Autenticação de Usuário: Uma comparação dos métodos Eigenfaces, Fisherfaces e Linear Pattern Histogram (LBPH)

Jonathan Sebastian da Silva Moraes

Resumo — Nos últimos anos o uso do reconhecimento facial para autenticação de usuário vem se tornando bastante popular, tendo isso em mente, este trabalho tem como objetivo realizar a comparação das técnicas Eigenfaces, Fisherfaces e Local Binary Pattern Histogram (LBPH) para verificar qual delas é mais eficiente durante o reconhecimento facial. Também serão apresentados como cada um dos algoritmos citados realizam o processo de reconhecimento facial e os dados estatisticos para cada um deles.

Palavras-Chave — Algoritmo, Eficacia, Eigenfaces, Fisherfaces, LBPH, Reconhecimento Facial

1 INTRODUÇÃO

Nos últimos anos o uso do reconhecimento facial para autenticação de usuário vem se tornando bastante popular devido ao crescimento do poder computacional [1]. Contudo, mesmo com um grande poder computacional existem diversos fatores que podem influenciar durante o reconhecimento, como por exemplo, o excesso ou a falta de iluminação, falta de qualidade nas imagens, gestos e posicionamento das faces.

Com a chegada da COVID-19 o ensino online cresceu bastante em todo o mundo, todo o ensino que antes era presencial se tornou em um ensino a distância, seja ele de nível fundamental, médio ou superior, até mesmo muitos serviços que antes eram presenciais se se transformaram em servicos online, com isso o uso do reconhecimento facial pode ser inserido em diversos ambientes, como por exemplo, o uso do reconhecimento facial em um ambiente virtual de aprendizagem (AVA), onde o aluno seria monitorado através de uma câmera, onde seria continuamente realizado a detecção da face do estudande e verificando se a mesma consta entre as faces cadastradas no sistema, podendo assim validar se o aluno que está assistindo a aula é realmente o aluno cadastrado no sistema. Tendo isso em mente, o objetivo deste estudo foi realizar a avaliação dos algoritmos Eigenfaces, Fisherfaces e LBPH disponibilizados pelo OpenCV, para então assim, poder selecionar qual o melhor deles para ser utilizado em um sistema de autenticação de usuário por reconhecimento facial.

Estudos revelam que os algoritmos Eigenfaces e Fisherfaces são sensiveis a variações de iluminação, ruidos e oclusão [2]. Entretanto, o LBPH é um algoritmo relativamente novo que comparado com o Eigenfaces e o Fisherfaces demonstra ser significativamente mais robusto em relação a iluminação e variação de poses [1].

As hipótezes eram:

H0: Os algoritmos não possuem diferença durante o reconhecimento facial.

H1: Ao menos 1 dos algoritmos possui diferença durante o reconhecimento facial, tendo uma taxa de acerto de 4% ou mais de diferença entre os algoritmos.

2 MÉTODOS UTILIZADOS

Nesta seção é explicado como que funciona cada um dos algoritmos utilizados para resolver o problema de reconhecimento facial.

2.1 Eigenfaces

O algoritmo eigenfaces implementa o Principal Component Analysis (PCA) [3]. PCA é um método utilizado para reduzir a dimensionalidade [4]. Após a redução da dimensionalidade e extraída a face média a

• *M. S. S. Jonathan Acadêmico em Ciência da Computação, Universidade Tiradentes, Aracaju, Brasil. E-mail: jonathan.ssm.dev@outlook.com.*

partir de todas as imagens de uma determinada pessoa. A face média é responsável por mostrar quais são os desvios e quais as variações da imagem.

A estratégia deste método é extrair os traços característicos da face representando a face em questão como uma combinação linear das chamadas 'eigenfaces' obtidas pelo processo de extração de características. Os principais componentes das faces do conjunto de treinamento são calculados. O reconhecimento é alcançado usando a projeção da face no espaço formado pelos eigenfaces. É realizada uma comparação com base na distância euclidiana dos eigenvectors dos eigenfaces e do eigenface da imagem em questão, se esta distância for pequena o bastante, a pessoa é identificada [5].

Para realizar o cálculo do funcionamento do algoritmo, considere ter um conjunto de imagens de rosto que podem ser representados em um espaço g-dimensional. O PCA visa obter um subespaço h-dimensional tendo vetores de base que correspondem as direções de variação máxima do espaço g-dimensional. Logo, h é muito menor que g. O subespaço original é mapeado dentro do menor por meio de uma transformação linear. As imagens de treinamento L são normalizadas e depois subtraídas da

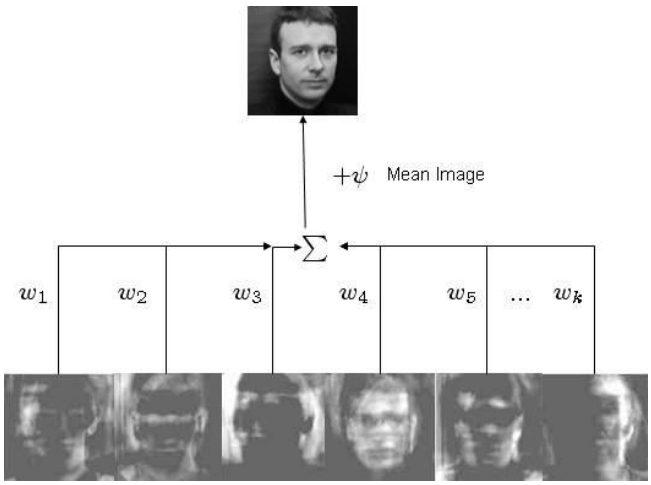

Fig. 1. Um exemplo do funcionamento do Eigenfaces

Fonte: https://onionesquereality.files.wordpress.com/2009/02/eigenfaces-reconstruction.jpg

imagem média calculada para obter a média central das imagens. Seja W uma matriz contendo a média central das imagens de treinamento W_i onde i = 1, 2, 3, ..., L. De W, obtemos a matriz de covariância D, onde:

$$D = WW^T$$

A matriz de covariância D pode ser muito grande e foi provado matematicamente que podemos usar $D = W^T W$ em vez disso. Então de D, obtemos os eigenvectors e_i e seus eigenvalues associados λ_i. Em seguida, uma matriz E é formada composta por eigenvectors associados aos maiores eigenvalues para obtermos:

$$Z_i = E^T w_i \text{ onde, } i = 1, 2, ...L$$

Z_i são os novos vetores no novo subespaço dimensional inferior. Portando, as imagens são projetadas desta forma no novo subespaço. Em um teste a imagem é tratada da mesma maneira até projetada no mesmo subespaço. Então é utilizada a distância medida para testar a semelhança entre o teste e as imagens de treinamento [4].

2.2 Fisherfaces

Para o algoritmo fisherfaces utilize os termos fisherfaces e LDA de forma intercambiável, Linear Discriminant Analysis (LDA) [4, 6, 7] tenta diferenciar entre as classes em vez de tentar apresentar os dados. Portanto, LDA se preocupa em obter vetores para a discriminação das classes. São definidas duas matrizes de disperção:

$$S_w = \sum_{j=1}^{R} \sum_{i=1}^{M_j} (x_i^j - \mu_j)(x_i^j - \mu_j)^T$$

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

A primeira é chamada de matriz de disperção dentro da classe equanto a segunda é chamada de disperção entre classes matrizes. j denota a classe enquanto i denota o número da imagem. μ_j é a medida da classe j equanto μ é a média de todas as classes. M_j é o número de imagens da classe j e R é o número de classes. O algoritmo visa maximizar a matriz entre classes enquanto minimiza a matriz dentro da classe isto pode ser feito maximizando a proporção ($\det | S_b | / \det | S_w |$). Como no PCA nos temos a projeção da matriz G. Esta matriz é usada para maximizar a proporção mencionada quando as colunas são eigenvectors de $S_w^{-1} S_b$. Um problema surge quando S_w se torna singular. [4, 7, 8] sugerem utilizar um espaço intermediario –

antes de transformar o espaço original no espaço final – que é o espaço PCA. Em outras palavras, o espaço original é projetado no espaço PCA antes de ser projetado no espaço LDA.

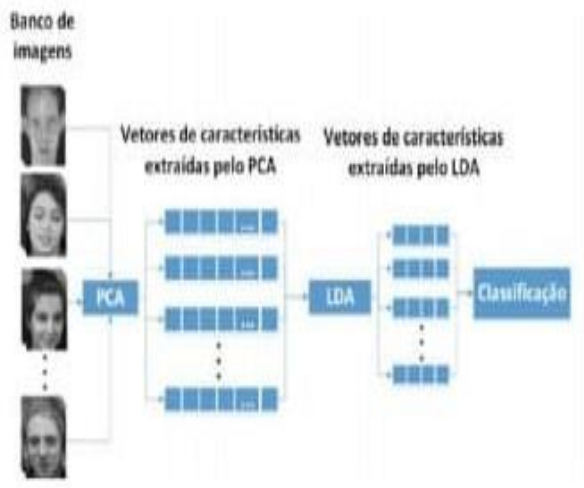

Fig. 2. Um exemplo do funcionamento do Fisherfaces

Fonte: https://repositorio.ufu.br/bitstream/123456789/22158/3/ReconhecimentoFacialAlgotimos.pdf

2.3 Local Binary Pattern Histogram (LBPH)

O Local Binary Pattern Histogram (LBPH), proposto em 2006 [9, 10] é um algoritmo utilizado para reconhecimento facial baseado no operador binário local [10, 11]. Segundo [10, 12] o LBPH é um algoritmo muito popular e amplamente utilizado, devido ao seu poder discriminativo e a sua simplicidade de computação.

O LBPH funciona da seguinte forma, é utilizada uma imagem N x M e é dividido em regiões. O mesmo tamanho é o ideal, na largura e na altura, resultando em regiões m x m. Em cada região, o operador binário local é utilizado. Quando aplicado em imagens, este opeador, compara um pixel a seus oito pixels vizinhos mais próximos. A coparação verifica se o valor do pixel vizinho é mais alto que o pixel central, retornando um valor de '1' neste caso. Em qualque outro caso, um valor '0' é retornado. Aplicando este processo a todos os 8 vizinhos, 8 valores binários são derivados. Um número binário de 8 bits é formado ao fundir estes valores. O número binário obtido pode ser traduzido em um valor decimal, chamado de valor LBP do pixel, em um intervalo entre 0-255 [10]. A Figura 3 demonstra como esta operação é realizada. O processo é executado para cada pixel na região.

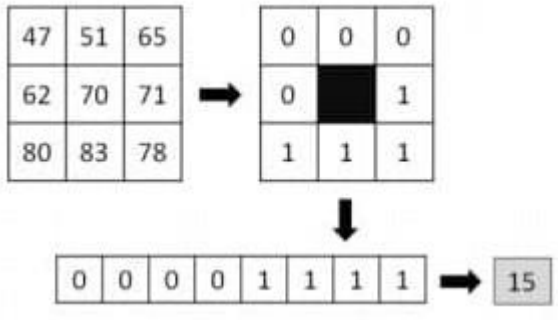

Fig. 3. Exemplo do operador LBP

O histograma para a região atual é então criado, calculando o número de aparecimentos de todos os valores LBP nesta região. Através deste processo, o histograma para cada região consiste em 256 bins. Este processo é representado por meio da seguinte equação:

$$H_i = \sum_{x,y} I\{LBP(J(x,y)) = i\}, i = 0, \dots, 255 \textit{(1)}$$

Onde H_i é o bin do valor i, J (x, y) é o pixel (x, y) da imagem e I é um operador condicional, retornando '1' se a afirmação é verdadeita ou '0' se a afirmação for falsa. Uma vez que os histogramas para todas as regiões foram calculados, um único histograma é criado unificando todos os histogramas de cada seção. Este histograma final conterá 256 * m * m bins e é definido como o vetor de recursos da imagem [10].

Para prosseguir com o reconhecimento facial, um banco de dados (bd) de vetores correspondentes às imagens de rostos são necessários. A distância entre os diferentes vetores de caracteristicas são então calculados. A imagem que corresponde ao vetor de recurso com a distância mais proxima é retornado como a melhor correspondência para a imagem testada. Isso pode ser derivado da seguinte equação:

$$(\mathrm{M}, \mathrm{Index}) = \min(\mathrm{dst}[\mathrm{bd\ size}])$$

Onde dst é a matriz das distâncias calculadas e min é a função que retorna o valor mínimo (M) da matriz e o valor do indice (Index), um limite também é aplicado para excluir faces que não correspondem a qualquer um dos rostos no banco de dados. Se M for menor que o Treshold o Index é uma correspondência, senão o Index não é uma correspondência [10].

3 METODOLOGIA

3.1 Hardware e Sistema Operacional Utilizado

Durante todo o desenvolvimento do trabalho foi utilizado uma máquina com uma Placa-Mãe Gigabyte X99-UltraGaming, Processador Intel(R) Core (TM) i7-6800K 3.40Ghz, Memória Hyperx 16GB 2666mhz e uma GPU Gigabyte Xtreme Gaming 8GB. O Sistema Operacional utilizado foi o Windows 10.

3.2 Ferramentas e Fases do Processo

Antes de iniciar o processo de testes, foi preciso definir quais ferramentas seriam utilizadas. A linguagem utilizada durante todo o processo de detecção de faces, treinamento dos algoritmos, testes estatísticos e reconhecimento facial foi Python 3.8.2 em conjunto com as bibliotecas Flask 1.1.2, click 7.1.2, itsdangerous 1.1.0, Jinja2 2.11.2, MarkupSafe 1.1.1, pip 20.2.4, setuptools 50.3.2, Werkzeug 1.0.1, Flask-Cors 3.0.9, numpy 1.19.3, opencv-contrib-python 4.3.0.38 e Pillow 8.0.1. Para a implementação da página web responsável pela chamada da API desenvolvida em Python para a realização da autenticação por reconhecimento facial, foi utilizado o framework Angular na versão 10.0.8 juntamente com o Node 10.22.0.

O Python foi escolhido por ser uma linguagem com uma comunidade bastante ativa e por ter inúmeras bibliotecas que facilitam o processo de desnvolvimento, sendo uma delas o OpenCV que apresenta inúmeras ferramentas na área de processamento de imagem, com algoritmos (Eigenfaces, Fisherfaces e LBPH) e funções otimizadas.

Para que seja possível realizar o reconhecimento facial é necessário realizar a coleta das imagens, podendo ser um dataset de imagens, também é necessário realizar a detecção de faces nas imagens para que o dataset possua apenas imagens de faces, depois do dataset preparado é necessário realizar o treinamento do algoritmo, após a finalização do treinamento é realizado o reconhecimento facial que pode ser em tempo real ou utilizando um dataset de imagens de teste. Estas estapas serão detalhadas subseções seguintes.

Os algoritmos utilizados para treinamento e reconhecimento facial em todo o projeto foi o Eigenfaces o Fisherfaces e o Local Binary Pattern Histogram (LBPH). As implementações destes algoritmos na biblioteca OpenCV são bem completas e de fácil uso na linguagem Python.

O processo de aprendizado utilizado foi o supervisionado, onde as estruturas das imagens possuem um (id) para cada pessoa, seguido pelas variações das poses de cada pessoa.

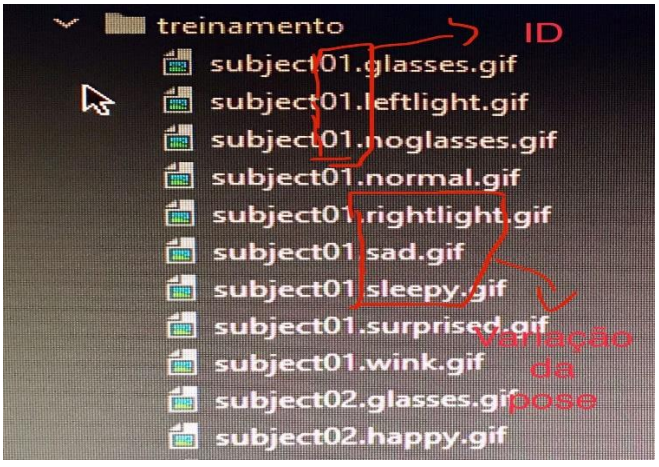

Fig. 4. Estrutura das imagens com seus identificadores e variações de poses

3.2.1 Dataset

O dataset utilizado para a realização dos testes foi o Yalefaces. O Yalefaces é composto por imagens que retratam diferentes indivíduos em diferentes poses sob condições de iluminação controlada [13] e possui 165 imagens de rosto.

3.2.2 Detecção Facial

Segundo [14] considera-se a primeira etapa para o reconhecimento facial, a detecção de características em uma determinada imagem ou frame de vídeo.

Para a detecção das faces foi utilizado a classe CascadeClassifier do OpenCV que utiliza como parâmetro um arquivo .xml, neste trabalho o arquivo utilizado foi o haarcascade_frontalface_default.xml.

3.2.3 Reconhecimento Facial

Segundo [15] reconhecimento facial é o processo de identificação de pessoas em uma imagem ou frame de vídeo. O mesmo autor ainda diz que o reconhecimento facial consiste em duas técnicas:

Características geométricas, que consiste em calcular um conjunto de características geométricas, como largura e comprimento do nariz, posição da boca e formato do queixo etc., da imagem do rosto que

queremos reconhecer. Estes recursos então correspondem as características de uma pessoa conhecida. Uma métrica adequada quanto a distância euclidiana pode ser usada para encontrar a correspondência mais próxima.

Comparação de modelos, onde a base da estratégia é extrair regiões faciais inteiras e comparalas com as imagens de pessoas conheciadas armazenadas. Novamente a distância euclidiana pode ser utilizada para encontrar a correspondência mais próxima. No entando existem métodos muito melhores para a comparação de modelos, como por exemplo o PCA.

4 RESULTADOS

As métricas utilizadas para a avaliação dos algoritmos foram, a porcentagem de acerto, a porcentagem de erro, a acurácia, a média de confiança, o tempo de treinamento e o tempo de teste, para avaliar os algoritmos primeiramente foi realizada a coleta dos dados de todas as métricas sem altercação nos parâmetros dos algoritmos, após a primeira coleta de dados foram realizados outros 4 testes variando os parâmetros de cada algoritmo.

4.1 Eigenfaces

Acerto (%)	Erro (%)	Acurácia (%)	Média Confiança	Tempo Treinamento (s)	Tempo Teste (s)
70.37	29.62	70.37	5997.60	9.68	0.67
11.11	88.88	11.11	1472.60	9.59	0.67
48.14	51.85	48.14	4061.50	9.30	0.78
55.55	44.44	55.55	3169.90	4.24	0.43
62.96	37.03	62.96	2956.63	3.30	0.43

Tab. 1. Resultados dos testes do Eigenfaces

Na primeira linha os parâmetros utilizados foram os padrões do próprio algoritmo, na segunda linha foi utilizado um threshold de 2000, na terceira 8000, na quarta foi utilizado 40 num_components e um threshold de 8000, e na quinta 15 num_components e 8000 no threshold.

Como pode ser analisado na Tabela 1, o algoritmo Eigenfaces possui uma média de confiança ruim, sendo 1472.60 o melhor valor na média de confiança porém com uma taxa de acerto de apenas 11.11% e uma taxa de erro de 88.88%, por ser um algoritmo que possui uma alta sensibilidade a variações de iluminação [2], suas taxas de acerto não são muito boas, como pode ser visto na tabela a maior taxa de acerto do algoritmo foi de 70.37% porem com uma média de confiaça ruim e um maior tempo de treinamento, ao realizar a variação dos parâmetros nota-se que ao aumentar o threshold a taxa de acerto aumenta, entretando, o nível de confiança piora. Além de variar o threshold foi também testado a variação do num_componenents, onde foi possível obter a melhor taxa de acerto em relação a média de confiança, que foi 62.96% de acerto com uma média de confiança de 2956.63. Ao analisar estes dados pode ser concluído que este não seria um algoritmo ideal para realizar uma autenticação de usuário por reconhecimento facial, pois ele precisa de um threshold muito alto para conseguir reconhecer os rostos, o que gera uma média de confiança muito alta, sendo o valor ideal o mais próximo de 0 possivel, o que pode ocasionar reconhecimentos equivocados. Também pode ser notado que ao não passar o número de componentes e utilizar o padrão do próprio algoritmo o tempo de treinamento aumenta.

A análise do nível de confiança em relação ao número de componentes utilizados pode ser observada na Figura 5, onde foi gerado um gráfico demonstrativo dos dados utilizando de 0-200 componentes.

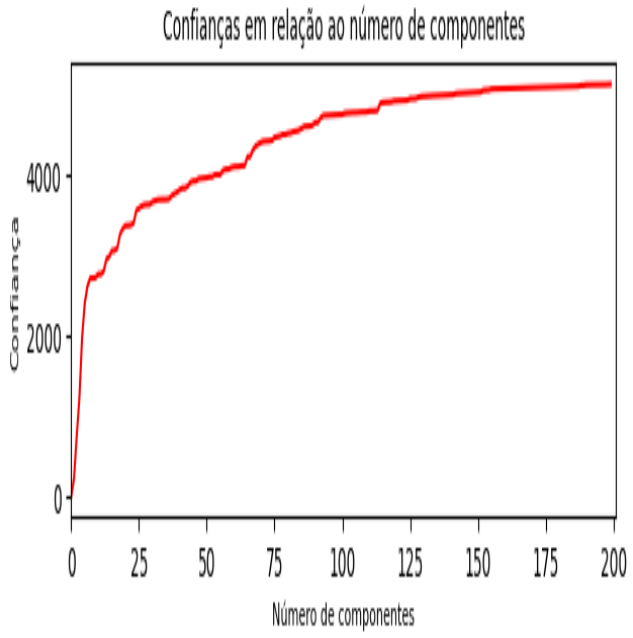

Fig. 5. Relação entre número de componentes e o nível de confiança

Ao analisar o gráfico nota-se que quanto maior o número de componentes pior é o nível de confiança.

A seguir podemos observar a matriz de confusão para cada uma das linhas de resultados da tabela 1:

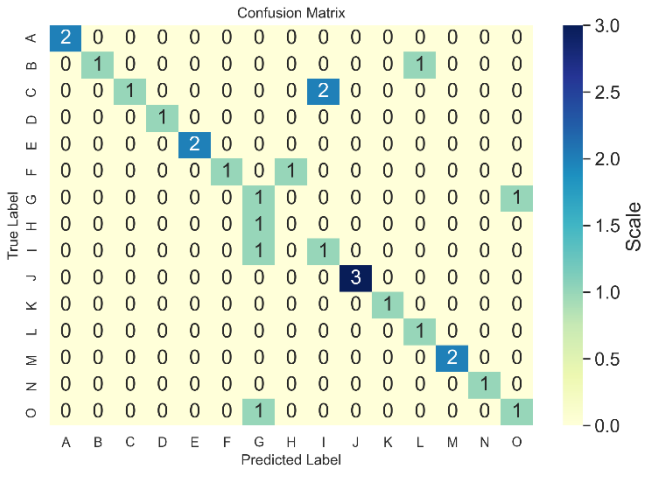

Fig. 6. Matriz Confusão da linha 1 da tabela

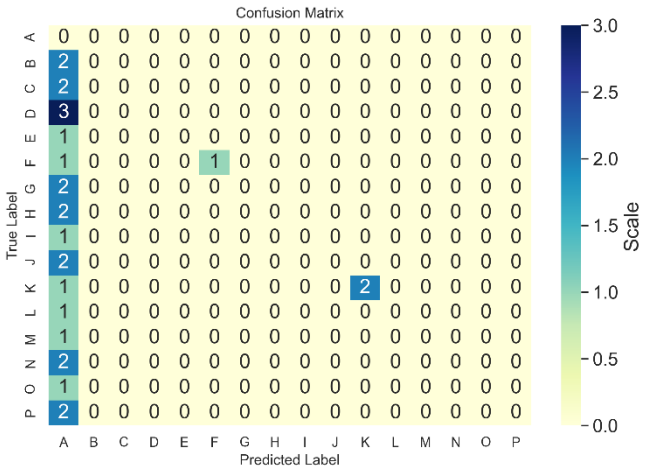

Fig. 7. Matriz Confusão da linha 2 da tabela

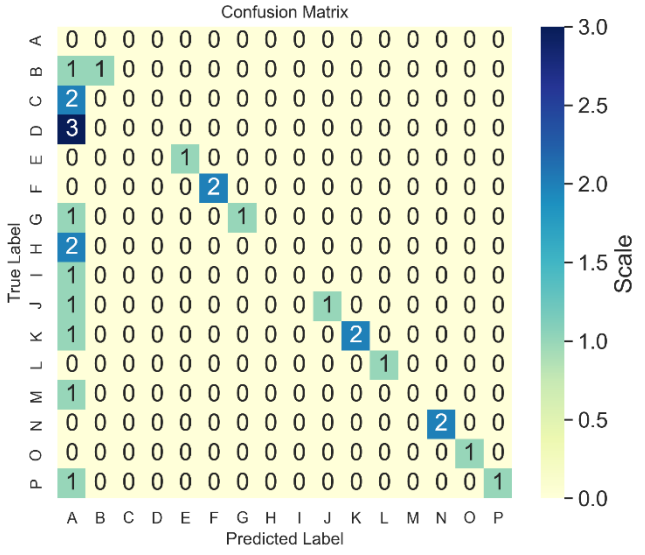

Fig. 8. Matriz Confusão da linha 3 da tabela

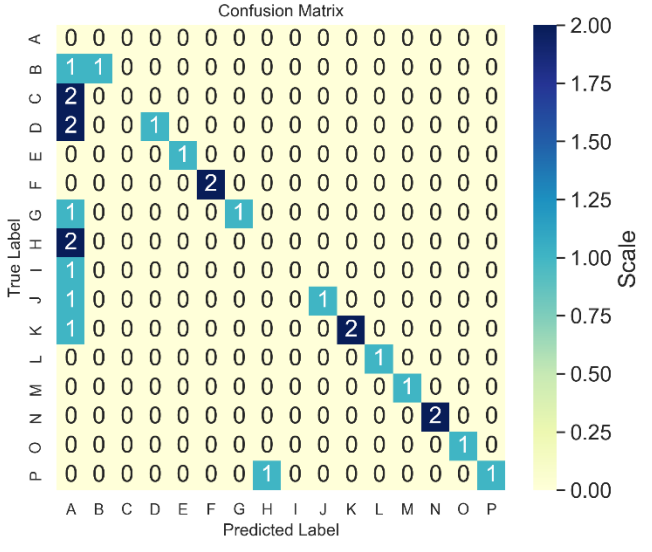

Fig. 9. Matriz Confusão da linha 4 da tabela

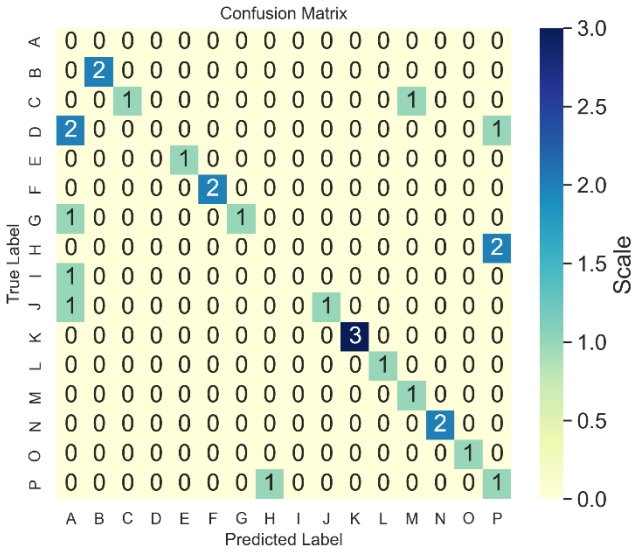

Fig. 10. Matriz Confusão da linha 5 da tabela

4.2 Fisherfaces

Em relação ao algoritmo Eigenfaces o Fisherfaces conseguiu obter melhores taxas de acerto, melhores níveis de confiança e melhores tempos de treinamento.

Acerto (%)	Erro (%)	Acurácia (%)	Média Confiança	Tempo Treinamento (s)	Tempo Teste (s)
70.37	29.62	70.37	1520.37	2.86	0.45
51.85	48.14	51.85	1121.54	2.86	0.36
70.37	29.62	70.37	1520.37	2.86	0.34
77.77	22.22	77.77	491.81	2.31	0.35
48.14	51.85	48.14	241.03	2.59	0.39

Tab. 2. Resultados dos testes do Fisherfaces

Na primeira linha os parâmetros utilizados foram os padrões do próprio algoritmo, na segunda linha foi utilizado um threshold de 2000, na terceira 8000, na quarta foi utilizado 3 num_components e um threshold de 2000, e na quinta 2 num_components e 2000 no threshold.

Ao realizar a análise dos dados obtidos através dos testes do algoritmo Fisherfaces, pode ser observado que sua maior taxa de acerto foi de 77.77% com uma média de confiança de 491.81, uma taxa de erro de 22.22% e 2.31s de tempo de treinamento, também foi notado que a variação do threshold não impacta tanto na média de confiança, entretando, a taxa de acerto é bastante afetada caso o threshold seja muito baixo, no teste onde foi utilizado um threshold de 2000 taxa de acerto foi de 51.85% e quando foi utilizado um threshold de 8000 a taxa de acerto foi de 70.37%, uma diferença de 18.52% maior em relação ao uso de um threshold mais baixo. O tempo de treinamento também não é muito afetado quando não passado o numero de componentes, como acontece com o Eigenfaces, as taxas de acerto e médias de confinaça se manteram estáveis mesmo com a variação do threshold, porém ao variar o número de componentes, tanto a média de confiança quanto a taxa de acerto são bastante afetadas, ao utilizar 3 componentes e 2000 de threshold a taxa de acerto foi de 77.77% e a média de confiança 491.81, e ao diminuir apenas 1 componente, utilizando assim apenas 2, a taxa de acerto caiu 29.63% porém a média de confiança melhorou, ou seja, no algoritmo Fisherfaces quanto maior o número de componentes melhor é a taxa de acerto, porém, ao mesmo tempo que a taxa de acerto sobe a média de confiança diminui.

Também foi realizada uma análise do número de componentes em relação a confiança, a mesma pode ser visualizada na Figura 11:

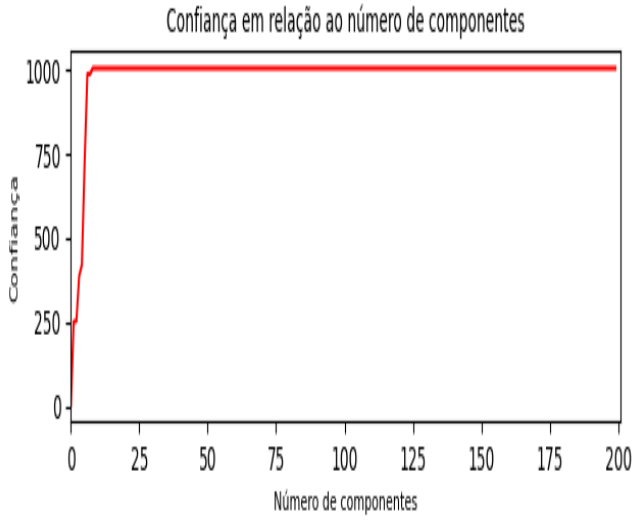

Fig. 11. Relação entre o número de componentes e o nível de confiança

Ao analisar o gráfico percebe-se que a média de confiança também piora quando utilizado um número maior de componentes, porem somente entre 0-20 componentes, depois do 20 nota-se que a média de confiança se estabiliza em 1000.

A seguir podemos observar a matriz de confusão para cada uma das linhas de resultados da tabela 2:

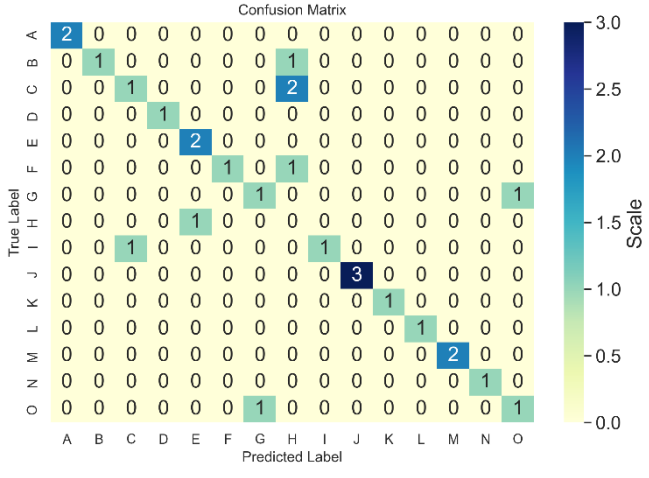

Fig. 12. Matriz Confusão da linha 1 da tabela

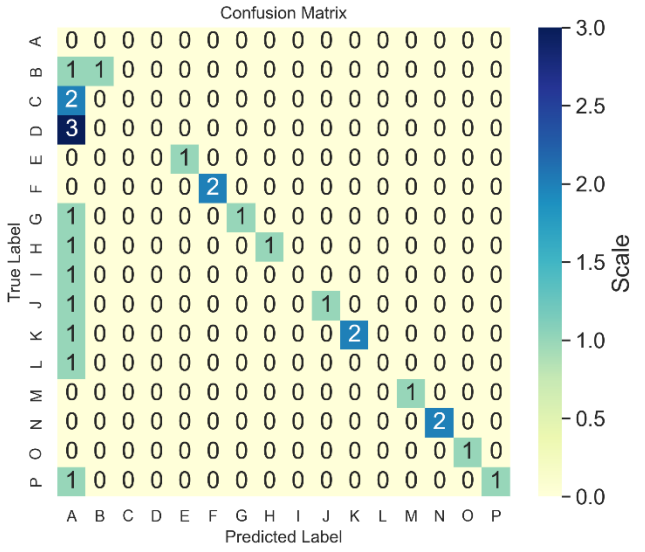

Fig. 13. Matriz Confusão da linha 2 da tabela

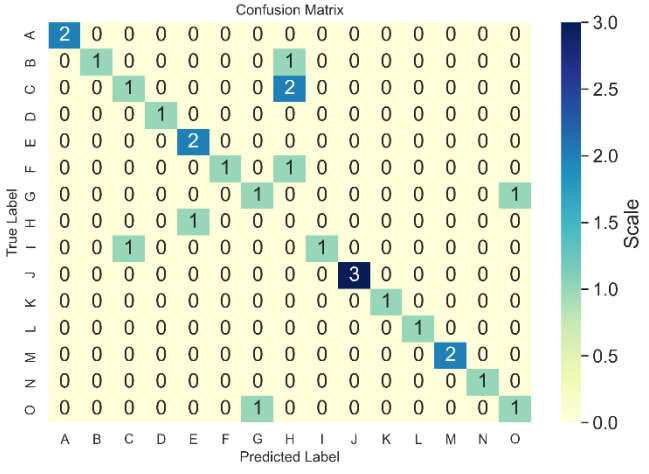

Fig. 13. Matriz Confusão da linha 3 da tabela

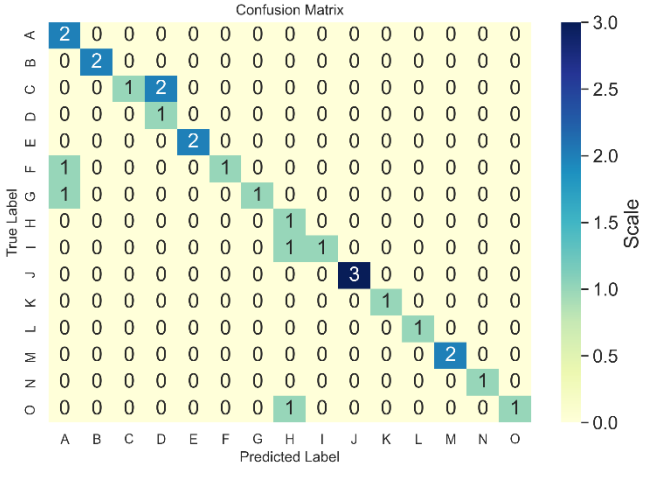

Fig. 14. Matriz Confusão da linha 4 da tabela

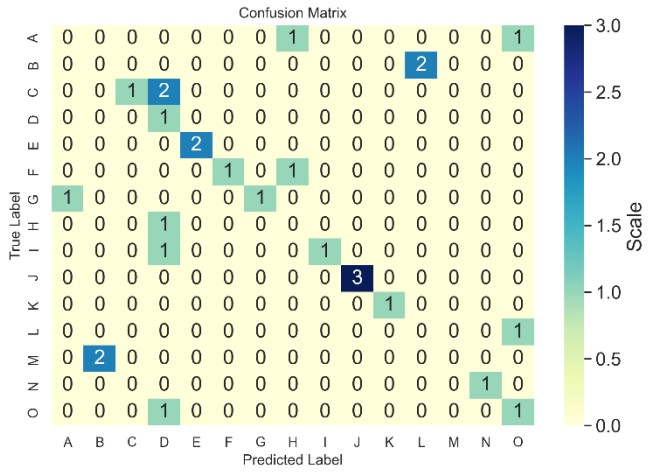

Fig. 15. Matriz Confusão da linha 5 da tabela

4.3 Local Binary Pattern Histogram (LBPH)

O LBPH possui cinco parâmetros que podem ser alterados, são eles, o radius, neighbors, grid_x, grid_y e o threshold. Durante o teste do LBPH também foi realiado o teste inicial sem alteração nos parâmetros, para analisar como o algoritmo se comporta sem modificações e comparar com os dados variando os parâmetros.

A tabela 3 a seguir mostra o resultado dos testes:

Acerto (%)	Erro (%)	Acurácia (%)	Média Confiança	Tempo Treinamento (s)	Tempo Teste (s)
59.25	40.74	59.25	19.09	1.26	0.96
59.25	40.74	59.25	19.09	1.25	0.77
59.25	40.74	59.25	19.09	1.25	0.74
59.25	40.74	59.25	2.25	0.22	0.41
59.25	40.74	59.25	0.89	0.14	0.47

Tab. 3. Resultado dos testes do LBPH

Na primeira linha os parâmetros utilizados foram os padrões do próprio algoritmo, na segunda linha foi utilizado um threshold de 2000, na terceira 4000, na quarta 8000, na quinta foi utilizado 2 no radius e no neighbors, 7 no grid_x e grid_y e 2000 no threshold, e na sexta 1 no radius e neighbors, 7 no grid_x e grid_y e 18 no threshold.

Podemos perceber que a taxa de acerto do LBPH é menor em relação ao Fisherfaces e que ela se manteve em 59.25% mesmo variando os parâmetros, porém suas médias de confiança são bem melhores, a maior taxa de acerto do Fisherfaces foi de 77.77% e a maior do LBPH foi de 59.25%, entretando a média de confiança em relação a taxa de acerto do LBPH foi de 0.89 enquanto a do Fisherfaces foi de 491.81.

No LBPH pode ser notado que a não utilização dos parâmetros impactam um pouco no tempo de treinamento e no tempo de reconhecimento, também pode ser notado que a taxa de acerto e a média de confiança não alteram ao variar apenas o threshold, estes resultados so começam a mudar quando são variados os valores do radius, neighbors, grid_x, grid_y juntamente com o threshold, nota-se que quando alterados esses parâmetros a média de confiança melhrou consideravelmente. Ao diminuir o valor do radius, do neighbors e do threshold, pode ser notado que a média de confiança obteve uma melhora sem alterar a taxa de acerto, mantendo também um baixo tempo de treinamento e de reconhecimento.

A combinação de parâmetros que gerou os melhores resultados para este algoritmo foi, 1 no radius e neighbors, 7 no grid_x e grid_y e 18 no threshold, gerando um resultado de 59.25% de taxa de acerto, 40.74% de taxa de erro, uma média de confiança de 0.89, um tempo de treinamento de 0.14s e um tempo de teste de 0.47s.

Para o LBPH também foram realizados testes que demonstram o nível de confiança em relação ao radius, da média de confiança em relação ao número de vizinhos e da média de confiança em relação a quantidade de células. Os resultados dos testes podem ser observados na Figura 16:

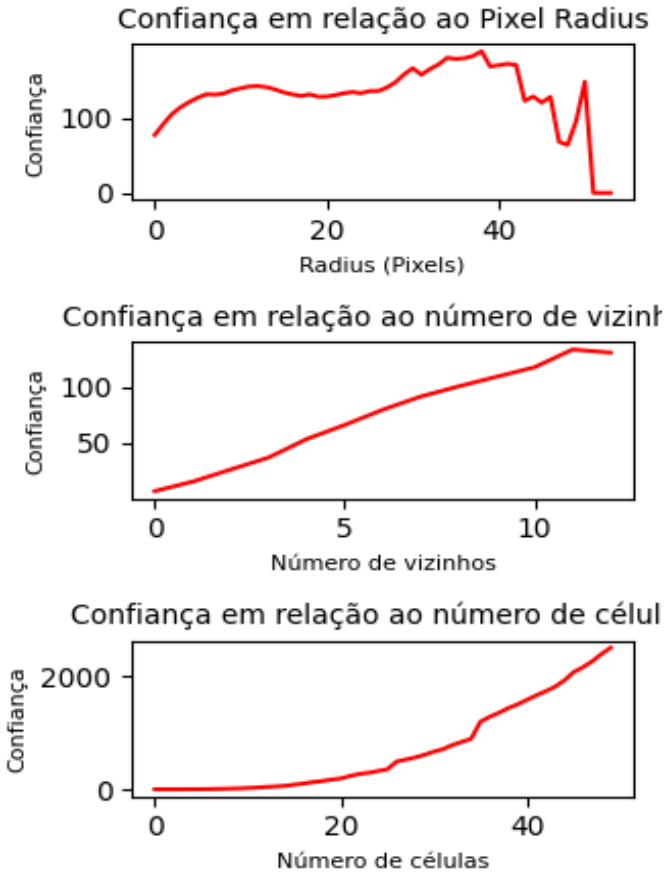

Fig. 16. Relação entre o nível de confiança e o radius, vizinhos e células

Ao analisar os gráficos foi notado que quanto mais células são utilizadas pior fica a média de confiança, pode-se notar também que quanto mais vizinhos são utilizados a média de confiança também piora.

A seguir podemos observar a matriz de confusão para cada uma das linhas de resultados da tabela 2:

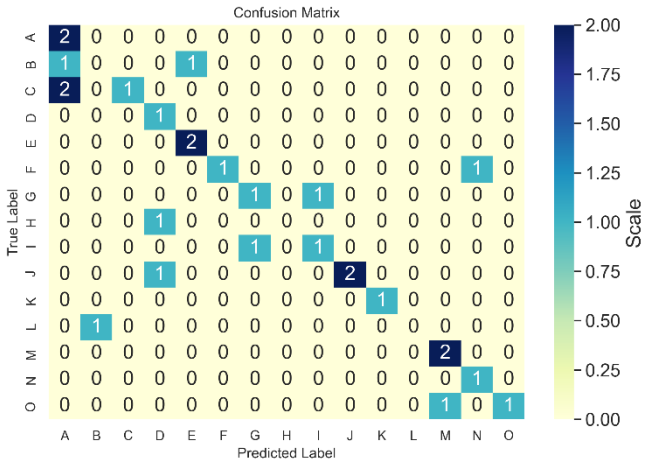

Fig. 17. Matriz Confusão da linha 1 da tabela

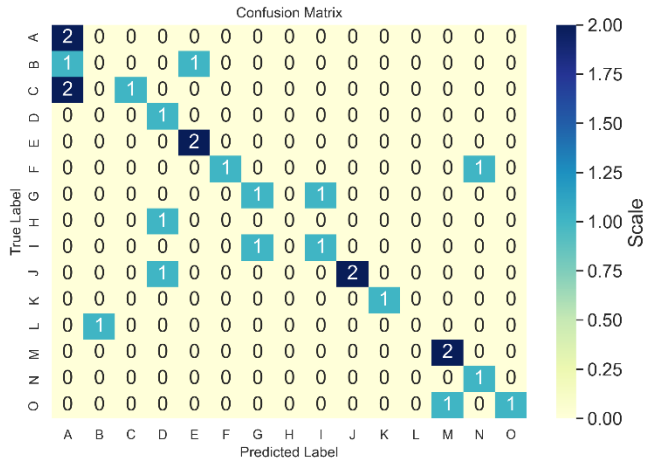

Fig. 18. Matriz Confusão da linha 2 da tabela

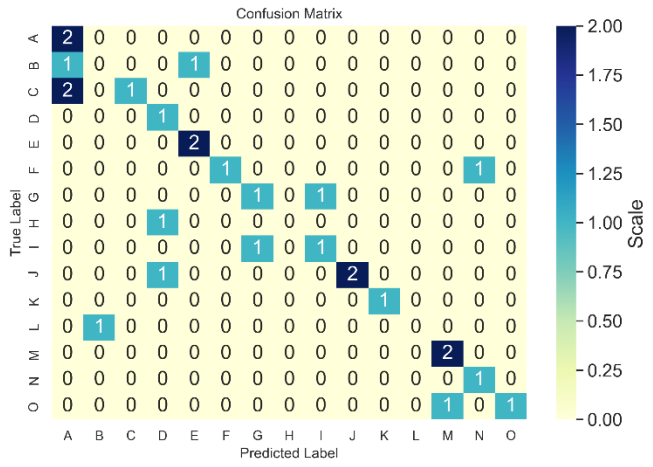

Fig. 19. Matriz Confusão da linha 3 da tabela

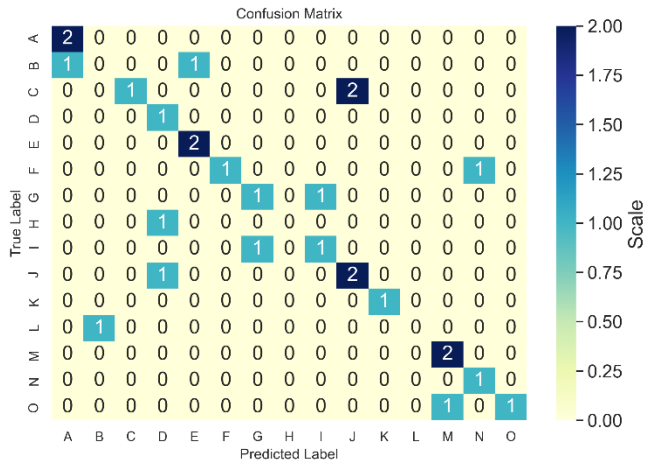

Fig. 20. Matriz Confusão da linha 4 da tabela

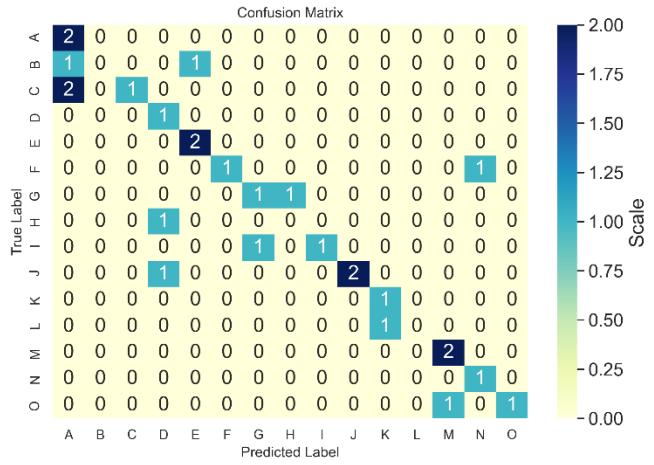

Fig. 21. Matriz Confusão da linha 5 da tabela

5 APLICAÇÃO

Nesta seção serão mostradas algumas imagens da aplicação funcionando.

5.1 Aplicação na IDE

Na figura 22 abaixo podemos visualizar um exemplo do reconhecimento fácil direto na IDE:

Fig. 22. Exemplo de reconhecimen facial na IDE

5.2 Aplicação na Web

As imagens abaixo são exemplos da aplicação web desenvolvida para a realização do reconhecimento facial.

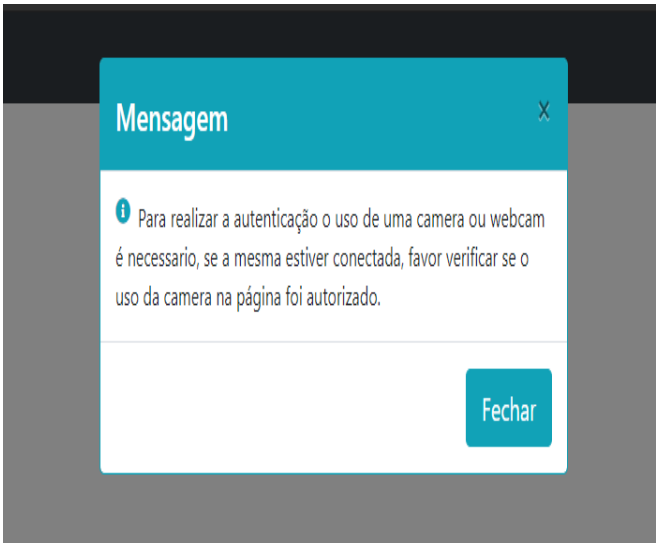

Fig. 23. Exemplo câmera não encontrada

Este aviso informativo irá aparecer quando nenhuma câmera for detectada, “Para realizar a autenticação o uso de uma câmera ou webcam é necessário, se a mesma estiver conectada, favor verificar se o uso da câmera na página foi autorizado.”.

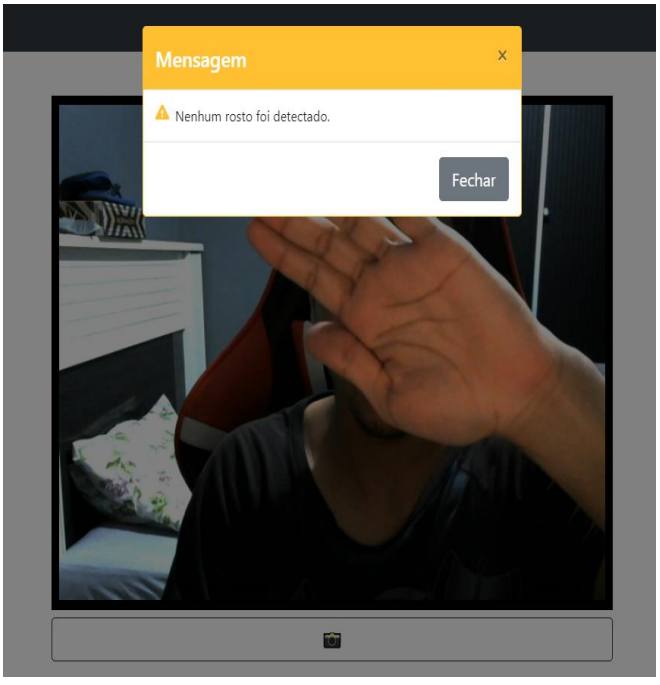

Fig. 24. Exemplo rosto não detectado

Quando nenhum rosto for detectado, este alerta será exibido na página, “Nenhum rosto foi detectado.”.

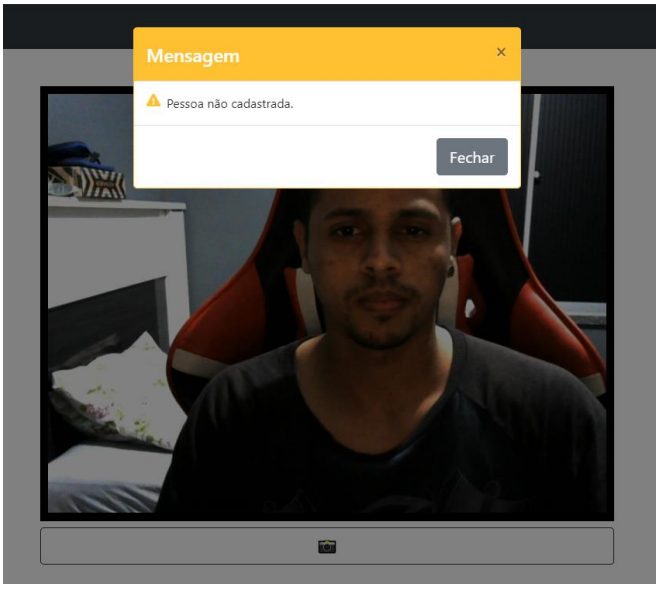

Fig. 25. Exemplo pessoa não cadastrada

No caso de algum rosto ser detectado, porém ele não for encontrado na base treinada, a página retornara este alerta, “Pessoa não cadastrada.”.

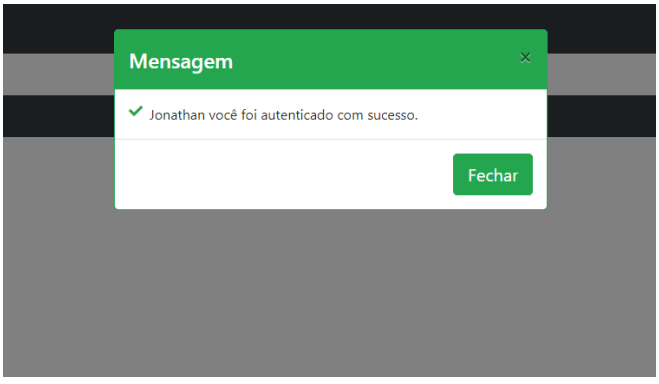

Fig. 26. Exemplo pessoa autenticada

Por fim, quando um rosto for detectado e estiver cadastrado na base de treinamento, está mensagem de sucesso será retornada para o usuário, com o nome da pessoa que foi autenticada concatenado com “você foi autenticado com sucesso.”

6 Conclusão

Três algoritmos foram avaliados para verificar se eles possuiam resultados diferentes durante o reconhecimento facial, a partir das análises, concluimos que a hipotése H0 foi refutada pois os algoritmos demonstraram ter resultados diferentes durante o processo de reconhecimento facial, com uma diferença de 4% na taxa de acerto em seus melhores resultados em relação a média de confiança, o algoritmo Fisherfaces obteve a melhor taxa de acerto entre os três, com uma taxa de 77.77%, entretando o algoritmo Local Binary Pattern Histogram obteve a melhor taxa de acerto em relação a média de confiança que foi 59.25% na taxa de acerto e 0.89 na média de confiança. Como o objetivo do trabalho era avaliar os três algoritmos e verificar com base nos dados obtidos qual dos três algoritmos iria obter o melhor resultado durante o reconhecimento facial para autenticação de usuário, podemos concluir que a melhor escolha seria o LBPH, o mesmo não obteve a melhor taxa de acerto, mas obteve a melhor taxa de acerto em relação a média de confiança, por isso a melhor escolha seria o LBPH, em um sistema de autenticação de usuário, uma média de confiança muito baixa pode gerar a autenticações erradas no sistema.

Referências

[1] A. Özdil and M. M. Özbilen, "A survey on comparison of face recognition algorithms," *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*, Astana, 2014, pp. 1-3, doi: 10.1109/ICAICT.2014.7035956.

[2] N. Hegde, S. Preetha and S. Bhagwat, "Facial Expression Classifier Using Better Technique: FisherFace Algorithm," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 604-610, doi: 10.1109/ICACCI.2018.8554499.

[3] M. Turk and A. Pentland, Eigenfaces for recognition, J Cogn Neurosci 3 (1991), 71–86. W.

[4] M. Sharkas and M. A. Elenien, "Eigenfaces vs. fisherfaces vs. ICA for face recognition; a comparative study," 2008 9th International Conference on Signal Processing, Beijing, 2008, pp. 914-919, doi: 10.1109/ICOSP.2008.4697276.

[5] M.üge Çarıkçı, Figen Özen, A Face Recognition System Based on Eigenfaces Method, Procedia Technology, Volume 1, 2012, Pages 118-123, ISSN 2212-0173, https://doi.org/10.1016/j.protcy.2012.02.023. (http://www.sciencedirect.com/science/article/pii/S2212017312000242)

[6] Zhao, R. Chellappa, A. Krishnaswamy, Discriminant analysis of principal components for face recognition, Proc Third IEEE Int Conf Automatic Face and Gesture Recogn, Nara, Japan, 14–16 April, 1998, pp. 336–341.

[7] P. Belhumeur, J. Hespanha, and D. Kriegman, Eigenfaces vs. fisherfaces: Recognition using class specific linear projection, Proc Fourth Eur Conf Computer Vision, Vol. 1, 14–18 April 1996, Cambridge, UK, pp. 45–58.

[8] D. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," IEEE Transactions on Patt Analysis and Mach Intell, vol. 18, pp. 831-836, 1996.

[9] H. A. Ahonen, T. and M. Pietikinen, “Face description with local binary patterns,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037–2041, 2006.

[10] N. Stekas and D. Van Den Heuvel, "Face Recognition Using Local Binary Patterns Histograms (LBPH) on an FPGA-Based System on Chip (SoC)," 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, 2016, pp. 300-304, doi: 10.1109/IPDPSW.2016.67.

[11] P. M. Ojala, T. and D. Harwood, “A comparative study of texture measures with classification based on feature distributions,” Pattern Recognition, vol. 19, no. 3, pp. 51–59, 1996.

[12] M. Pietikinen, “Local Binary Patterns,” vol. 5, no. 3, p. 9775, 2010.

[13] Damianou, Andreas (2015) *Deep Gaussian Processes and Variational Propagation of Uncertainty.* PhD thesis, University of Sheffield.

[14] Santana, L. M. Q. de, & Rocha, F. G. (2015). Processo de detecção Facial utilizando Viola; Jones. *Interfaces Científicas - Exatas E Tecnológicas*, *1*(1), 35-40. https://doi.org/10.17564/2359-4942.2015v1n1p35-40

[15] Análisis Comparativo de Los Algoritmos Fisherfaces Y LBPH Para El Reconocimento Facial En Diferentes Condiciones de Iluminación Y Poes, TACNA – 2015. http://repositorio.unjbg.edu.pe/handle/UNJBG/2479

Primeiro A. Autor Todas as iografias b devem limitar-se a um parágrafo que consiste nos seguintes: lista de diplomas sequencialmente ordenados, incluindo anos alcançados; locais de contratação sequencialmente ordenados concluindo com o emprego atual; associação com quaisquer revistas oficiais ou conferências; grandes realizações profissionais e/ou acadêmicas, ou seja, melhores prêmios de papel, bolsas de pesquisa, etc.; quaisquer informações de publicação (número de artigos e títulos de livros publicados); interesses atuais de pesquisa; associação com qualquer associação profissional e profissional. As informações de adesão ao autor, por exemplo, são membros do IEEE e da IEEE Computer Society, se aplicável, é notada no final da biografia.

Segundo B. Autor Jr. biografia aparece aqui.

Terceira biografia de C. Autor aparece aqui.