UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Engenharia e Ciências de Guaratinguetá

EDUARDO MARTINS FAVERI

Estudo da eficácia dos algoritmos de reconhecimento facial

Guaratinguetá
2023

Eduardo Martins Faveri

Estudo da eficácia dos algoritmos de reconhecimento facial

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia elétrica da Faculdade de Engenharia e Ciências do Campus de Guaratinguetá, Universidade Estatual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia elétrica .

Orientador: Profº Dr. Daniel Julien B. da S. Sampaio

Guaratinguetá
2023

F273e

Faveri, Eduardo Martins
Estudo da eficácia dos algoritmos de reconhecimento facial / Eduardo Martins Faveri - Guaratinguetá, 2023.
31 f : il.
Bibliografia: f. 31

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia e Ciências de Guaratinguetá, 2023.
Orientador: Prof. Dr. Daniel Julien B. da S. Sampaio

1. Algoritmos. 2. Reconhecimento facial (Computação).
3. Biometria. I. Título.

CDU 519.712

Luciana Máximo
Bibliotecária/CRB-8 3595

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE GUARATINGUETÁ

EDUARDO MARTINS FAVERI

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE **"GRADUANDO EM ENGENHARIA ELÉTRICA "**

APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Profº Dr. DANIEL JULIEN B. DA S. SAMPAIO
Coordenador

BANCA EXAMINADORA:

Profº Dr. Daniel Julien B. da S. Sampaio
Orientador/UNESP-FEG

Profº Dr. Leonardo Mesquita
UNESP-FEG

Profº Dr. Teófilo Miguel de Souza
UNESP-FEG

Janeiro , 2023

AGRADECIMENTOS

Agradeço a minha família, ao meu orientador e a todas as pessoas que me apoiaram de alguma forma na minha jornada.

“O ser humano é aquilo que a educação faz dele.“
(Kant)

RESUMO

A monografia tem como objetivo avaliar a eficácia dos três algoritmos (EigenFaces, FisherFaces e Local Binary Pattern Histogram) de reconhecimento facial presentes na biblioteca OpenCV. Realizado estudo de caso utilizando um banco de dados somente de faces, com variações de expressões faciais e luminosidade; foi treinado um grupo de imagens e outro grupo foi usado para testes. Foram estabelecidos critérios para tal avaliação, como a acurácia (verdadeiros positivos e negativos sob número de testes) e tempo de execução. Notou-se que o algoritmo Local Binary Pattern Histogram é o mais eficaz entre os três, contudo nenhum deles é ideal para aplicações reais.

PALAVRAS-CHAVE: OpenCV. EigenFaces. FisherFaces. LBPH.

ABSTRACT

The monograph aims to evaluate the effectiveness of the three facial recognition algorithms (EigenFaces, FisherFaces and Local Binary Pattern Histogram) present in the OpenCV library. A case study was carried out using a database of faces only, with variations in facial expressions and luminosity; one group of images was trained and another group was used for testing. Criteria for such an evaluation were established, such as accuracy (true positives and negatives under the number of tests) and execution time. It was noted that the Local Binary Pattern Histogram algorithm is the most effective among the three, however none of them is ideal for real applications.

KEYWORDS: OpenCV. EigenFaces. FisherFaces. LBPH.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OpenCV	Open Source Computer Vision Library
PCA	Principal Components Analysis
LDA	Linear Discriminant Analysis
LBPH	Local Binary Pattern Histogram
GTdb	Georgia Tech face database

LISTA DE SÍMBOLOS

Γ	Foto com nxn pixels que pode ser tratada como uma matriz de valores entre 0 e 255 em escala de cinza.
Ψ	Vetor formado pelos valores médios de cada pixel de todas as fotos utilizadas para treinamento.
Φ	Matriz que contém a diferença entre os valores dos Γ em relação a média Ψ.
μ_k	Vetor que corresponde a um EigenFace.
ω	Peso calculado para uma face.
Ω	Vetor de pesos para uma imagem.
ϵ	Distância euclidiana.
W_{opt}	Projeção otimizada para encontrar os pesos no método FisherFaces.
w_k	Vetor que corresponde a um FisherFace.
S_B	Matriz de dispersão interclasses.
S_W	Matriz de dispersão intraclasses.
$LBP_{P,R}$	Notação para LBPH com P amostras e raio R.
$LBP^{U2}_{P,R}$	Notação para LBPH com P amostras e raio R, utilizando *uniform patterns*.

SUMÁRIO

1 INTRODUÇÃO

Todos os dias milhões de autenticações são realizadas através de informações biométricas, tais como: impressão digital, íris, palma da mão, face etc. Segundo Jain, Flynn e Ross (2010), exemplo de aplicações como o compartilhamento de recursos computacionais, acesso a instalações nucleares, realizar transações financeiras remotas ou embarcar um voo comercial, necessitam de sistemas para gerenciamento de identidade com a determinação precisa da identidade de um indivíduo. O que reforça a relevância da biometria na sociedade moderna.

De acordo com Li e Jain (2011), a ampla disponibilidade de computadores e sistemas embarcados, poderosos e de baixo custo, criou um interesse enorme em processamento automático de imagens digitais em uma variedade de aplicações, incluindo autenticação biométrica, vigilância e interação homem-computador; consequentemente pesquisa e desenvolvimento em reconhecimento facial automático surgiria naturalmente.

Para Li e Jain (2011), reconhecimento facial tem várias vantagens em relação a outras modalidades de biometrias, como impressão digital e íris. Além de ser natural e não intrusivo, a mais importante vantagem é que pode ser capturado a distância.

Existem diversos métodos e implementações para reconhecimento facial, um deles é o OpenCV (Open Source Computer Vision Library), uma biblioteca de código aberto para visão computacional e aprendizagem de máquina com mais de 2500 algoritmos OpenCV (2008), de grande popularidade e vasta documentação.

O presente trabalho visa estudar o funcionamento dos métodos de reconhecimento presentes no OpenCV, implementar um modelo e verificar a acurácia dos algoritmos.

1.1 OBJETIVOS

O presente trabalho tem como objetivo avaliar a eficácia dos três algoritmos (EigenFaces, FisherFaces e Local Binary Pattern Histogram) de reconhecimento facial presentes na biblioteca OpenCV.

1.2 MÉTODOS

Utilizando da linguagem Python e de um banco de imagens contendo 750 fotos de 50 pessoas, foi implementado diferentes algoritmos da biblioteca. Estabeleceu-se critérios para avaliação da acurácia. Desenvolvido o código e com os critérios definidos, analisou-se a taxa de acerto no reconhecimento facial de cada um dos três algoritmos.

1.3 ORGANIZAÇÃO DO TRABALHO

A introdução, objetivos, métodos e a organização do trabalho foram apresentadas no capítulo 1.

No capítulo 2, definiu-se na seção 2.1 o que é biometria, suas propriedades e principais aplicações; também foi descrito o processo para reconhecimento facial. Nas seções 2.2 e 2.3 explicou-se o que é a biblioteca OpenCV e quais são os algoritmos.

Nas subseções 2.3.1, 2.3.2 e 2.3.3 descreveu-se o funcionamento dos algoritmos e suas respectivas vantagens.

Detalhou-se no capítulo 3 o desenvolvimento e os critérios estabelecidos para realização dos testes.

Já no capítulo 4, foi apresentado os resultados obtidos e feito uma análise da eficácia de cada algoritmo sob diferentes valores de *threshold*. Por último foi feito a conclusão no capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, na seção 2.1 definiu-se o que é biometria, também foi explicado o processo e quais são as possíveis aplicações dos algoritmos de reconhecimento facial.

Na seção 2.2 foi dado uma breve descrição da biblioteca OPENCV. A seção 2.3 foi dividida nas subseções 2.3.1, 2.3.2 e 2.3.3, que explicam e descrevem os algoritmos EigenFaces, FisherFaces e LBPH respectivamente.

2.1 BIOMETRIA

Conforme Phillips, McCabe e Chellappa (1998), pode-se definir biometria como: "Uma biometria é um dado observado de um humano, que permite a identidade daquela pessoa ser determinada' e para biometria ser eficiente, é necessário que tenha as propriedades da tabela 1:

Tabela 1 – Propriedades para biometria ser eficiente

Propriedades	
Universalidade	Todos os membros de uma população sendo identificada devem possuir a biometria.
Unicidade	Assinatura biométrica deve ser diferente para todos os membros da população.
Invariância	Assinatura biométrica deve ser invariante sob as condições que será coletada.
Resistência	A biometria deve ser resistente a potenciais contra medidas.

Fonte: Phillips, McCabe e Chellappa (1998).

O uso de dados biométricos para autenticação se tornou de uso diário, seja no desbloqueio de um celular ou no uso de um caixa eletrônico, estamos a todo momento usando informações biométricas.

Uma opção que recentemente começou a ser explorada comercialmente é a autenticação através do reconhecimento facial. Uma lista de possíveis usos foi descrita em Zhao et al. (2003), conforme tabela 2:

De acordo com Li e Jain (2011), o processo de reconhecimento facial pode ser separado nas seguintes etapas:

1. Detecção da face: etapa em que ocorre a detecção do rosto em relação a toda imagem.
2. Normalização: é necessário que as imagens estejam em tamanhos e escala de cores apropriadas.
3. Extração da característica: realizada na imagem normalizada, ocorre a separação da face do restante da imagem.
4. Identificação: a face extraída é comparada com outras faces no banco de imagens.

Dado o processo de reconhecimento facial, o presente trabalho foca na última etapa, a identificação.

2.2 OPENCV

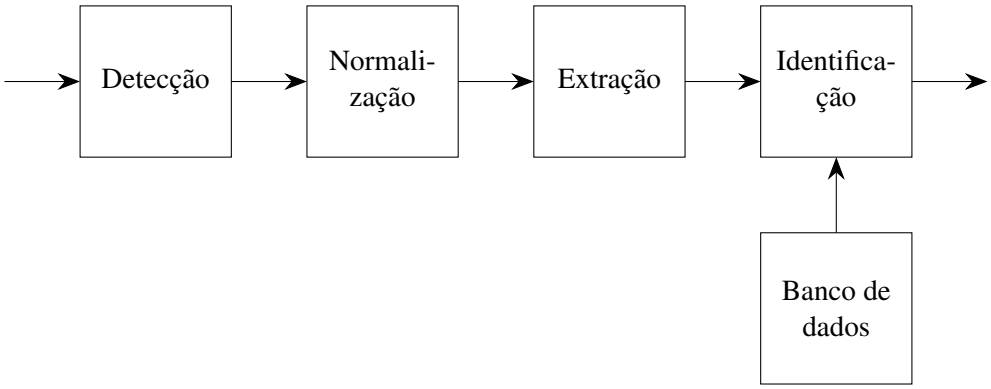
OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto para visão computacional e aprendizagem de máquina, originalmente desenvolvida pela Intel, em 2000.

Tabela 2 – Aplicações para reconhecimento facial

Áreas	Aplicações específicas
Entretenimento	Jogos eletrônicos, realidade virtual e programas de treinamento. Interação homem-robô e interação homem-computador.
Cartões inteligentes	Habilitação de motorista e programas governamentais . Imigração, identidade nacional, passaporte e registro de voto. Fraudes contra a previdência.
Segurança da informação	Controle parental, acesso a dispositivos pessoais e acesso ao computador. Segurança de aplicações (*softwares*), segurança de banco de dados e encriptação de arquivos. Segurança da intranet, acesso a internet e registros médicos. Terminal de negociação seguro.
Aplicação da lei e vigilância	Vigilância avançada de vídeos e controle de CCTV. Controle de portais e análise pós-evento. Furto em lojas, rastreamento e investigação de suspeitos.

Fonte: Zhao et al. (2003).

Figura 1 – Processo de reconhecimento facial



Fonte: Adaptado de Li e Jain (2011)

2.3 ALGORITMOS

A biblioteca OpenCV (2008) disponibiliza três algoritmos para reconhecimento facial, são eles:

- EigenFaces
- FisherFaces
- Local Binary Pattern Histogram (LBPH)

Nas próximas seções, será explicado o funcionamento de cada um deles.

2.3.1 EigenFaces

O algoritmo EigenFaces utiliza *Principal Component Analysis* (PCA), que de acordo com Jolliffe (2002), têm como ideia central, a redução da dimensionalidade de um conjunto de dados, enquanto mantém o máximo possível da variação dos dados.

Considere uma imagem de 256x256 pixels em escala de cinza (valores podem variar somente de 0 a 255), pode-se trata-la como uma matriz e a partir disso, transforma-la em um vetor que estará em um espaço dimensional de tamanho 65536, portanto, um conjunto de imagens mapearia uma coleção de pontos em um espaço imenso. O objetivo do uso de PCA é reduzir tal dimensionalidade e facilitar a identificação de pessoas com somente as característica mais importantes.

Turk e Pentland (1991a) afirma que as imagens de faces, tendo um aspecto geral similar, não irão ser distribuídas aleatoriamente em um espaço imenso e portanto podem ser descritas por um subespaço relativamente pequeno. A principal ideia da análise dos componentes principais é encontrar os vetores que melhor descrevem a distribuição dos rostos dentro do espaço das imagens. Esses vetores definem um subespaço de imagens de faces, ao qual é chamado "espaço de faces".

Observa-se que Belhumeur, Hespanha e Kriegman (1997) afirmam que ao escolher a projeção que maximiza a variância, PCA retém variações indesejadas devido à iluminação e expressão facial.

Baseado no trabalho de Turk e Pentland (1991a), pode-se dividir o algoritmo EigenFaces para o treinamento nas seguintes etapas:

1. Obtenção das imagens para o treinamento.

   Necessário um conjunto de imagens de faces, por exemplo: $\mathbf{\Gamma_1}$, $\mathbf{\Gamma_2}$, $\mathbf{\Gamma_3}$, ... , $\mathbf{\Gamma_M}$.

2. Transformação das imagens em vetores.

   Uma imagem pode ser considerada como uma matriz Γ_{nxn}, com os valores variando de 0 a 255 em escala de cinza.

$$\mathbf{\Gamma} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nn} \end{bmatrix}_{nxn}$$

   É transformado a matriz em um vetor, de tal forma que:

$$\mathbf{\Gamma_1} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{nn} \end{bmatrix}_{n^2x1}$$

   Procedimento é realizado para todas as imagens $\mathbf{\Gamma_1}$, $\mathbf{\Gamma_2}$, $\mathbf{\Gamma_3}$, ... , $\mathbf{\Gamma_M}$, obtém-se uma matriz n^2xM com todos os vetores.

3. Cálculo da média dos pixels.

   O passo seguinte é encontrar a face média $\mathbf{\Psi}$.

$$\mathbf{\Psi} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{\Gamma_n} \tag{1}$$

4. Cálculo da diferença entre cada um dos vetores e a imagem média.

   Obtido a face média $\mathbf{\Psi}$, é calculado a diferença em relação a média para cada elemento dos vetores.

$$\mathbf{\Phi_i} = \mathbf{\Gamma_i} - \mathbf{\Psi} \tag{2}$$

   Resulta em uma matriz n^2xM com os vetores $\mathbf{\Phi_1}$, $\mathbf{\Phi_2}$, $\mathbf{\Phi_3}$, ... , $\mathbf{\Phi_M}$.

5. Determinação da matriz de covariância, dos autovalores e autovetores.

   Para aplicação do PCA, é necessário o cálculo da matriz de covariância.

$$\begin{aligned} C &= \frac{1}{M}\sum_{n=1}^{M} \mathbf{\Phi_n}\mathbf{\Phi_n^T} \\ &= AA^T \end{aligned} \tag{3}$$

   Onde a matriz A = [$\mathbf{\Phi_1}$, $\mathbf{\Phi_2}$, $\mathbf{\Phi_3}$, ... , $\mathbf{\Phi_M}$]. A matriz C é n^2xn^2, o que seria computacionalmente inviável segundo Turk e Pentland (1991a). Portanto o algoritmo calcula A^TA, que resulta em uma matriz MxM, muito menor do que n^2xn^2.

   Após o cálculo da matriz de covariância, encontra-se os M autovalores e autovetores.

   Autovalores: λ_1, λ_2, λ_3, ... , λ_M.

   Autovetores: $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$, ... , $\mathbf{v_M}$.

6. Cálculo dos EigenFaces.

   Os autovalores são classificados de acordo com seu valor (do maior para o menor). Em seguida, os autovetores são multiplicados pelos elementos da matriz A para conseguir os "EigenFaces":

$$\mathbf{u_l} = \sum_{k=1}^{M} \mathbf{v_{lk}}\mathbf{\Phi_k} \quad \text{l = 1, 2, 3, ..., M.} \tag{4}$$

   Exemplo para o EigenFace $\mathbf{u_1}$:

$$\mathbf{u_1} = [\mathbf{v_{11}}, \mathbf{v_{12}}, \mathbf{v_{13}}, \mathbf{v_{14}}, ..., \mathbf{v_{1M}}]_{1xM} \begin{bmatrix} \mathbf{\Phi_1} \\ \mathbf{\Phi_2} \\ \mathbf{\Phi_3} \\ \vdots \\ \mathbf{\Phi_M} \end{bmatrix}_{Mxn^2} \tag{5}$$

   Os EigenFaces são a combinação linear dos autovetores $\mathbf{v_i}$ com a matriz $\mathbf{A}$.

   É definido quantos EigenFaces são necessários de acordo com os valores de autovalores (escolhe-se os maiores, por isso a classificação) para proporcionar um resultado aceitável.

São chamados de EigenFaces devido ao fato de parecerem-se com faces, se convertidos para uma matriz de dimensão *nxn*.

7. Cálculo dos pesos.

   Utilizando os EigenFaces e a matriz A são calculados os pesos $\omega_{\mathbf{k}}$, que serão efetivamente usados no reconhecimento facial:

$$\omega_{\mathbf{k}} = \mathbf{u}_{\mathbf{k}}^{\mathbf{T}} \mathbf{\Phi}_{\mathbf{i}} \tag{6}$$

   Para cada imagem treinada é obtido um conjunto de pesos (*weights*) de dimensão M. Para o caso em que são usados todos os EigenFaces, têm-se:

$$\mathbf{\Omega} = [\omega_{\mathbf{1}}, \omega_{\mathbf{2}}, \omega_{\mathbf{3}}, ..., \omega_{\mathbf{M}}] \tag{7}$$

   Por último estabelece-se classes, que são as médias dos pesos para indivíduos que tem mais de uma foto sendo treinada. Caso haja somente uma foto, a classe terá os pesos da respectiva imagem.

$$\mathbf{\Omega_k} = [\omega_{\mathbf{1}}, \omega_{\mathbf{2}}, \omega_{\mathbf{3}}, ..., \omega_{\mathbf{M}}] \tag{8}$$

   Pontua-se que é possível reconstruir as imagens seguindo a equação:

$$\mathbf{\Gamma_i} = \mathbf{\Psi} + [\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, ..., \mathbf{u_M}] \begin{bmatrix} \omega_{\mathbf{1}} \\ \omega_{\mathbf{2}} \\ \omega_{\mathbf{3}} \\ \vdots \\ \omega_{\mathbf{M}} \end{bmatrix} \tag{9}$$

Para o reconhecimento são as seguintes etapas:

1. Transformação da imagem em vetor.

   É transformado a imagem da face em um vetor assim como foi feito no treinamento:

$$\mathbf{\Gamma_{teste}} = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{nn} \end{bmatrix}_{n^2 x 1}$$

2. Cálculo da variância do vetor.

É calculado a diferença em relação a média para cada elemento do vetor.

$$\mathbf{\Phi_{teste}} = \mathbf{\Gamma_{teste}} - \mathbf{\Psi} \tag{10}$$

3. Cálculo dos pesos da imagem teste.

   São calculados os pesos para a imagem teste:

$$\omega_{\mathbf{k}} = \mathbf{u_k^T \Phi_{teste}} \tag{11}$$

   Sendo $\mathbf{\Omega_{teste}} = [\omega_{\mathbf{1}}, \omega_{\mathbf{2}}, \omega_{\mathbf{3}}, ..., \omega_{\mathbf{M}}]$, o conjunto de pesos para a imagem teste.

4. Determinação se a imagem teste pertence a face treinada.

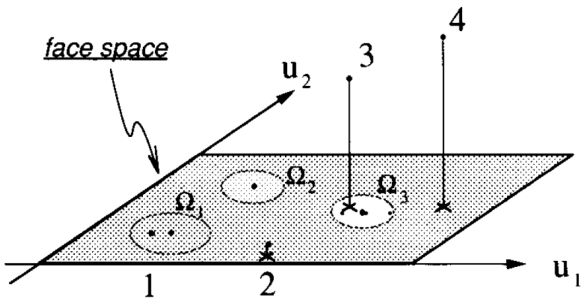
   É estabelecido um *threshold* e calculado a distância euclidiana da imagem teste para as classes do treinamento.

$$\epsilon_{\mathbf{k}}{}^2 = ||\mathbf{\Omega_{teste}} - \mathbf{\Omega_k}||^2 \tag{12}$$

   Se a distância estiver dentro do *threshold*, será considerado que a face testada corresponde a uma das faces da base de treinamento.

   Na Figura 2 tem-se a representação simplificada de um espaço de faces com dois EigenFaces e três classes (Ω_1, Ω_2 e Ω_3), cada uma delas sendo um indivíduo distinto.

Figura 2 – Representação simplificada de um espaço de faces



Fonte: Turk e Pentland (1991b).

2.3.2 FisherFaces

As etapas do algoritmo FisherFaces são similidares ao EigenFaces, tanto para o treinamento quanto na identificação. Entretanto o método não utiliza somente PCA, faz uso também de Linear Discriminant Analysis (LDA).

Sabendo que classes são conjuntos de imagens de um mesmo indivíduo, de acordo com Belhumeur, Hespanha e Kriegman (1997) a aplicação de PCA não maximiza somente a dispersão interclasses mas também a intraclasse, tornando-o menos eficiente.

O método FisherFaces (LDA) busca encontrar uma transformação otimizada W_{opt} (conjunto de autovetores e autovalores) que reduza a dimensionalidade do espaço enquanto maximiza a dispersão interclasses e diminua a dispersão intraclasse Belhumeur, Hespanha e Kriegman (1997).

Belhumeur, Hespanha e Kriegman (1997) afirma que apesar do PCA alcançar uma maior dispersão total, LDA obtém uma maior dispersão interclasses, consequentemente, a identificação é simplificada.

Dado um conjunto de imagens $[x_1, x_2, ..., x_n]$ que pertencem as classes $[X_1, X_2, ..., X_c]$, o algoritmo procura encontrar a projeção otimizada através da seguinte equação:

$$\begin{aligned} \mathbf{W_{opt}} &= \mathbf{argmax}\frac{|\mathbf{W^T S_B W}|}{|\mathbf{W^T S_W W}|} \\ &= [\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_m}] \end{aligned} \tag{13}$$

Onde o conjunto de autovetores $[\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_m}]$, são chamados de FisherFaces, equivalentes aos Eigenfaces $[\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, ..., \mathbf{u_M}]$. As matrizes de dispersão interclasses e intraclasses, $\mathbf{S_B}$ e $\mathbf{S_W}$ respectivamente, são definidas como:

$$\mathbf{S_B} = \sum_{\mathbf{i=1}}^{\mathbf{c}} \mathbf{N_i}(\mu_\mathbf{i} - \mu)(\mu_\mathbf{i} - \mu)^\mathbf{T} \tag{14}$$

$$\mathbf{S_W} = \sum_{\mathbf{i=1}}^{\mathbf{c}} \sum_{\mathbf{x_k} \in \mathbf{X_i}} (\mathbf{x_i} - \mu_\mathbf{i})(\mathbf{x_i} - \mu_\mathbf{i})^\mathbf{T} \tag{15}$$

Onde $\mu_\mathbf{i}$ é a imagem média da classe X_i.

2.3.3 Local Binary Pattern Histogram (LBPH)

De acordo com Pietikäinen (2010) LBP é um operador de textura simples e eficiente, o qual define o valor de um pixel em particular, baseado no limiar de cada pixel vizinho e considera o resultado como um número binário.

Ainda para Pietikäinen (2010), a mais importante propriedade do operador LBP em aplicações reais é a robustez para mudanças em escalas monotônicas de cinza causadas, por exemplo, por variações de iluminação. Outra importante propriedade é a simplicidade computacional, o que torna possível analisar imagens em desafiadoras configurações de tempo-real.

Pode-se dividir o algoritmo LBPH para o treinamento nas seguintes etapas:

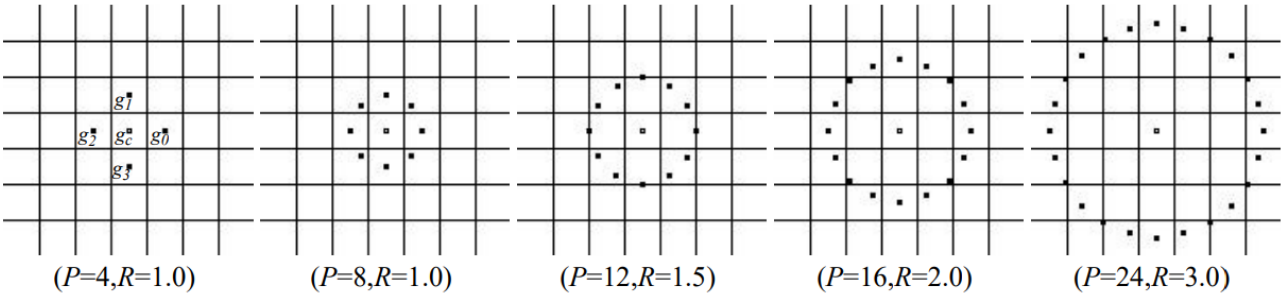
1. Converter a imagem para escala de cinza.

   Este passo está presente em todos os algoritmos contudo é especialmente necessário para o LBPH. Todos os possíveis valores estarão em uma escala de 0 a 255.

2. Cálculo dos valores dos pixels.

   Na Figura 3 têm-se um pixel central e diferentes valores para P (número de pontos) e R (raio, medido em número de pixels).

Figura 3 – Procedimento de comparação dos valores para pixel central em (8,1).

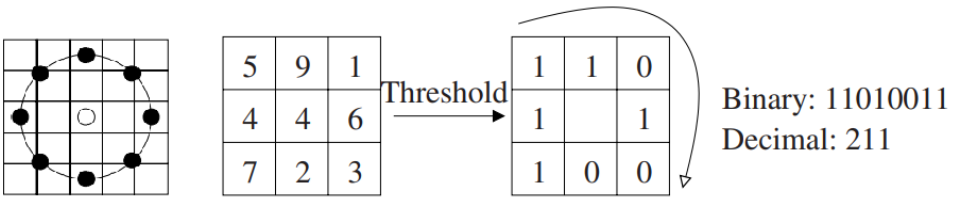


Fonte: Ojala, Pietikainen e Maenpaa (2002).

É definido o número de amostras P e o raio R da amostragem, configurações usuais são (8,1) e (16,2).

Foi indicado no item 1 que é especialmente importante converter para escala de cinza porque é comparado o valor do pixel central com os demais.

Figura 4 – Vizinhos circularmente simétricos para diferentes (P,R)



Fonte: Ahonen, Hadid e Pietikäinen (2004).

Na Figura 4, usa-se a configuração (8,1) e o pixel central tem valor 4, compara-se o valor central com os demais vizinhos, caso o pixel sendo avaliado seja maior ou igual ao pixel central, atribui-se 1; caso contrário, atribui-se 0.

A equação seguinte descreve formalmente o procedimento:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \tag{16}$$

Sabendo que a função s(x) é dada por:

$$s(x) = \begin{cases} 1, & \text{se x} \geq 0 \\ 0 & \text{se x} < 0 \end{cases} \tag{17}$$

Em seguida, rotacionando no sentido horário da esquerda para direita obtêm-se um número binário.

Descobriu-se a existência de um padrão binário, Ojala, Pietikainen e Maenpaa (2002) verificaram que padrões binários para quando ocorrem somente duas transições bit a bit correspondem por pouco menos de 90% dos casos para (8,1) e cerca de 70% para (16,2). Deu-se o nome para estes padrões de *uniform patterns*.

Um exemplo seriam os números binários 00000000 (0 transições), 01110000 (2 transições) e 11001111 (2 transições) que são uniformes, já os números binários 1001001 (4 transições) e 01010010 (6 transições) não são.

Anotações usuais são $LBP_{P,R}$ e $LBP^{U2}_{P,R}$ para quando se está usando *uniform patterns*.

Estes padrões facilitam na construção dos histogramas por que diminuem a escala, pode-se estabelecer marcações específicas para cada binário *uniform patterns* e um geral para não *uniform patterns*. Pietikäinen (2010) afirma que quando usando (8, R); há um total de 256 padrões, 58 deles são uniformes, o que produziria somente 59 marcações (58 uniformes e 1 para os restantes).

3. Definição do número de regiões.

Figura 5 – Exemplo de uma imagem dividida em janelas 7x7.

Fonte: Ahonen, Hadid e Pietikäinen (2004).

De acordo com Ahonen, Hadid e Pietikäinen (2004), para eficiência na representação da face, informações sobre a distribuição de micro padrões locais devem estar contidos nos histogramas. Para esse propósito, a imagem é dividida em regiões R_0, R_2, ... , R_{m-1}.

Ahonen, Hadid e Pietikäinen (2004) destaca que um grande número de pequenas regiões produz longos vetores de característica causando alto consumo de memória e lenta classificação, enquanto o uso de grandes regiões causa maior perda de informação espacial. É necessário uma escolha adequada no número de regiões para conseguir bons resultados.

4. Cálculo dos histogramas.

   Por último são calculados os histogramas para cada região e concatenados em um vetor para formar o vetor característico da imagem.

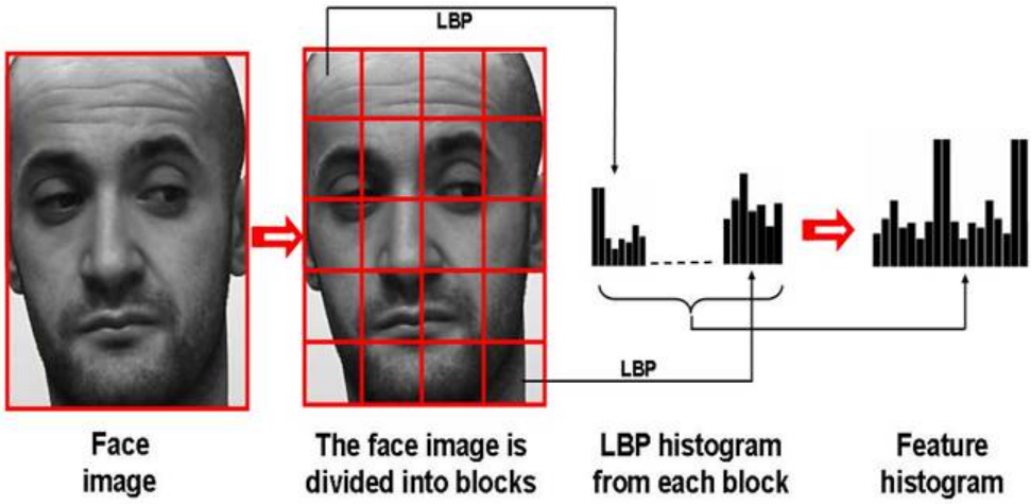
Para o reconhecimento são os seguintes passos:

Os passos de 1 a 3 são realizados da mesma maneira que foi feito no treinamento.

1. Cálculo dos valores dos pixels.
2. Definição do número de regiões.
3. Cálculo dos histogramas.
4. Uso de medições de distância.

   É utilizado o classificador K-nearest neighbour (KNN) e medições de distância para identificação das faces, exemplo de possíveis medições são:

Figura 6 – Procedimento para construção dos histogramas.



Fonte: Pietikäinen (2010).

Interseção do Histograma:

$$D((\mathbf{S},\mathbf{M}) = \sum_i min(S_i, M_i) \tag{18}$$

Máxima verossimilhança:

$$L(\mathbf{S},\mathbf{M}) = -\sum_i S_i log M_i \tag{19}$$

Chi-quadrado:

$$\chi^2(\mathbf{S},\mathbf{M}) = \sum_i \frac{(S_i - M_i)^2}{S_i + M_i} \tag{20}$$

3 DESENVOLVIMENTO

As fotos utilizadas são do banco de imagens da Georgia Tech face database (GTdb)Tech (2000), o banco contém 750 fotos de 50 pessoas com variação de ângulo, expressões faciais, luminosidade e escala.

Separou-se o conjunto de fotografias em dois grupos, o primeiro contendo 500 imagens (10 faces por indivíduo) para o treinamento e o segundo grupo com 250 imagens (5 faces por indivíduo) para os testes.

As faces foram ajustadas para a mesma dimensão(200x200 pixels) e transformadas para escala de cinza antes de ser realizado o treinamento.

Figura 7 – Imagem redimensionada e transformada para escala de cinza

Fonte: Produção do próprio autor.

Em seguida, criou-se os modelos de treinamentos para os três algoritmos.

Figura 8 – Parte do código para gerar os três modelos.

```
29  modelo_lbph = cv2.face.LBPHFaceRecognizer_create()
30  modelo_lbph.train(dados_treinamento, sujeitos)
31  modelo_lbph.write('modelo_lbhf.yml')
32
33  modelo_fisherfaces = cv2.face.FisherFaceRecognizer_create()
34  modelo_fisherfaces.train(dados_treinamento, sujeitos)
35  modelo_fisherfaces.write("modelo_ff.yml")
36
37  modelo_eingenfaces = cv2.face.EigenFaceRecognizer_create()
38  modelo_eingenfaces.train(dados_treinamento, sujeitos)
39  modelo_eingenfaces.write("modelo_eg.yml")
```

Fonte: Produção do próprio autor.

Na Figura 8 constata-se a criação do modelo_lbhf.yml, modelo_ff.yml e modelo_eg.yml que são os modelos dos algoritmos LBPH, FisherFaces e EigenFaces respectivamente.

Realizado o treinamento, foram feitos os testes com os três algoritmos para as 250 fotos das 50 pessoas.

Figura 9 – Parte do código para o teste do algoritmo LBPH

```
29
30
31  def lbph(teste):
32      modelo_lbph = cv2.face.LBPHFaceRecognizer_create()
33      modelo_lbph.read('modelo_lbhf.yml')
34      inicio = time.time()
35      id, conf = modelo_lbph.predict(dados_teste[teste])
36      final = time.time()
37      return id, conf, final-inicio
38
39
40  for i, arq in enumerate(lista):
41      id_identificado, conf, tempo = lbph(i)
42      id_testado = arq[1:3]
43      with open('resultador_lbph.csv', 'a', newline='') as file:
44              writer = csv.writer(file)
45              writer.writerow([tempo, id_testado, id_identificado, conf])
46
```

Fonte: Produção do próprio autor.

Os resultados como tempo de processamento, limiar, indivíduo testado e identificado foram salvos em um arquivo csv para posterior análise.

Figura 10 – Dados retirados para cada amostra

	A	B	C	D	E
1	0.0468742	3744262695,01,2,47.03805947909994			
2	0.06249570846557617,01,1,34.37813417083789				
3	0.0639586448669433	6,01,1,43.841776389844			
4	0.046875715255737305,01,1,32.04059310783996				
5	0.0625057220458984	4,01,33,43.195838997783916			
6	0.06250429153442383,02,1,41.5856595817382				
7	0.046875,02,2,37.21438475055947				
8	0.0468719005584716	8,02,2,29.7688513274864			
9	0.046875,02,2,31.6700855319012				
10	0.04687762260437012,02,2,40.575929837609344				
11	0.046875715255737305,03,3,29.41762469291041				
12	0.04687237739562988,03,3,39.41460861817177				
13	0.062497615814208984,03,3,29.009639314599234				
14	0.06249785423278808	6,03,3,27.579680654372186			
15	0.0428631305694580	1,03,3,26.892445280456037			
16	0.06249928474426269	5,04,4,39.588423329585694			
17	0.0528533458709716	8,04,4,24.56831671260634			
18	0.046875953674316406,04,4,28.3435220600451				
19	0.046875,04,4,26.969033893770433				
20	0.0468769073486328	1,04,4,34.287561049555244			
21	0.0625,05,42,46.56301410690111				

Fonte: Produção do próprio autor.

Alguns dos critérios estabelecidos para análise dos algoritmos foram baseados no artigo de Coco (2016) e no livro Li e Jain (2011).

Quadro 1 – Critérios estabelecidos

Verdadeiro Positivo	Resultado do teste é dentro do *threshold* (limiar) e a pessoa identificada corresponde a testada
Verdadeiro Negativo	Resultado do teste é fora do *threshold* e a pessoa identificada não corresponde a testada.
Falso Positivo	Resultado do teste é dentro do *threshold* e a pessoa identificada não corresponde a testada.
Falso Negativo	Resultado do teste é fora do *threshold* e a pessoa identificada corresponde a testada

Fonte: Produção do próprio autor

O tempo de processamento foi considerado como o tempo para executar o comando de predição (*predict*). A acurácia total é dada pela relação abaixo:

$$Ac = \frac{N_a}{N_t}.100 \tag{1}$$

Sendo:

Ac = Acurácia

N_a = Número de acertos (verdadeiros positivos e verdadeiros negativos)

N_t = Número de testes

Após obtenção dos dados em csv, foi ajustado para retirar informações.

Figura 11 – Dados tratados

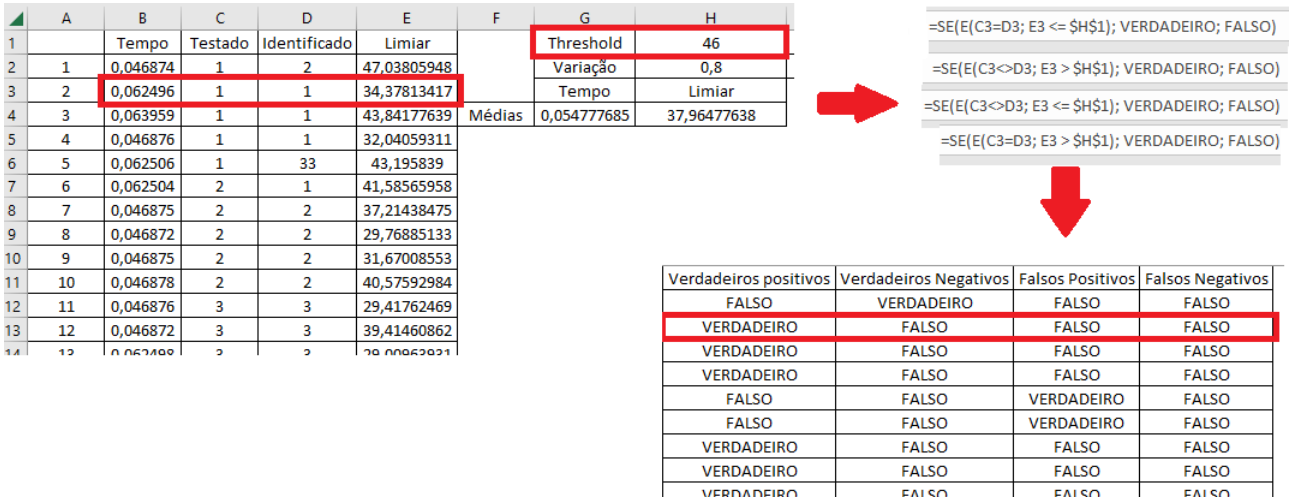
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Tempo	Testado	Identificado	Limiar		Threshold	46				Verdadeiros positivos	Verdadeiros Negativos	Falsos Positivos
2	1	0,046874	1	2	47,03805948		Variação	0,8	1	1,2		FALSO	VERDADEIRO	FALSO
3	2	0,062496	1	1	34,37813417		Tempo	Limiar				VERDADEIRO	FALSO	FALSO
4	3	0,063959	1	1	43,84177639	Médias	0,054777685	37,96477638				VERDADEIRO	FALSO	FALSO
5	4	0,046876	1	1	32,04059311							VERDADEIRO	FALSO	FALSO
6	5	0,062506	1	33	43,195839							FALSO	FALSO	VERDADEIRO
7	6	0,062504	2	1	41,58565958							FALSO	FALSO	VERDADEIRO
8	7	0,046875	2	2	37,21438475			Verdadeiros positivos	77,6			VERDADEIRO	FALSO	FALSO
9	8	0,046872	2	2	29,76885133			Verdadeiros Negativos	10			VERDADEIRO	FALSO	FALSO
10	9	0,046875	2	2	31,67008553			Falsos Positivos	10,8			VERDADEIRO	FALSO	FALSO
11	10	0,046878	2	2	40,57592984			Falsos Negativos	1,6			VERDADEIRO	FALSO	FALSO
12	11	0,046876	3	3	29,41762469			Acurácia	87,6			VERDADEIRO	FALSO	FALSO
13	12	0,046872	3	3	39,41460862							VERDADEIRO	FALSO	FALSO
14	13	0,062498	3	3	29,00963931							VERDADEIRO	FALSO	FALSO
15	14	0,062498	3	3	27,57968065							VERDADEIRO	FALSO	FALSO
16	15	0,042863	3	3	26,89244528							VERDADEIRO	FALSO	FALSO
17	16	0,062499	4	4	39,58842333							VERDADEIRO	FALSO	FALSO
18	17	0,052853	4	4	24,56831671							VERDADEIRO	FALSO	FALSO
19	18	0,046876	4	4	28,34352206							VERDADEIRO	FALSO	FALSO
20	19	0,046875	4	4	26,96903389							VERDADEIRO	FALSO	FALSO
21	20	0,046877	4	4	34,28756105							VERDADEIRO	FALSO	FALSO
22	21	0,0625	5	42	46,56301411							FALSO	VERDADEIRO	FALSO
23	22	0,046877	5	5	32,93858437							VERDADEIRO	FALSO	FALSO

Fonte: Produção do próprio autor.

Calculou-se a média do tempo de processamento e da confiança(*confidence*), em seguida comparou-se o valor do indivíduo testado com o identificado. Caso fossem iguais, indicando que o algoritmo encontrou a pessoa correta e estivesse abaixo do *threshold*, seria considerado um verdadeiro positivo. Caso fossem iguais mas estivesse acima do *threshold*, seria considerado um falso negativo e assim por diante.

Uma situação ilustrativa destes critérios, seria uma pessoa que deseja entrar em um prédio com sistema de autenticação através de reconhecimento facial, e não consegue apesar de estar cadastrado, está situação caracterizaria um falso negativo.

Figura 12 – Verificação se os dados atendem aos critérios



	A	B	C	D	E	F	G	H
1		Tempo	Testado	Identificado	Limiar		Threshold	46
2	1	0,046874	1	2	47,03805948		Variação	0,8
3	2	0,062496	1	1	34,37813417		Tempo	Limiar
4	3	0,063959	1	1	43,84177639	Médias	0,054777685	37,96477638
5	4	0,046876	1	1	32,04059311			
6	5	0,062506	1	33	43,195839			
7	6	0,062504	2	1	41,58565958			
8	7	0,046875	2	2	37,21438475			
9	8	0,046872	2	2	29,76885133			
10	9	0,046875	2	2	31,67008553			
11	10	0,046878	2	2	40,57592984			
12	11	0,046876	3	3	29,41762469			
13	12	0,046872	3	3	39,41460862			

Verdadeiros positivos	Verdadeiros Negativos	Falsos Positivos	Falsos Negativos
FALSO	VERDADEIRO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	FALSO
FALSO	FALSO	VERDADEIRO	FALSO
FALSO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	FALSO

Fonte: Produção do próprio autor.

Logo em seguida, foi calculado o número de ’verdadeiros’ para cada uma das quatro possibilidades e calculado a acurácia.

Figura 13 – Resultados para os critérios do algoritmo LBPH

Verdadeiros positivos	77,6
Verdadeiros Negativos	10
Falsos Positivos	10,8
Falsos Negativos	1,6
Acurácia	87,6

Fonte: Produção do próprio autor.

4 RESULTADOS

Após a realização dos testes , variou-se o *threshold* em 80%, 100% e 120% da média do valor de *confidence* encontrado e obteve-se os resultados das tabelas 3, 4 e 5:

Tabela 3 – Resultados dos testes para *threshold* em 80%

	LBPH	EigenFaces	FisherFaces
Threshold (Limiar)	30	3926	790
Verdadeiro Positivo	14,4	18,4	14,8
Verdadeiro Negativo	20,8	27,6	47,2
Falso Positivo	0	0	0,8
Falso Negativo	64,8	54	37,2
Acurácia total (%)	35,2	46,0	62,0

Fonte: Produção do próprio autor.

Na tabela 3 estão os valores para o *threshold* em 80% da média, o que torna mais exigente a avaliação. Neste caso, destaca-se a baixa acurácia e o número de falsos negativos dos algoritmos LBPH e do EigenFaces. O usuário de um sistema de autenticação que utiliza-se qualquer um dos três algoritmos precisaria tentar sucessivas vezes até que fosse autenticado, isto é, se o sistema permiti-se várias tentativas. Entretanto, pontua-se o baixo número de falsos positivos em todos os algoritmos.

Tabela 4 – Resultados dos testes para *threshold* em 100%

	LBPH	EigenFaces	FisherFaces
Threshold (Limiar)	38	4908	988
Verdadeiro Positivo	48,4	42	32,0
Verdadeiro Negativo	20,8	24	35,6
Falso Positivo	0,0	3,6	12,4
Falso Negativo	30,8	30,4	20,0
Acurácia total (%)	69,2	66,0	67,6

Fonte: Produção do próprio autor.

Na tabela 4 estão os valores para o *threshold* na média do *confidence*. Houve uma melhora expressiva no número de verdadeiros positivos de todos os algoritmos e uma grande redução nos falsos negativos. A acurácia também melhorou contudo ainda é muita baixa para aplicações reais.

Tabela 5 – Resultados dos testes para *threshold* em 120%

	LBPH	EigenFaces	FisherFaces
Threshold (Limiar)	46	5890	1186
Verdadeiro Positivo	77,6	64,8	47,2
Verdadeiro Negativo	10,0	10	8,4
Falso Positivo	10,8	17,6	39,6
Falso Negativo	1,6	7,6	4,8
Acurácia total (%)	87,6	74,8	55,6

Fonte: Produção do próprio autor.

Na tabela 5 estão os valores para o *threshold* em 120% da média, o que torna menos exigente a avaliação. O LBPH apresentou uma acurácia muita boa, com baixo nível de falso negativo mas o número de falsos positivos aumentou, o que já era esperado, devido ao limiar mais alto. EigenFaces também mostrou um desempenho significativamente melhor do que nos *threshold* anteriores. O destaque negativo é FisherFaces que diminuiu a acurácia e aumentou muito os falsos positivos (39,6%).

Tabela 6 – Resultados do tempo de processamento

	LBPH	EigenFaces	FisherFaces
Tempo de proces. (s)	0,0548	0,0594	0,0038

Fonte: Produção do próprio autor.

O algoritmo LBPH tem a maior acurácia e menor limiar que os demais contudo necessita de mais tempo de processamento que o FisherFaces. Pode-se observar que utilizando o *threshold* na média já é possível uma acurácia de 87,6%.

O EigenFaces tem uma acurácia e limiar inferior ao LBPH e leva mais tempo que todos os demais, contudo apresenta uma acurácia razoável em comparação aos outros. Quanto ao FisherFaces, é o algoritmo com pior desempenho, acertando somente 67,6% das faces na melhor das situações.

5 CONCLUSÃO

Considerando o objetivo de estudar a eficácia dos algoritmos de reconhecimento facial presentes na biblioteca OpenCV, foi estabelecido critérios e analisado o desempenho dos algoritmos em relação a um banco de imagens que contém variações de expressões faciais e luminosidade.

O LBPH é o mais eficaz, considerando todos os critérios e condições testadas. FisherFaces tem o menor tempo de processamento e sob condições adequadas pode apresentar bons resultados assim como EigenFaces, apesar de levar o maior tempo de processamento e ter o maior limiar.

Um ponto importante é a definição do *threshold*, um ajuste fino do mesmo pode proporcionar resultados satisfatórios em ambientes controlados, onde se tem luminosidade estável e a coleta de imagens é padrão.

Constata-se que os resultados obtidos demonstram que a acurácia dos algoritmos é baixa para aplicações reais. O algoritmo LBPH tem maior acurácia do que os demais entretanto ainda é baixa (87,6%).

Por fim, seria proveitoso novas análises com banco de imagens diferentes e comparação com mais algoritmos de reconhecimento facial, para identificar possíveis pontos de aperfeiçoamento.

REFERÊNCIAS

AHONEN, T.; HADID, A.; PIETIKÄINEN, M. Face recognition with local binary patterns. In: EUROPEAN CONFERENCE ON COMPUTER VISION. **Proceedings [...]**. 2004. p. 469–481. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-540-24670-1_36>. Acesso em: 6 jan 2023.

BELHUMEUR, P.; HESPANHA, J.; KRIEGMAN, D. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 19, n. 7, p. 711–720, 1997. Disponível em: <https://ieeexplore.ieee.org/document/598228>. Acesso em: 10 jan 2023.

COCO, P. C. M. D. Face recognition algorithms: performance evaluation. **PARSEC 3.26**, p. 10, 2016. Disponível em: <https://www.parsec326.it/images/doc/Report_Reco-Performance-Evaluation---CNR.PDF>. Acesso em: 4 jan 2023.

JAIN, A. K.; FLYNN, P.; ROSS, A. A. **Handbook of Biometrics**. New York: Springer, 2010. *E-book*.

JOLLIFFE, I. T. **Principal Component Analysis**. New York: Springer, 2002. *E-book*.

LI, S. Z.; JAIN, A. K. **Handbook of Face Recognition**. London: Springer, 2011. *E-book*.

OJALA, T.; PIETIKAINEN, M.; MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 24, n. 7, p. 971–987, 2002. Disponível em: <http://vision.stanford.edu/teaching/cs231b_spring1415/papers/lbp.pdf>. Acesso em: 6 jan 2023.

OPENCV. **Face recognition with opencv**. 2008. Disponível em: <https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html>. Acesso em: 9 ago 2022.

PHILLIPS, P. J.; MCCABE, R. M.; CHELLAPPA, R. Biometric image processing and recognition. In: EUROPEAN SIGNAL PROCESSING CONFERENCE, 9. **Proceedings [...]**. 1998. p. 1–8. Disponível em: <http://www.eurasip.org/Proceedings/Eusipco/Eusipco1998/sessions/T%20M/Plenary%20Talk/VISUALDE.PDF>. Acesso em: 15 jan 2023.

PIETIKäINEN, M. Local Binary Patterns. **Scholarpedia**, v. 5, n. 3, p. 9775, 2010. Disponível em: <http://www.scholarpedia.org/article/Local_Binary_Patterns>. Acesso em: 6 jan 2023.

TECH, G. **Georgia Tech Face Database**. 2000. Disponível em: <http://www.anefian.com/research/face_reco.htm>. Acesso em: 3 jan 2023.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. **Journal of Cognitive Neuroscience**, v. 3, p. 71–86, 1991. Disponível em: <https://direct.mit.edu/jocn/article/3/1/71/3025/Eigenfaces-for-Recognition>. Acesso em: 10 jan 2023.

TURK, M.; PENTLAND, A. Face recognition using eigenfaces. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. **Proceedings [...]**. 1991. p. 586–591. Disponível em: <https://ieeexplore.ieee.org/document/139758>. Acesso em: 12 jan 2023.

ZHAO, W. et al. Face recognition: A literature survey. **ACM Computing Surveys**, v. 35, p. 339–458, 2003. Disponível em: <https://inc.ucsd.edu/mplab/users/marni/Igert/Zhao_2003.pdf>. Acesso em: 11 jan 2023.