Substitutiva

Proposta: “Avaliação de métodos de reconhecimento facial utilizando visão computacional”

Aluno: Fernando Martins de Oliveira

RM 331114

Turma: 2IA

FIAP

Sumário

FIAP

INTRODUÇÃO

Este projeto tem por objetivo aprofundar o conhecimento nos classificadores de faces presentes na biblioteca OpenCV (versão 3): Eingenfaces, Fisherfaces e Local Binary Patterns Histograms.

CLASSIFICADOR EINGENFACES

Motivação

A capacidade do ser humano de reconhecer faces é notável. Uma pessoa pode reconhecer milhares de faces ao longo de sua vida e identificar rostos familiares de forma rápida mesmo após anos de separação. Embora haja diversas mudanças faciais ocorridas devido a diferentes expressões faciais, distrações como acessórios (óculos, tapa-olho, etc.) ou mudanças de corte de cabelo e barba, o ser humano consegue realizar esta tarefa de forma robusta.

Computadores que reconhecem faces podem ser utilizados em uma grande variedade de problemas: identificação criminal, sistemas de segurança, processamento de imagens e vídeos e interação homem-máquina.

Entretanto, desenvolver sistemas de reconhecimento facial não é uma tarefa trivial devido à complexidade das faces e suas grandes variações. A fim de desenvolver nos computadores a habilidade de reconhecer rostos, foram desenvolvidos modelos de reconhecimento facial tais como Linear Discriminant Analysis, Elastic Bunch Graph Matching e Reconhecimento Facial utilizando Eigenfaces.

Eigenfaces são um conjunto de autovetores de uma matriz de covariância formada por imagens de faces. A ideia de utilizar eigenfaces no reconhecimento de faces surgiu com a técnica desenvolvida por Sirovich e Kirby para representar imagens de rostos de forma eficiente utilizando Análise de Componentes Principais. Turk e Pentland utilizam esta representação para avaliar a distância entre rostos em um “espaço de faces” e assim determinar rostos similares.

Funcionamento

Os passos principais para a modelagem utilizando este método, PCA, são:

1. Dada uma coleção de m imagens de treinamento identificadas, ou seja, tendo uma base de imagens com cada imagem de tamanho matricial de n × o , com alguma identificação, cria-se uma matriz Xij , onde j=1,2,...,m é a quantidade de imagens de treinamento, e i é o tamanho das imagens em formato de vetor, isto é, i = n × o , fazer:

    a. Computar a imagem média

$$M = \sum_{r=1}^{j} X_{ir}, i = 1,2,...,(n \times o)$$

    b. Centralizar os vetores das imagens subtraindo cada um dos vetores pela média dos vetores encontrados.

$$\overline{X} = X - M$$

c. Calcular a matriz de covariância

$$\Lambda = M * M^T$$

d. Computar os k autovetores, vk, da matriz de covariância correspondente aos k maiores autovalores, λk. Como a matriz de covariância é real e simétrica, todos os autovalores e autovetores serão também reais e simétricos [SPERANDIO2003]. Além disso, se essa matriz é de ordem i, então existirá i autovetores associados à i autovalores [BURDEN2003]. Os autovetores são, de certa forma, imagens, que são agrupadas em uma matriz W com k colunas.

$$W_{ik} = \{v_1, v_2, ..., v_k\}$$
$$\lambda_k = \{\lambda_1, \lambda_2, ..., \lambda_n\}$$

e. Projetar cada uma das imagens de treinamento no autoespaço k-dimensional criando um vetor de tamanho reduzido para cada uma das imagens, facilitando a comparação entre os vetores. A projeção é realizada multiplicando cada um dos vetores imagens pelo autoespaço.

$$\hat{X} = W \cdot \overline{X}$$

Os maiores autovalores da matriz de covariância tende não ser fixo. Após ter realizado esses primeiros cálculos na banco de faces, realiza-se o reconhecimento:

1. Dada uma imagem de teste Y, projetá-la no autoespaço, após tê-la centralizada também com aquele mesmo vetor de médias, assim como as de treinamento.

$$\overline{Y} = Y - M$$
$$\hat{Y} = W \cdot \overline{Y}$$

2. Classificá-la com as imagens de treinamento projetadas, fazendo uso de um classificador definido ou, às vezes, pode-se combinar dois ou mais classificadores.

Pontos Positivos

- Reduz a dimensionalidade dos dados de treinamento.
- Reduz a matriz de covariância, o que reduz o processamento necessário para fazer o cálculo de seus auto vetores e autovalores.
- Buscam identificar um pequeno número de características que são relevantes para diferenciar uma face de outras faces.

Pontos Negativos

- Alterações na luminosidade são consideradas componentes de variação entre as faces.

FIAP

Exemplo Prático

No diretório lab.

CLASSIFICADOR FISHERFACES

Motivação

O discriminante linear de Fisher (FLD), também conhecido como análise de discriminantes linear (LDA), foi desenvolvido por R. A. Fisher na década de 1930, porém, apenas recentemente tem sido utilizado para o reconhecimento de objetos. É um método específico à classe, pois, ele trabalha com o uso de "rótulos", isto é, uma vez identificado os rostos dizendo qual face pertence a qual pessoa, os mesmos são agrupados por pessoa, e cada agrupamento desses é conhecido como classe. O método tenta modelar a dispersão dos pontos visando maior confiabilidade para a classificação. O LDA busca otimizar a melhor linha em uma superfície que separa satisfatoriamente as classes [BELHUMEUR1997].

Funcionamento

Inicia-se o algoritmo obtendo as matrizes de dispersão entre classes, interclasse, e dentro das classes, intraclasse. A projeção é feita maximizando a dispersão interclasse e minimizando a intraclasse, formulado pela razão entre as determinantes de ambas as matrizes, com isso diferindo do PCA, que maximiza o espalhamento, dispersão, dos padrões no espaço de características, independente da classe em que esses pertencem [CAMPOS2001] apud [JAIN2000]. As duas medidas citadas, matematicamente são definidas como:

1. matriz de dispersão intraclasses, within class:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{|T_j|} (x_i^j - \mu_j) \cdot (x_i^j - \mu_j)^T ,$$

em que x_i^j é o i-ésimo exemplo da classe j, μ_j é a média da classe j, c é o número de classes, e $|T_j|$ o número de exemplos na classe j;

2. matriz de dispersão interclasses, between class:

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu) \cdot (\mu_j - \mu)^T ,$$

,em que μ representa a média de todas as classes.

A maximização da medida inter-classes e a minimização da intra-classes são obtidas ao maximizar a taxa $\frac{\det(S_b)}{\det(S_x)}$

O espaço de projeção é então encontrado resolvendo a equação $S_b W = \lambda S_w W$, onde W é a matriz com autovetores generalizados associados com λ, que é a matriz diagonal com autovalores. Essas matrizes estão limitadas à ordem c-1, em que c é i número de classes, limitação devido à comparação ser realizada entre duas classes diferentes.

Para identificar uma imagem de teste funciona da mesma forma que o Eigenface. A imagem de teste é projetada e comparada com cada uma das faces de treinamento também projetadas, identificando-a com a de treinamento que mais se aproxima. A comparação, de novo, é feita utilizando um classificador específico ou a combinação de dois ou mais.

Pontos Positivos

- É um bom discriminador de múltiplas faces.
- Quanto mais parecidas forem as imagens de uma pessoa mais próximas deverão estar suas projeções.
-

Pontos Negativos

- Quanto mais parecidas forem as imagens de uma pessoa mais próximas deverão estar suas projeções.

Exemplo Prático

No diretório lab.

CLASSIFICADOR LOCAL BINARY PATTERNS HISTOGRAMS (LBPH)

Motivação

Local Binary Pattern (LBP) é um operador de textura simples, porém eficiente, que rotula os pixels de uma imagem ao limitar a vizinhança de cada pixel e considera o resultado como um número binário.

Foi descrito pela primeira vez em 1994 (LBP) e, desde então, foi considerado um recurso poderoso para a classificação de textura. Ainda, quando o LBP é combinado com os histograms of oriented gradients (HOG), ele melhora o desempenho da detecção consideravelmente em alguns conjuntos de dados.

Usando o LBP combinado com histogramas, podemos representar as imagens do rosto como um vetor de dados simples.

Como o LBP é um descritor visual, ele também pode ser usado para tarefas de reconhecimento facial, como pode ser visto na seguinte explicação.

Funcionamento

As etapas do algoritmo são:

1. **Parâmetros**: o LBPH usa 4 parâmetros:

- **Raio**: o raio é usado para construir o padrão binário circular e representa o raio ao redor do pixel central. Geralmente, é definido como 1.

- **Vizinhos**: o número de pontos de amostra para construir o padrão binário circular local. Tenha em mente que: quanto mais pontos de amostra você incluir, maior será o custo computacional. Geralmente é definido como 8.

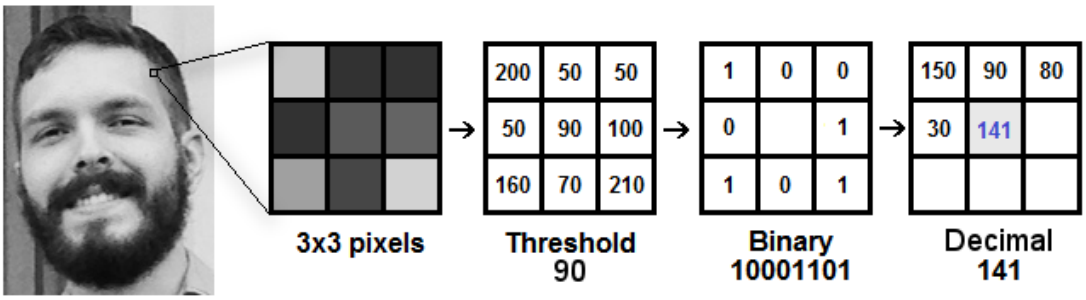
- **Grade X**: o número de células na direção horizontal. Quanto mais células mais fina é a grade e maior é a dimensionalidade do vetor de características resultante. Geralmente é definido como 8.

- **Grade Y**: o número de células na direção vertical. Quanto mais células, mais fina é a grade e maior é a dimensionalidade do vetor de características resultante. Geralmente é definido como 8.

2. **Treinando o Algoritmo**: Primeiro, precisamos treinar o algoritmo. Para fazer isso, precisamos usar um conjunto de dados com as imagens faciais das pessoas que queremos reconhecer. Nós também precisamos definir um ID (pode ser um número ou o nome da pessoa) para cada imagem, então o algoritmo usará essas informações para reconhecer uma imagem de entrada e dar-lhe uma saída. Imagens da mesma pessoa devem ter o mesmo ID. Com o conjunto de treinamento já construído, vejamos os passos computacionais do LBPH.

3. **Aplicando a operação LBP**: O primeiro passo computacional do LBPH é criar uma imagem intermediária que descreva melhor a imagem original, destacando as características faciais. Para fazer isso, o algoritmo usa um conceito de janela deslizante, com base nos parâmetros raio e vizinhos.

A imagem abaixo mostra esse procedimento:

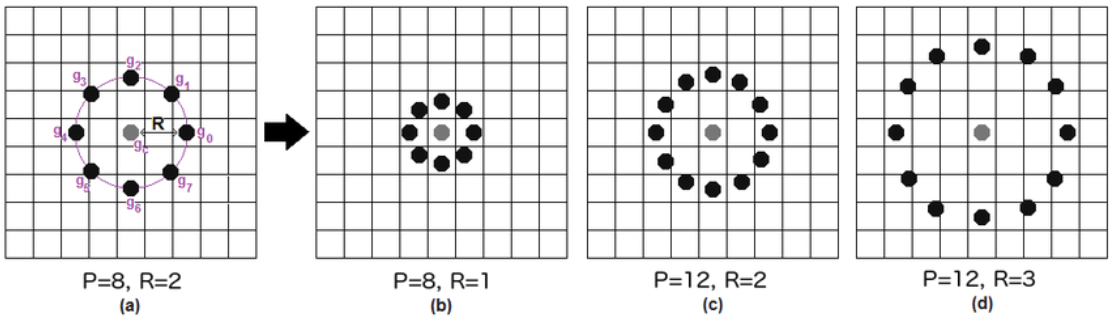


Com base na imagem acima, vamos dividir em várias pequenas etapas para que possamos entender isso facilmente:

- Suponha que tenhamos uma imagem facial em escala de cinza.
- Podemos obter parte desta imagem como uma janela de 3×3 pixels.
- Ele também pode ser representado como uma matriz 3×3 contendo a intensidade de cada pixel (0 ~ 255).
- Então, precisamos tomar o valor central da matriz para ser usado como limiar.
- Esse valor será usado para definir os novos valores dos 8 vizinhos.
- Para cada vizinho do valor central (limiar), estabelecemos um novo valor binário.

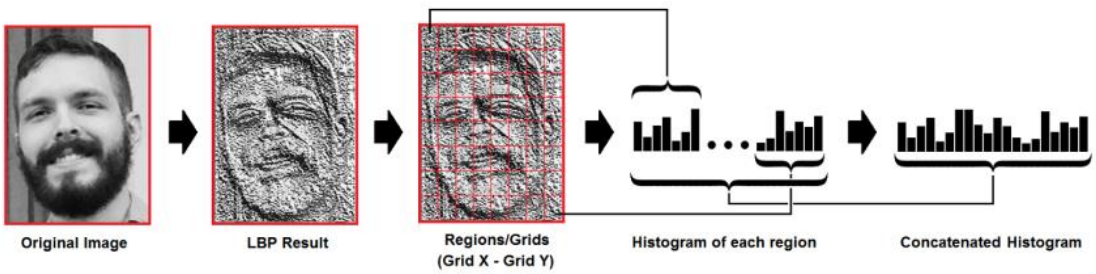
- Definimos 1 para valores iguais ou superiores ao limiar e 0 para valores inferiores ao limiar.
- Agora, a matriz conterá apenas valores binários (ignorando o valor central).
- Precisamos concatenar cada valor binário de cada posição da matriz linha por linha para um novo valor binário (por exemplo, 10001101). **Nota**: alguns autores usam outras abordagens para concatenar os valores binários (por exemplo, no sentido horário), mas o resultado final será o mesmo.
- Então, convertemos esse valor binário para um valor decimal e colocamos ele na posição central da matriz, que é realmente um pixel da nova imagem.
- No final deste procedimento (chamado LBP), temos uma nova imagem que representa melhor as características da imagem original.

Nota: O procedimento LBP foi expandido para usar um número diferente de raio e vizinhos, é chamado de Circular LBP.



Isso pode ser feito usando a interpolação bilinear. Se algum ponto de dados estiver entre os pixels, ele usa os valores dos 4 pixels mais próximos (2×2) para estimar o valor do novo ponto de dados.

4. **Extraindo os histogramas**: agora, usando a imagem gerada no último passo, podemos usar os parâmetros Grade X e Grade Y para dividir a imagem em múltiplas grades, como pode ser visto na imagem a seguir:



Com base na imagem acima, podemos extrair o histograma de cada região da seguinte maneira:

- Como temos uma imagem em escala de cinza, cada histograma (de cada grade) conterá apenas 256 posições (0 ~ 255) que representam as ocorrências de cada intensidade de pixel.
- Então, precisamos concatenar cada histograma para criar um histograma novo e maior. Supondo que tenhamos redes 8×8, teremos 8x8x256 = 16.384 posições no

histograma final. O histograma final representa as características da imagem original da imagem.

5. **Realizando o reconhecimento facial**: nesta etapa, o algoritmo já está treinado. Cada histograma criado é usado para representar cada imagem do conjunto de dados de treinamento. Assim, dada uma imagem de entrada, nós executamos as etapas novamente para esta nova imagem e criamos um histograma que representa a imagem.

Então, para encontrar a imagem que corresponde à imagem de entrada, precisamos comparar dois histogramas e devolver a imagem com o histograma mais próximo.

Podemos usar várias abordagens para comparar os histogramas (calcular a distância entre dois histogramas), por exemplo: distância euclidiana, qui-quadrado, valor absoluto, etc. Neste exemplo, podemos usar a distância euclidiana (que é bastante conhecida) baseada na seguinte fórmula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

Portanto, a saída do algoritmo é o ID da imagem com base no histograma mais próximo. O algoritmo também deve retornar a distância calculada, que pode ser usada como medida de "confiança". **Nota**: não se deixe enganar com o nome da "confiança", pois as confianças mais baixas são melhores porque significa que a distância entre os dois histogramas é mais próxima. Podemos usar um limite e a "confiança" para estimar automaticamente se o algoritmo reconheceu corretamente a imagem. Podemos assumir que a pessoa foi reconhecida com sucesso se a confiança for menor do que um limiar definido.

Pontos Positivos

- LBPH é um dos algoritmos de reconhecimento facial mais fáceis de compreender.
- Pode representar características locais nas imagens.
- É possível obter excelentes resultados (principalmente em um ambiente controlado).
- É robusto contra transformações monotônicas em escala de cinza.
- É o mais robusto entre os 3 fornecidos apresentados nesse trabalho

Pontos Negativos

- A precisão é dependente de um número predefinido de características ou uma taxa predefinida de reconhecimento.

Exemplo Prático

No diretório lab.

CONCLUSÃO

O algoritmo LBPH apresentou o melhor resultado para o reconhecimento facial.

O algoritmo Fisherfaces eu não consegui executar, mas pela literatura ele apresentaria um resultado intermediário em relação aos outros dois.

Já o algoritmo Eigenfaces não se mostrou eficiente, pois não conseguiu reconhecer nenhuma vez o rosto utilizado para os testes.

FIAP

REFERÊNCIA BIBLIOGRÁFICA

https://www.lcg.ufrj.br/marroquim/courses/cos756/trabalhos/2013/abel-nascimento/abel-nascimento-report.pdf

https://www.aedb.br/seget/arquivos/artigos06/916_Copia%20de%20Artigo%20Comparativo%20Facial.pdf

http://www.pucrs.br/facin-prov/wp-content/uploads/sites/19/2016/03/tr045.pdf

http://www.sinfic.pt/SinficWeb/displayconteudo.do2?numero=44666

https://www.aedb.br/seget/arquivos/artigos06/916_Copia%20de%20Artigo%20Comparativo%20Facial.pdf

https://updatedcode.wordpress.com/2017/11/26/reconhecimento-facial-como-funciona-o-lbph/