Curso de Ciência da computação
Universidade Estadual de Mato Grosso do Sul

ESTUDO E ANÁLISE DE TÉCNICAS PARA RECONHECIMENTO E ARMAZENAGEM ONLINE DE FACES

Leonel Ceolin Farias

Prof. MSc. André Chastel Lima (Orientador)

Dourados - MS
2018

ESTUDO E ANÁLISE DE TÉCNICAS PARA RECONHECIMENTO E ARMAZENAGEM ONLINE DE FACES

Leonel Ceolin Farias

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Leonel Ceolin Farias e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 17 de novembro de 2018

Prof. MSc. André Chastel Lima (Orientador)

Leonel Ceolin Farias

ESTUDO E ANÁLISE DE TÉCNICAS PARA RECONHECIMENTO E ARMAZENAGEM ONLINE DE FACES

Banca Examinadora:

Prof. Msc. André Chastel Lima (Orientador)
Área de Computação - UEMS

Prof. Dr. Osvaldo Vargas Jaques
Área de Computação - UEMS

Prof.a Dr.a Mercedes Rocío Gonzales Marques
Área de Computação - UEMS

Dourados - MS
17 de novembro de 2018

Este trabalho é dedicado aos meus pais Angela e Nilson que me proporcionaram a oportunidade de ingressar no ensino superior e me apoiaram durante todo o curso. Além deles, este trabalho também é dedicado aos meus amigos e familiares que torceram por mim durante essa jornada. Muito obrigado!

"You never know how strong you are until being strong is your only choice."

Bob Marley

"É necessário sempre acreditar que o sonho é possível. Que o céu é o limite e você, truta, é imbatível."

Racionais MC's

AGRADECIMENTOS

Agradecer primeiramente a Deus que permitiu o desenvolvimento deste trabalho e que tudo isso acontecesse.

Ao meu professor Orientador, Prof. Msc. André Chastel Lima, pela orientação, apoio, confiança, pela paciência, por suas correções e pela oportunidade de elaboração deste trabalho.

E a todos que direta ou indiretamente fizeram parte da minha formação, muito obrigado.

RESUMO

A utilização de tecnologias para o reconhecimento facial é algo que vem crescendo muito nos últimos anos, esse aumento se dá pelo simples motivo que a segurança oferecida pelo reconhecimento facial é maior do que se fosse protegido somente por senha. Levando isso em consideração, a possibilidade de ter um sistema que monitore o acesso de uma sala com essa tecnologia é algo realmente interessante e muito tentador.

O presente trabalho aplicou conceitos de reconhecimento e detecção facial baseados no métodos Eigenfaces, K-Nearest-Neighbors, LBPH(Local Binary Patterns Histograms), Fisherfaces e Viola-Jones, além de utilizar a técnica de análise de componentes principais (PCA) e análise de discriminante linear (LDA) para extrair os padrões existentes nos dados e agrupá-los da melhor maneira sem perder as informações importantes para desenvolver o software de reconhecimento facial.

Após a realização dos estudos dos métodos foi definida a linguagem de programação utilizada para o desenvolvimento do *software*, toda a implementação das rotinas de reconhecimento foi feita em Java e algumas instruções em SQL também foram utilizadas para o envio das imagens para o banco de dados.

Tendo os métodos estudados e linguagem de programação definidos o software foi desenvolvido e testado. Os testes foram realizados nos laboratórios do curso de ciência da computação da UEMS. Além dos testes realizados em laboratório foi feita uma avaliação dos algoritmos em cima da base de dados *yalefaces* que ajudou no processo de escolha do algoritmo mais eficaz no processo de reconhecimento. O algoritmo escolhido foi o LBPH que obteve uma taxa de 83,3 acertos em um conjunto de 100 imagens com um excelente valor de confiança.

O sistema de reconhecimento facial desse estudo tem a tarefa de detectar e reconhecer corretamente uma pessoa de acordo com sua base de dados. Esse processo é feito com um classificador de distância mínima onde tem-se uma variação do valor de segurança de acordo com as imagens cadastradas no banco.

Palavras-chave: Reconhecimento facial; Eigenfaces; K-Nearest-Neighbors; Análise dos Componentes Principais; LBPH; Fisherfaces; PCA.

ABSTRACT

Nowadays the use of facial recognition technologies is growing rapidly, the increase can be explain by the mainly reason which is the higher security provided by the facial recognition instead of only by password. Taking this into consideration, the possibility of having a system that secure the access of a room with this kind of technology is something really interesting and very fascinating.

The present work applied facial recognition and detection concepts based on the Eigenfaces, K-Nearest-Neighbors, LBPH (Local Binary Patterns Histograms), Fisherfaces and Viola-Jones methods, in the addition to using the main component analysis(PCA) and linear discriminat analysis (LDA) technique to extract the existing patterns in the data and group them in the best way without losing important information from the software.

After this, the methods' studies were carried out and right after the program language, which was used for the development of the software, was defined. All the recognition routines implementation were done in java and also SQL was used, as a secondary help tool to send the images to the database.

Having the first tasks done, the methods studies and program language defined, the software was developed and running. The samples were carried out in the UEMS computer science course laboratories. In addition to this, an evaluation of the algorithms was made on top of the database which helped to evaluate the most efficient algorithm. The LBPQ obtained a rate of 83,3 hits in a set of 100 images with an excelente value of confidence, showing that its performance was better in that situation.

The facial recognition system in this study has the task to correctly detect and recognize a person according to your database. This process requires a minimum distance classifier where there is a variation of the safety value according to the images in the database.

Keywords: Facial Recognition; Eigenfaces; K-Nearest-Neighbors; Principal Component Analysis; LBPH; Fisherfaces; PCA.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

PCA	Análise de Componentes Principais
LDA	Análise de Discriminante Linear
KNN	K-Nearest Neighbours
LBP	Local Binary Patterns
LBPH	Local Binary Patterns Histograms
OpenCv	Open Source Computer Vision Library
iOS	Sistema Operacional de Dispositivos da Apple
SQL	Structured Query Language

LISTA DE ILUSTRAÇÕES

1 INTRODUÇÃO

Humanos, consciente ou inconscientemente, participam de interações sociais em seu dia-a-dia. Essas interações são atos, ações ou práticas entre duas ou mais pessoas mutuamente orientadas uma para a outra. Elas podem ocorrer de diversas maneiras, como por exemplo, piscando, comendo, lendo, escrevendo ou caminhando. Dentre elas a mais importante é a visão, que desempenha o papel de estabelecer e manter as interações sociais. Tendo em vista que a maioria da troca de informações entre duas pessoas é não-verbal e se dá através de gestos faciais e corporais. (PARRA, 2013)

A capacidade em reconhecer faces e gestos desempenha um papel de grande importância em nossas vidas. Essa habilidade é desenvolvida muito cedo e afeta diretamente na maneira de como iremos nos relacionar na sociedade. Para nós humanos, identificarmos outra pessoa é uma tarefa simples, porque conseguimos reconhecer mesmo com pouca luz observando apenas uma parte da face ou até com algumas variações na mesma, mas traduzir essa habilidade para o meio computacional não é uma tarefa que consiga ser resolvida de forma trivial.

A compreensão desse processo de reconhecimento facial é de grande importância para que ele seja simulado posteriormente em um computador. Nas últimas décadas essa área tem sido muito explorada e estudada devido ao aumento nos requisitos de segurança em lugares públicos e privados. Também se dá, devido a disponibilidade de sistemas computacionais mais rápidos e eficientes.

Com a grande evolução da tecnologia foi possível aumentar a velocidade de processamento dos computadores, dessa forma abriu-se um leque de caminhos desconhecidos até pouco tempo atrás e gerando a capacidade para a realizar novas tarefas. Uma função que se destaca entre as demais é a de permitir que a máquina “enxergue” o mundo externo através de imagens, fazendo com que haja uma integração entre os meios em que ela se encontra inserida. Dentro das áreas de atuação da computação a visão computacional foi a mais influenciada através do avanço no processamento.

Visão computacional é a área da computação cujo o objetivo é modelar a realidade para o meio digital ou reconhecer objetos em imagens digitalizadas e com isso desenvolver teorias e métodos para a extrair informações relevantes contidas nessas imagens (Equipe Data Science Academy, 2017), sendo essas capturadas através de dispositivos imageadores, como câmeras de vídeo, scanners, radares entre outros. Essas técnicas de reconhecimento facial são amplamente usadas com diversas finalidades, seja para sistemas de

segurança, para o controle de frequência ou para acessar um computador portátil ou celular.

O trabalho apresentado tem suas informações divididas em capítulos. No capítulo 1 (um) é abordada a introdução do tema escolhido, junto com os objetivos do mesmo; No capítulo 2 (dois) é descrita a metodologia utilizada no estudo, além de apresentar as tecnologias e materiais utilizados no desenvolvimento; No capítulo 3 (três) o referencial teórico utilizado para o desenvolvimento do software. O capítulo 4 (quatro) fica reservado para análise e apresentação dos resultados, alcançados por meio das técnicas utilizadas; No capítulo 5 (cinco) é apresentada a conclusão dos estudos e; O capítulo 6 (seis) contém as referências bibliográficas.

1.1 Objetivos

1.1.1 Objetivo Central

O tema desse estudo é realizar a análise de imagens de vídeo feitas através da Webcam e a partir dela realizar comparações com imagens contidas no banco de dados do programa para identificar pessoas.

1.1.2 Objetivos Específicos

• Estudar métodos de reconhecimento facial.

• Estudar e definir qual a melhor linguagem de programação a ser utilizada para o desenvolvimento do software de reconhecimento facial.

• Escolher o melhor algoritmo para utilizar na técnica de reconhecimento.

• Escolher o banco de dados que seja mais adequado para a aplicação.

• Desenvolver um sistema de reconhecimento facial que seja eficaz.

• Realizar testes e comprovar a eficiência do sistema.

2 METODOLOGIA

O presente estudo é do tipo pesquisa tecnológica e está dividido em duas atividades principais, que são: pesquisa dos fundamentos teóricos e pesquisa de desenvolvimento tecnológico.

Durante as atividades de pesquisa dos fundamentos teóricos foram consultados livros, artigos e estudos sobre a área de detecção e reconhecimento facial, para assim obter uma melhor compreensão dos conceitos que estão envolvidos nesse trabalho. Estes conceitos têm grande importância porque foram eles que guiaram o desenvolvimento do trabalho para garantir que os resultados a serem obtidos sejam satisfatórios. Após o entendimento desses, está a pesquisa de quais tecnologias seriam utilizadas para a implementação do trabalho, elas encontram-se listadas nesse capítulo e descritas no capítulo 3.

Com relação a pesquisa de desenvolvimento tecnológico, é necessário para este trabalho realizar estudos aprofundados sobre técnicas e tecnologias utilizadas para o processo de implementação de um software de reconhecimento facial e como torná-lo real e funcional. Tornando possível determinar as melhores tecnologias para desenvolvimento do software de reconhecimento. Estas estarão dispostas no capítulo posterior com mais detalhes de utilização.

As tecnologias utilizadas para o desenvolvimento do software de reconhecimento são: softwares *OpenSource*[1]ou versões gratuitas de ferramentas disponíveis na web. (CORDEIRO, 2007)

Linguagem de programação, Tecnologias e Ferramentas utilizadas:

- Java
- Análise de Componentes Principais (PCA)
- OpenCV
- JavaCV
- Viola-Jones
- K Nearest Neighbors (KNN)
- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms (LBPH)

[1] Trata-se de software de código aberto, isto é, em que o código fonte está acessível para inspeção e é passível de manipulação; adicionalmente, e na maior parte dos casos, surge também com licenciamento livre de encargos. É a frequente associação destes dois aspectos, que não são típicos do software comercial, que confere a qualificação comum de 'livre' ao software de código aberto. (CORDEIRO, 2007)

- PostgreSQL

Para tornar real o trabalho foi utilizado um ambiente de testes com o intuito de servir como um experimento para as tentativas de implementação e compreensão das tecnologias. Neste ambiente foram instalados o Java, OpenCV e JavaCV além do desenvolvimento de uma aplicação simples. Nesta pequena aplicação foram realizados testes em laboratório com estudantes do curso de Ciência da Computação da Universidade Estadual de Mato Grosso do Sul para verificar a funcionalidade do programa. Foi conduzido através das imagens obtidas por webcam sendo analisadas com o uso dos algoritmos *K Nearest Neighbors*(KNN), *Eigenfaces* e também com a utilização de técnicas de análise dos componentes principais (PCA).

Onde após as análises das imagens o software deve tomar uma das decisões a seguir:

1 - caso a pessoa esteja catalogada no banco de dados cria-se um *log* informando dia e hora que ela esteve no local em que a câmera se encontra;

2 - caso contrário além de criar o *log* de aviso contendo dia e hora, a imagem da pessoa é salva em anexo a esse *log*.

3 REVISÃO DA LITERATURA

3.1 Detecção de Faces

Uma das tarefas mais importantes a serem realizadas em um Sistema de Reconhecimento de Faces é a de detectar a presença de uma face em uma determinada imagem.(LOPES, 2016) Feito esse processo de detecção, extrai-se as informações mais importantes contidas na face tendo em vista que, a maioria dos algoritmos utiliza essas informações como base para realizar o reconhecimento sendo assim, ao realizar a detecção antes da análise dessas características otimiza-se o processo de reconhecimento.

Para esse estudo foi utilizado o método de detecção baseado em aparência, isto é, não utilizam nenhum conhecimento a *priori* sobre o objeto ou características a ser detectada. Nesta classe de algoritmos surgem os conceitos de aprendizado e treinamento, uma vez que as informações necessárias para realizar a tarefa de detecção são retiradas do próprio conjunto de imagens sem intervenção externa.(LOPES, 2016) O algoritmo utilizado nesse estudo para a detecção de faces em uma imagem chama-se *Viola-Jones*.

3.1.1 K-Nearest Neighbours

O algoritmo *K-Nearest Neighbours* (KNN) é um método não paramétrico utilizado para a classificação e regressão. Em ambos os casos a entrada consiste dos k exemplos de treinamento mais próximos no espaço característico. Sua saída depende da finalidade do algoritmo, sendo ela para classificação ou regressão, no presente estudo, a saída utilizada será de classificação das imagens. (PARRA, 2013)

• Utilizando o *KNN* para a classificação, a saída é a associação das classes. Nesse caso o objeto é classificado por uma maioria de votos vizinhos, isto é, a classe mais comum presente ao redor do objeto será a qual ele irá pertencer. Se k=1, então o objeto é atribuído à classe daquele único vizinho mais próximo.

• Já quando utilizada para regressão, a saída é valor de propriedade para o objeto. Este valor é a média dos valores de seus k-vizinhos mais próximos.

Esse método possui o aprendizado baseado em exemplo, ou aprendizagem preguiçosa, onde a função é apenas aproximada localmente e toda computação é adiada até a classificação.

O *KNN* está entre os algoritmos mais simples de aprendizado de máquina.

Ambos métodos podem ser úteis para ponderar as contribuições dos vizinhos assim, os mais próximos contribuem para a média daqueles mais distantes.

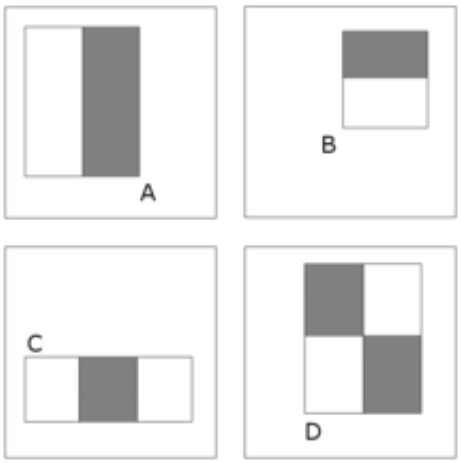
3.1.2 Viola-Jones

O algoritmo de *Viola-Jones* é um mecanismo amplamente utilizado para realizar o reconhecimento de objetos e faces. A sua principal propriedade é o seu treinamento lento mas com detecção rápida, sendo assim, o treinamento não interfere de forma negativa no rendimento do algoritmo. Ele utiliza os filtros de *Haar* e, por conta disso não realiza multiplicações, deixando o algoritmo mais eficiente. (Vocal Tecnologies, 2017)

Este método tem uma alta taxa de acerto no processo de reconhecimento facial e realiza esse procedimento com grande precisão. Com a utilização de falsos positivos, esse algoritmo tem um baixo custo computacional e por isso foi escolhido para ser implementado no presente estudo.

As características simples utilizadas são funções bases reminiscentes de *Harr* que foram utilizadas por (Papageorgiou et al. 1998). Mais especificamente, são utilizados três recursos que serão explicados a seguir. O *recurso do valor de dois retângulos* é a diferença entre a soma dos *pixels* entre duas regiões retangulares. As áreas têm o mesmo tamanho e forma e são horizontalmente ou verticalmente adjacentes (ver figura 1). O *recurso do valor de três retângulos* realiza o cálculo da soma entre os dois retângulos externos subtraídos da soma de um retângulo central. E por último o *recurso do valor de quatro retângulos* que calcula a diferença entre pares diagonais de retângulos. (VIOLA; JONES, 2003)

Figura 1 – Exemplo dos recursos de retângulos



Exemplo dos recursos de retângulo mostrados em relação a janela de detecção. A soma dos *pixels* que estão dentro dos retângulos brancos são subtraídos da soma dos *pixels* dentro dos retângulos cinzas. O recurso de dois retângulos é mostrado em (A) e (B). O de três retângulos em (C) e (D) o de quatro retângulos. Adaptado de (Vocal Tecnologies, 2017).

O cálculo desses recursos pode ser feito de forma rápida usando uma representação intermediária para a imagem, que será chamada de imagem integral. Essa imagem integral no local x,y contém a soma dos *pixels* acima e à esquerda de x,y inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.1}$$

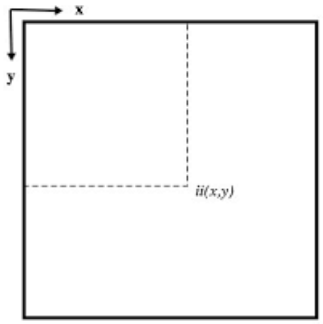
onde ii(x,y) é a imagem integral e i(x,y) é a imagem original (ver figura 2). Usando o seguinte par de recorrências:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{3.2}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{3.3}$$

Onde s(x,y) é a soma das linhas acumuladas, s(x, -1) = 0, e ii(-1, y) = 0) a imagem integral pode ser calculada em uma passagem sobre a imagem original.

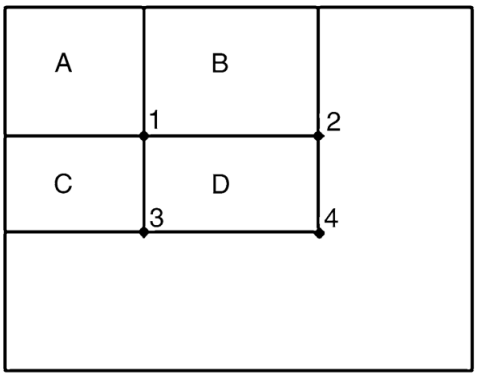
Figura 2 – Imagem integral



O valor da imagem integral no ponto (x,y) é a soma de todos os *pixels* acima e à esquerda. Adaptado de (Vocal Tecnologies, 2017).

Usando a imagem integral, qualquer soma retangular pode ser calculada em quatro referências de matriz (ver figura 3). Claramente a diferença entre duas somas retangulares pode ser calculada em oito referências. Desde que as características dos dois retângulos definidas acima envolvam as somas retangulares adjacentes então eles podem ser calculados em seis referências de matriz, oito no caso do *recurso do valor de três retângulos*, e nove para o caso do *recurso do valor de quatro retângulos*.

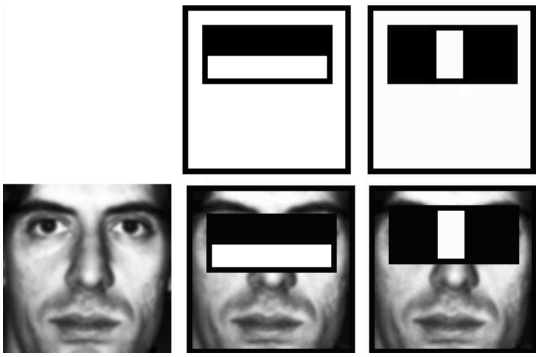
Figura 3 – Soma dos recurso dos retângulos



A soma dos *pixels* no retângulo D pode ser calculada com quatro referências de matriz. O valor da imagem integral no número 1 é a soma dos *pixels* no retângulo A. O valor no número 2 é A+B, em 3 é A+C, e em 4 é A+B+C+D. A soma dentro de D pode ser calcula como 4+1 – (2-3). Adaptado de (Vocal Tecnologies, 2017)

Para melhor compreensão de como são aplicados os filtros de Haar no processo de detecção de uma face basta olhar a figura 4 onde cada um dos recursos dos retângulos é aplicado e após isso obtêm-se a média da imagem.

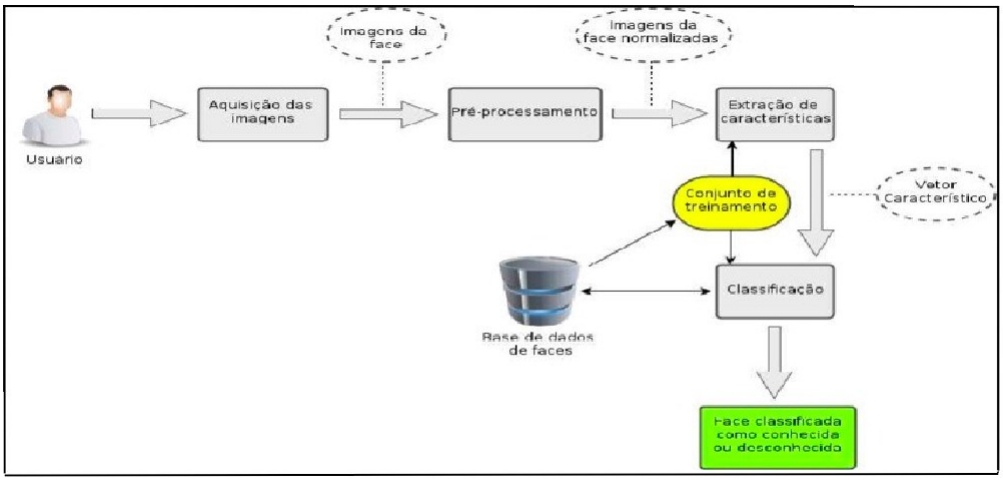
Figura 4 – Aplicação filtros de Haar



3.2 Reconhecimento Facial

O reconhecimento facial tem uma importância fundamental na nossa interação social, sendo extremamente importante para as nossas atividades do dia-a-dia. Atualmente, tal mecanismo motiva grandes esforços para pesquisas, porque, satisfaz vários critérios em escolher o método de solução biométrica ideal, é rápido, conveniente e remoto. Muitas técnicas estão sendo estudadas e precisam de melhoramentos, assim como o método do *Eigenfaces*(FONTANIVA, 2014) proposto nesse estudo. Abaixo segue a imagem com a representação da arquitetura do sistema de reconhecimento facial e quais as etapas presente nesse processo. Ver Figura 5.

Figura 5 – Arquitetura do sistema de reconhecimento facial



3.2.1 Análise de Componentes Principais (PCA)

Segundo (RIBEIRO; CHICHIA; MARANA, 2015),uma das principais abordagens para a identificação humana baseada em reconhecimento de face é a utilização da aparência. Nesta categoria os conceitos de aprendizado e treinamento emergem. Alguns métodos dessa abordagem são holísticos, ou seja, para encontrar as características da face eles consideram todos os *pixels* da imagem, apresentando, portanto, a desvantagem de possuir

alta dimensionalidade. Para a solução deste problema, métodos estatísticos podem ser utilizados para a redução de dimensionalidade, como a Análise de Componentes Principais. (em inglês, *Principal Component Analysis* – PCA).

A análise de componentes principais (PCA) é uma forma de identificar padrões em dados e de expressar essas informações de tal maneira a realçar suas características em comum e divergências. Além disso, há pontos positivos de que depois de encontrados os padrões é possível realizar a compressão dos elementos, reduzindo o número de dimensões sem que haja muita perda de informação.

PCA é uma técnica matemática que descreve um conjunto de informações usando "componentes principais" escrita como combinações lineares dos dados originais. Esses componentes são determinados em ordem decrescente de importância. Essa técnica tenta construir um pequeno conjunto de segmentos que resumem os dados originais, reduzindo a dimensionalidade dos mesmos, preservando os componentes mais significantes. (JUNIOR et al., 2013)

Como sendo uma das técnicas mais antiga e conhecida na análise multivariada e mineração de dados, o PCA foi apresentado por Pearson que a usou em um contexto biológico e, posteriormente, foi desenvolvido por Hotelling em trabalhos feitos na psicometria. Também foi criada independentemente por Karhunen no contexto da teoria da probabilidade e, generalizada por Loeve.

Os objetivos que se destacam dentro da PCA, são:

1. Redução da dimensionalidade;
2. Seleção de características: escolha dos componentes mais significantes.

No processamento de imagens e visão computacional, as representações do PCA são usadas para solucionar problemas da face e de objetos, tais como reconhecimento, detecção, determinação de formas, aparência e gestos.

O PCA é motivado e relacionado por dois problemas:

1. Dado um vetor x $\in R^n$, o objetivo é encontrar um subespaço linear m-dimensional (m $<$ n) que minimize a distância de x a esse subespaço. Esse problema surgiu na área de compressão de informações, cujo objetivo é representar todos os dados por um número reduzido de parâmetros.

2. Dado um vetor x $\in R^n$, a idéia é encontrar um subespaço linear m-dimensional que

armazene a máxima variância de x. Esse problema é associado à extração de características em que o objetivo é reduzir a dimensão dos dados preservando o máximo de informações.

Ambos os problemas são baseados nos autovalores e autovetores da matriz de covariância dos dados. O objetivo é encontrar um conjunto de vetores ortonormais vi que melhor descreva a distribuição dos dados de entrada.

Neste trabalho, a técnica PCA é utilizada não somente para a extração das informações mais importantes presentes na imagem, mas também para a comparação dessas com as presentes no banco de dados do software.

3.2.2 Eigenfaces

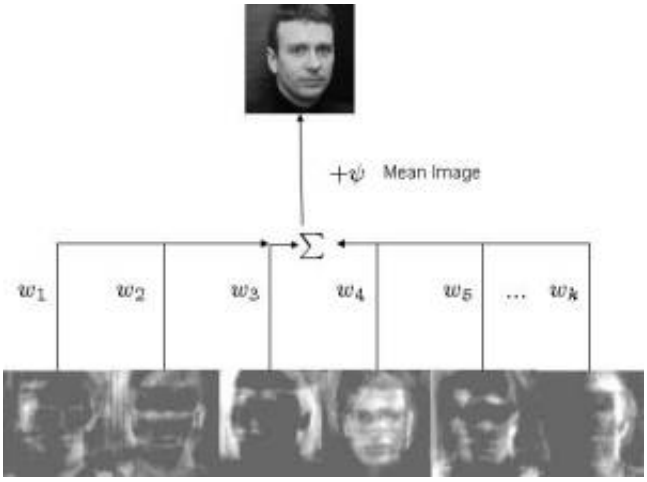
Na linguagem da teoria da informação, é necessário extrair as informações mais relevantes presentes na imagem, codificar da maneira mais eficiente possível e comparar com as imagens já presentes no banco de dados que possuem a codificação parecida. Uma abordagem simples para extrair essas informações contidas em uma imagem, é capturando a variação em uma coleção de imagens, independente do julgamento dos recursos, e usar esses dados para codificar e comparar cada imagem individualmente.

A técnica conhecida como *Eigenfaces* tem como conceito básico o fato de que faces não se encontram aleatoriamente em um espaço relativamente grande, sendo assim, podem ser representadas por espaços menores definidos espaços faciais.(FONTANIVA, 2014) De acordo com um estudo proposto por (DINIZ, 2016), diferente do citado acima, ele busca um conjunto de características que não dependa das formas geométricas da face (olhos, nariz, orelhas e boca) e utiliza toda a informação da representação facial. Baseada na teoria da informação, *Eigenfaces* identificam um pequeno número de características que são relevantes para diferenciar uma face de outra.

Essas características podem ser analisadas apenas com a variação dos valores assumidos pelos *pixels*, em um conjunto de imagens de faces. Os autovalores da matriz de covariância de um conjunto de imagens de faces descrevem a variação dos *pixels* em uma região diferente das imagens, ou seja, cada autovetor descreve a variação dos *pixels* associados a diferentes características faciais.(DINIZ, 2016)

O algoritmo **Eigenfaces** aplica PCA para codificar a imagem e então compará-la com um conjunto de imagens do banco de dados previamente codificados para encontrar o melhor par. A imagem **I** é primeiramente convertida para um vetor **p**, juntando as colunas. A imagem vetorial média (ver Figura 6) $\overline{a}$ contém o valor médio para cada *pixel* calculado durante o processo de treinamento, calculando a média no pixel ao longo de todas as imagens de treinamento. O vetor $\overline{p}$ subtraído pela média é então projetado para

Figura 6 – Imagem vetorial média Eigenfaces



o subespaço do autovetor multiplicando o vetor pela transposta da matriz subespaço, U^T. O vetor resultante, $\breve{p}$, é o vetor codificado.(PARRA, 2013)

3.2.3 Fisherfaces

Fisherface é outro método utilizado para o reconhecimento de faces por meio de características globais da face. De forma análoga ao *Eigenfaces*, esse método baseia-se também na redução de dimensionalidade do espaço de características. A projeção ótima neste caso é obtida através da Análise de Discriminante Linear (LDA - Linear Discriminant Analysis). (SOUZA, 2014)

Na Análise de Discriminante Linear, o conjunto de imagens faciais de treinamento é constituído por várias classes. Cada classe representa a identidade de uma pessoa e com base num conjunto de classes previamente conhecidas, o problema de reconhecimento é formulado de modo a determinar a que classe pertence uma determinada imagem desconhecida.(SOUZA, 2014)

O PCA é utilizado para comprimir uma imagem do rosto com perda mínima de dados a fim de minimizar o tempo correspondente. Porém, este não diferencia as informações discriminativas da imagem. O *Fisherfaces* resolve este problema utilizando o LDA, pois seu foco não está só na compressão da imagem, mas também na maximização da discriminação no processo. O reconhecimento de uma imagem de face se dá com a projeção nos **N** espaços dos vetores próprios criados e utilizando uma medida de semelhança, para efetuar a comparação com as outras imagens da face projetadas no mesmo espaço, utilizando um classificador específico ou a combinação de dois ou mais.(JESUS et al., 2015)

Diferença entre PCA e LDA segundo (SODHI, 2013):
A principal diferença entre a LDA e o PCA é que a LDA lida diretamente com a discriminação entre classes, enquanto o PCA lida com os dados em sua totalidade para a análise principal de componentes sem prestar atenção especial para a estrutura de classes subjacente.

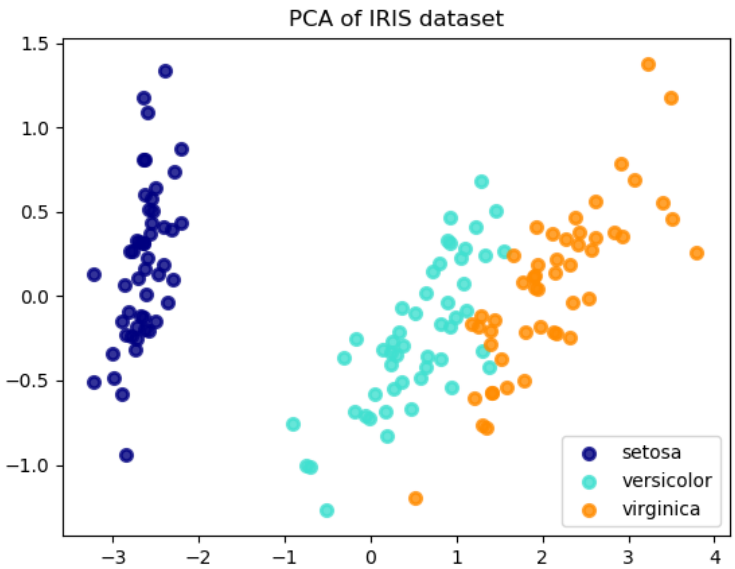
Na PCA, a forma e a localização dos conjuntos de dados originais muda quando transformado em um espaço diferente, enquanto o LDA não muda o local, mas apenas tenta fornecer mais classes de separação e desenha uma região de decisão entre as classes dadas.

O objetivo da Análise Linear Discriminante (LDA) é encontrar uma maneira eficiente de representar o espaço vetorial da face. O PCA constrói o espaço da face usando todo o treinamento da face dados como um todo e não usa as informações das classes da face.

Por outro lado, o LDA usa informações específicas das classes que melhor diferencia as mesmas entre si. LDA produz uma ótima função discriminante linear que mapeia a entrada para o espaço de classificação em que a identificação de classe desta amostra é decidida com base em algumas métricas como a distância Euclidiana.

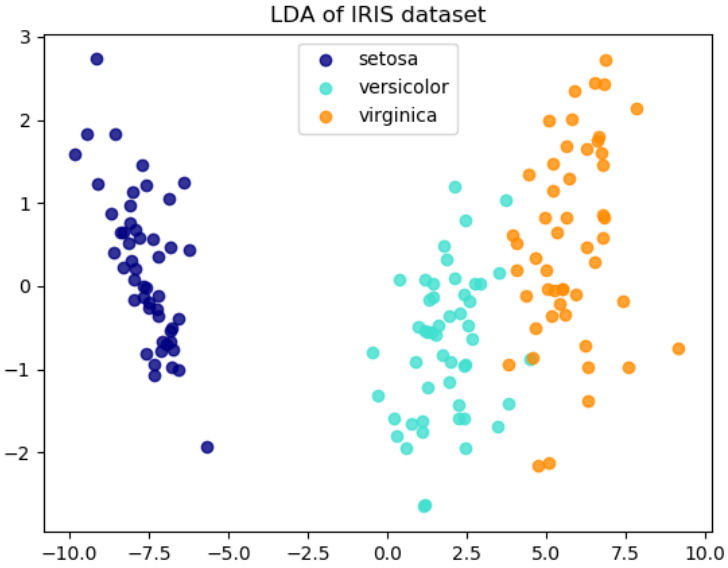
Imagens diferenciando PCA da LDA, olhar figuras 7 e 8:

Figura 7 – PCA



A PCA foca na maior variação entre as classes através da combinação dos atributos. Fonte: (scikit-learn developers, 2017)

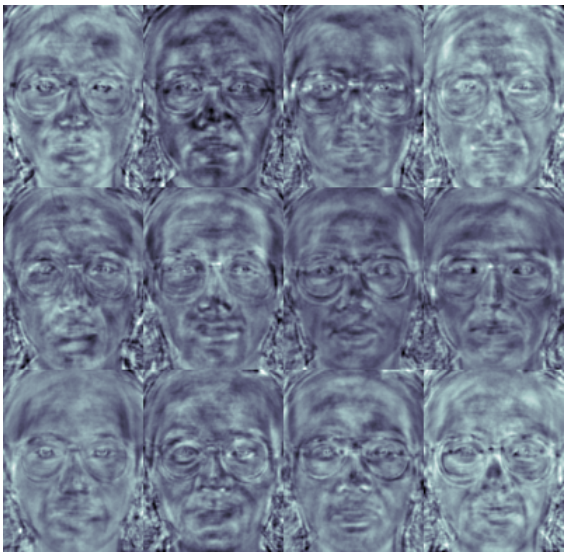
Figura 8 – LDA



A LDA foca na separação das classes, analisando cada uma de forma individual. Fonte (scikit-learn developers, 2017)

Para melhor entender o funcionamento do algoritmo *Fisherfaces* e qual o resultado gerado após sua aplicação na imagem, olhar figura 9 onde é possível identificar de forma mais clara as características individuais de uma face.

Figura 9 – Fisherfaces - face



3.2.4 Local Binary Patterns Histograms (LBPH)

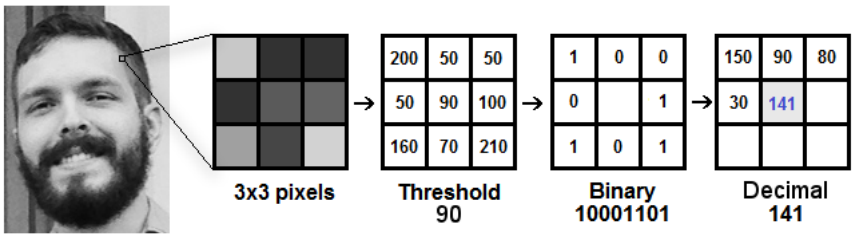
Conforme o estudo realizado por (LóPEZ, 2010) *Local Binary Patterns*(LBP) foram utilizados pela primeira vez para descrever texturas comuns onde relação espacial não foi tão

significativa quanto é para imagens faciais. Um rosto pode ser visto como uma composição de micro texturas dependendo da situação local. O LBP é basicamente dividido em dois descritores diferentes: um global e um local. O global é usado para discriminar a maioria dos objetos que não são faces, enquanto o segundo fornece informações faciais específicas e detalhadas que pode ser usado não apenas para selecionar faces, mas também para fornecer informações faciais para o reconhecimento.

Então, todo o descritor consiste em uma textura global e uma representação de textura local calculado dividindo a imagem em blocos e computando o histograma de textura para cada um. Os resultados serão concatenados em um vetor descritor geral. Em tal representação, a textura das regiões faciais é codificada pelo *LBP* enquanto a forma é recuperada pela concatenação de diferentes histogramas locais. Posteriormente, o vetor de caraterísticas será usado para alimentar um classificador adequado ou um esquema discriminativo para decidir um limite (uma medida para determinar a similaridade de um objeto com uma face) da imagem de entrada ou a identidade da face de entrada em caso de reconhecimento facial.

O *LBP* apareceu originalmente como um descritor de textura genérico. O operador atribui um rótulo para cada pixel de uma imagem, limitando uma vizinhança de 3x3 com o valor de pixel central e considerando o resultado como um número binário. O resultado binário será obtido pela leitura dos valores no sentido horário, começando pelo vizinho superior esquerdo, pode ser visto na figura 10. Após esse processo é gerado um histograma médio da imagem. Ver figura 11.

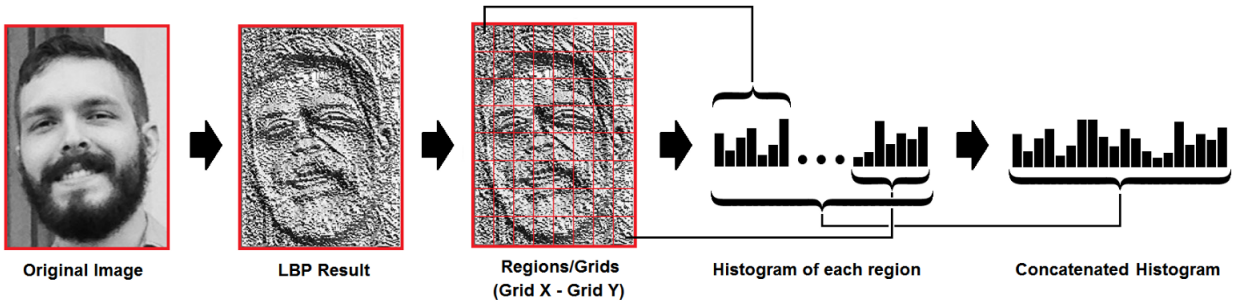
Figura 10 – Rotulagem LBPH



A etiqueta binária é lida no sentido horário a partir do vizinho superior esquerdo. Adaptado de (LóPEZ, 2010)

Em outras palavras, dada uma posição de pixel (xc, yc), *LBP* é definido como um conjunto binário de comparações de intensidades de pixel entre o *pixel* central e seus *pixels* adjacentes.

Figura 11 – Histogramas LBPH



3.3 Ferramentas

3.3.1 Java

Para o desenvolvimento de um software eficaz e obtenção de melhores resultados, a linguagem de programação escolhida para a codificação segundo (FLANAGAN, 2006), foi o Java por ser eficiente, mais fácil que as demais e possuir bibliotecas específicas para o reconhecimento facial. Possui o princípio de programação orientada a objetos e por ser derivada da linguagem (C) possuem a sintaxe parecida. Quando foi criada os projetistas focaram em eliminar os recursos complexos de outras linguagens orientadas a objeto que fizeram as mesmas serem arruinadas e tornar assim o Java mais eficiente. Ao mantê-la simples, a vida dos programadores foi facilitada e permitiu a eles que escrevessem o código de forma robusta e isento de falhas. Além das facilidades com relação a otimização do código, o Java possui uma plataforma que é compatível em sistemas operacionais diferentes, isto é, o programador pode desenvolver seu aplicativo no *Windows, Mac* ou *Unix* e ele será executado da mesma forma em todas elas desde que tenha sido desenvolvido dentro da plataforma Java. É a partir desse diferencial, de rodar em todos os sistemas operacionais, que surge o lema "escrever uma vez, executar em qualquer lugar".

3.3.2 OpenCV

OpenCv (Open Source Computer Vision Library) é uma biblioteca aberta voltada para o desenvolvimento de aplicações em visão computacional e compreende em seu escopo tópicos como o aprendizado de máquina, estrutura de dados e álgebra linear. (FONTANIVA, 2014)

Ela é distribuída sobre uma licença BSD[1], portanto é gratuita para uso no meio acadêmico e no meio comercial. Possui interfaces em C++, Java e Python e é compatível nos sistemas operacionais *Windows, Linux, Mac OS, iOS* e *Android*. Ela foi

[1] A licensa BSD é uma classe de licensas simples e liberal para softwares de computadores que foi originalmente desenvolvida na Universidade da California – Berkeley (UCB). A primeira vez que foi utilizada foi em 1980 para a Berkeley Source Distribuition (BSD), também conhecido como BSD UNIX, uma versão aprimorada do sistema operacional UNIX, que foi escrito em 1969 por Ken Thompson na Bell Labs.

desenvolvida buscando a melhor eficiência computacional e focada em aplicativos em tempo real. Escrita em C/C++ otimizado podendo, a biblioteca, tirar vantagens do processamento multi-core.(OpenCV team, 2018) Além de disponibilizar ferramentas de visão computacional, desde operações como filtro de ruídos até análise de movimentos, reconhecimento de padrões e reconstrução em 3D. Podendo ser dividida em cinco grupos: análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões e calibração de câmera e reconstrução 3D. (MARENGONI; STRINGHINI, 2009)

O *JavaCV*, por sua vez, é uma biblioteca que serve como interface para os métodos do *OpenCV* e, realiza as conversões necessárias entre a linguagem C e o Java. (KUROIWA; CARRO, 2015)

3.3.3 PostgreSQL

O PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código-fonte aberto que usa e estende a linguagem Structured Query Language [1](SQL ou Linguagem de Consulta Estruturada) combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas. (PostgreSQL, 1986)

O PostgreSQL, normalmente chamado de Postgres, é um sistema de gerenciamento de banco de dados do tipo objeto-relacional com ênfase em extensibilidade e em padrões de conformidade. Como um servidor de banco de dados, sua principal função é armazenar dados de forma segura, apoiando as melhores práticas permitindo a recuperação dos dados a pedido de outras aplicações de software. Ele pode lidar com cargas de trabalho que vão desde pequenas aplicações single-machine a aplicações de grande porte voltadas para a Internet, onde será utilizada de forma simultânea por vários usuários. (POSTGRESQL. . . , 2015)

Para o presente estudo o Postgres foi escolhido por conta de todas as funções disponibilizadas que foram listadas acima. No banco de dados foram criadas duas tabelas para armazenar as informações, onde em uma tabela são armazenados os dados referentes aos "logs"de todas as pessoas que estiveram no local onde a câmera está realizando o monitoramento e na outra foram armazenadas as imagens em binário das pessoas que não estão cadastradas no banco de dados e as informações das mesmas.

[1] Uma linguagem padrão de gerenciamento de dados que interage com os principais bancos de dados baseados no modelo relacional. (Significados, 2012)

4 IMPLEMENTAÇÃO

Na imagem abaixo seguem as importações das bibliotecas utilizadas no desenvolvimento do software. A biblioteca *OpenCV* disponibiliza várias funções que facilitam o processo de implementação, para identificar quais são as funções da *OpenCV* utilizadas no processo de implementação basta olhar a Figura 12 onde estão listadas todas as importações que foram necessárias para implementar o software. Onde o nome da função está no fim da linha de importação como por exemplo "core.Mat"que indica o uso da funcionalidade "OpenCvFrameConverter.ToMat"presente na linha 42 do programa que é responsável por realizar a conversão das imagens para matriz.

Figura 12 – Bibliotecas utilizadas

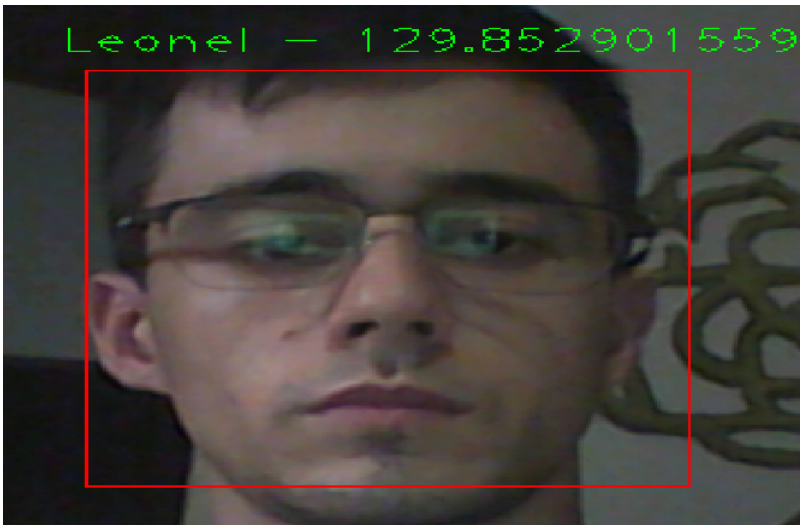
```
8  import java.io.IOException;
9  import java.sql.SQLException;
10 import java.util.Date;
11 import org.bytedeco.javacpp.DoublePointer;
12 import org.bytedeco.javacpp.IntPointer;
13 import static org.bytedeco.javacpp.opencv_core.FONT_HERSHEY_PLAIN;
14 import org.bytedeco.javacpp.opencv_core.Mat;
15 import org.bytedeco.javacpp.opencv_core.Point;
16 import org.bytedeco.javacpp.opencv_core.Rect;
17 import org.bytedeco.javacpp.opencv_core.RectVector;
18 import org.bytedeco.javacpp.opencv_core.Scalar;
19 import org.bytedeco.javacpp.opencv_core.Size;
20 import org.bytedeco.javacpp.opencv_face.FaceRecognizer;
21 import static org.bytedeco.javacpp.opencv_face.createLBPHFaceRecognizer;
22 import static org.bytedeco.javacpp.opencv_imgcodecs.imwrite;
23 import static org.bytedeco.javacpp.opencv_imgproc.COLOR_BGRA2GRAY;
24 import static org.bytedeco.javacpp.opencv_imgproc.cvtColor;
25 import static org.bytedeco.javacpp.opencv_imgproc.putText;
26 import static org.bytedeco.javacpp.opencv_imgproc.rectangle;
27 import static org.bytedeco.javacpp.opencv_imgproc.resize;
28 import org.bytedeco.javacpp.opencv_objdetect.CascadeClassifier;
29 import org.bytedeco.javacv.CanvasFrame;
30 import org.bytedeco.javacv.Frame;
31 import org.bytedeco.javacv.FrameGrabber;
32 import org.bytedeco.javacv.OpenCVFrameConverter;
33 import org.bytedeco.javacv.OpenCVFrameGrabber;
```

Na figura 13, na linha 42 temos a presença da variável **converteMat** que faz a conversão da imagem obtida pela *webcam* para uma matriz que será comparada com a matriz de imagens cadastradas no banco de dados. Na linha 55 a variável **detectorFace** recebe o arquivo xml responsável por identificar uma face frontalmente. Na linha 61 o algoritmo de reconhecimento escolhido é chamado e armazenado na variável **reconhecedor**, na linha 62 essa variável carrega o arquivo que foi treinado previamente e gerou uma matriz contendo os dados referentes as imagens cadastradas no banco de dados. Na linha 63 temos o limiar de segurança para classificar uma face, ao entrar em frente a *webcam* cada pessoa gera um limiar (threshold) próprio, no programa foi definido um limite de 150 que é referente as pessoas cadastradas no banco, sendo assim se outra pessoa entrar e gerar um número acima desse, não sera reconhecida. Ver figura 14.

Figura 13 – Classe principal

```
41  public static void rec() throws FrameGrabber.Exception, InterruptedException, SQLException, IOException {
42      OpenCVFrameConverter.ToMat converteMat = new OpenCVFrameConverter.ToMat();//realiza a conversão da imagem em uma matriz
43      OpenCVFrameGrabber camera = new OpenCVFrameGrabber(0);
44      camera.start(); //inicia a captura da webcam
45      String[] pessoas = {"", "Angela", "Leonel"};
46
47      String[] logReconhecidos = new String[20];
48      String[] reconhecidos = new String[20];
49      String[] desconhecidos = new String[20];
50      String[] logDesconhecidos = new String[20];
51      int contReconhecidos = 1;
52      int contDesconhecidos = 1;
53
54
55      CascadeClassifier detectorFace = new CascadeClassifier("src\\recursos\\haarcascade-frontalface-alt.xml"); //linha responsável p
56
57      /*FaceRecognizer reconhecedor = createEigenFaceRecognizer();
58      reconhecedor.load("src\\recursos\\classificadorEigenFaces.yml");
59      //reconhecedor.setThreshold(7000); //se o valor do threshold for menor do que o identificado na imagem
60      //o algoritmo reconhece a pessoa, caso seja maior, ele não reconhece*/
61      FaceRecognizer reconhecedor = createLBPHFaceRecognizer();
62      reconhecedor.load("src\\recursos\\classificadorLBPH.yml");
63      reconhecedor.setThreshold(150);
64      //reconhecedor.setThreshold(100);
65
66      CanvasFrame cFrame = new CanvasFrame("Reconhecimento", CanvasFrame.getDefaultGamma() / camera.getGamma());
```

Figura 14 – Threshold



Enquanto houver uma face diante da *webcam* os procedimentos dentro do laço serão executados (ver figura 15), que são eles: na linha 72 o frame capturado pela *webcam* é armazenado na variável **imagemColorida**, após isso na linha 73 esse frame é convertido para a escala de cinza. Na linha 77 caso o programa tenha identificado uma face diante da câmera a variável **dadosFace** recebe os dados da imagem que foi detectada, isto é as informações contidas dentro do quadrado de detecção mostrado na figura 16. Caso a pessoa esteja cadastrada no banco de dados e foi reconhecida, somente um *log* de que ela esteve no local monitorado será gerado e enviado para o banco de dados (ver figura 17). Este procedimento é realizado de chamando a função **sendLog**(ver figura 19) que

pertence a classe **EnviarBD**.

Caso a pessoa não esteja cadastrada no banco de dados o algoritmo de reconhecimento retornará -1 (menos um) e assim um *log* de desconhecido será criado e enviado para o banco de dados juntamente com uma imagem desse desconhecido. Isso é feito nas linhas 101 e 102 onde são chamadas as funções **send**(ver figura 18) e **sendLog**(Ver figura 19) presentes na classe **EnviarBD** que são as responsáveis pelo envio dos dados ao banco.

Figura 15 – Estrutura de repetição de controle

```
71      while ((frameCapturado = camera.grab()) != null) {
72          imagemColorida = converteMat.convert(frameCapturado); //está colocando o frameCapturado dentro da imagem colorida
73          cvtColor(imagemColorida, imagemCinza, COLOR_BGRA2GRAY); //converte da colorida para escala de cinza
74          RectVector facesDetectadas = new RectVector();//armazena todas a faces detectadas passando na webcam
75          detectorFace.detectMultiScale(imagemCinza, facesDetectadas, 1.1, 2, 0, new Size(100, 100), new Size(350, 350)); //detecç
76
77          for (int i = 0; i < facesDetectadas.size(); i++) { //entra no laço caso tenha reconhecido uma face
78              Rect dadosFace = facesDetectadas.get(i); //dadosFace recebe os dados da imagem que foi detectada, dentro do quadrado
79              rectangle(imagemColorida, dadosFace, new Scalar(0, 0, 255, 0)); //desenha o retangulo
80
81              Mat faceCapturada = new Mat(imagemCinza, dadosFace); //faceCapturada recebe somente os dados presentes dentro do qua
82              resize(faceCapturada, faceCapturada, new Size(160, 160));//redimensiona as imagens para ajudar no processo de reconh
83
84              IntPointer rotulo = new IntPointer(1); //para identificar qual é rotulo da imagem, se é a pessoa 1 ou 2
85              DoublePointer confianca = new DoublePointer(1);
86              reconhecedor.predict(faceCapturada, rotulo, confianca); //quando executar o predict irá passar o rotulo(1 ou 2) e a
87              String nome;
88
89              int predicao = rotulo.get(0); //resposta final, na posição 0(zero) se encontra o resultado, a qual classe a imagem p
90              Date data = new Date();
91
92              if (predicao == -1) { //nao conseguiu identificar a pessoa
93                  nome = "Desconhecido";
94                  imwrite("src\\desconhecidos\\" + nome + "." + contDesconhecidos + ".jpg", faceCapturada);
95                  logDesconhecidos[contDesconhecidos] = nome + "-" + contDesconhecidos + "-" + data.toString(); //log desconhecido
96                  desconhecidos[contDesconhecidos] =  nome + "." + contDesconhecidos + ".jpg"; //salvando imagem no vetor de desco
97
98                  System.out.println("LOG - " + logDesconhecidos[contDesconhecidos]);
99                  System.out.println(desconhecidos[contDesconhecidos]);
100
```

Figura 16 – Detecção da face

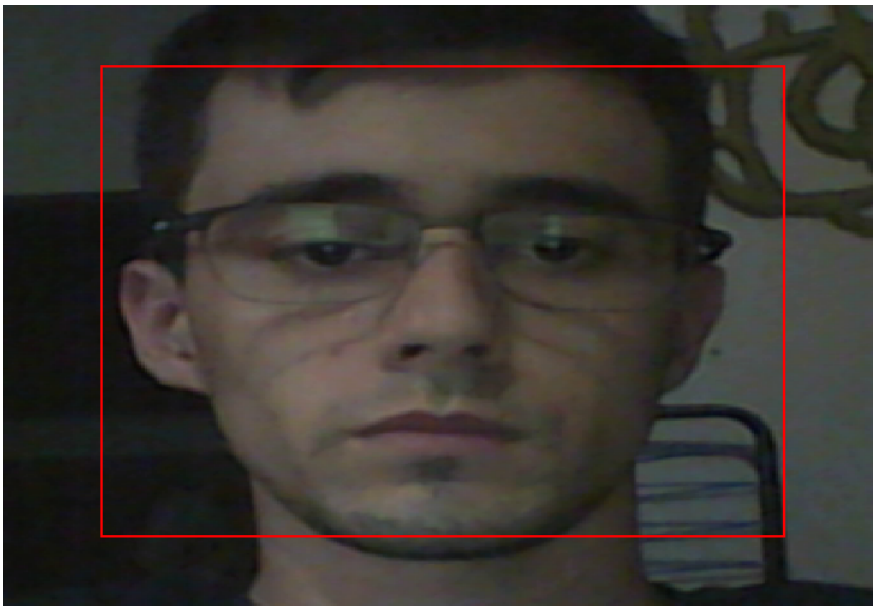


Figura 17 – Continuação da estrutura de controle

```
101                 EnviarBD.send(desconhecidos[contDesconhecidos]);
102                 EnviarBD.sendLog(logDesconhecidos[contDesconhecidos]);
103                 contDesconhecidos++;
104             } else {
105                 //salva na variável nome, o nome da pessoa que foi reconhecida e a data que esteve no local
106                 nome = pessoas[predicao];// + " - " + confianca.get(0);
107                 //imwrite("src\\desconhecidos\\" + nome +".jpg", faceCapturada);
108                 logReconhecidos[contReconhecidos] = nome + '-' + data.toString();
109                 //reconhecidos[contReconhecidos] = nome + '.' + "jpg";
110                 System.out.println("LOG - " + logReconhecidos[contReconhecidos]);
111                 //EnviarBD.send(reconhecidos[contReconhecidos]);
112                 EnviarBD.sendLog(logReconhecidos[contReconhecidos]);
113                 contReconhecidos++;
114                 //TimeUnit.SECONDS.sleep(1);
115             }
116 
117             int x = Math.max(dadosFace.tl().x() - 10, 0);
118             int y = Math.max(dadosFace.tl().y() - 10, 0);
119             putText(imagemColorida, nome, new Point(x, y), FONT_HERSHEY_PLAIN, 1.4, new Scalar(0, 255, 0, 0));
120         }
121         if (cFrame.isVisible()) {
122             cFrame.showImage(frameCapturado);
123         }
124     }
125     cFrame.dispose();
126     camera.stop();
127 
```

Figura 18 – Função send

```
22  public class EnviarBD {
23  
24      public static void send(String nome) throws SQLException, IOException {
25          Connection connection = null;
27          PreparedStatement statement = null;
28          FileInputStream imageInputStream = null;
29          try {
30              // retrieve connection
31              connection = DriverManager.getConnection("jdbc:postgresql://localhost:5432/imagedb", "leonel", "leonel"
32              // get statement object
33              statement = connection.prepareStatement("insert into teste(imagem, nomeimagem) values(?, ?)");
34              // create an input stream pointing to the image file to store
35                  imageInputStream = new FileInputStream(new File("src\\desconhecidos\\" + nome));
36              // inform the statement that first parameter in the query is of binary type
37              statement.setBinaryStream(1, imageInputStream); //dps do "statement" primeiro é o tipo da variável
38              //que será setada .string .setbinary // 1 valor referente a imagem
39              statement.setString(2, nome);//referente ao nome da imagem no banco para buscar e ser retornada posteri
40              // execute query
41              statement.execute();
42          } catch (SQLException sqe) {
43              throw sqe;
44          } finally {
45              // close resources
46              if (connection != null) {
47                  connection.close();
48              }
49              if (imageInputStream != null) {
50                  imageInputStream.close();
51              }
```

Figura 19 – Função sendLog

```
55      public static void sendLog(String nome) throws SQLException, IOException {
56          Connection connection = null;
58          PreparedStatement statement = null;
58          FileInputStream imageInputStream = null;
59
60          try {
61              // retrieve connection
62              connection = DriverManager.getConnection("jdbc:postgresql://localhost:5432/imagedb", "leonel", "leonel");
63              // get statement object
64              statement = connection.prepareStatement("insert into relatorio(log) values(?)");
65              statement.setString(1, nome);//referente ao log
66              // execute query
67              statement.execute();
68          } catch (SQLException sqe) {
69              throw sqe;
70          } finally {
71              // close resources
72              if (connection != null) {
73                  connection.close();
74              }
75              if (imageInputStream != null) {
76                  imageInputStream.close();
77              }
78          }
79      }
80  }
```

5 ANÁLISE DOS RESULTADOS

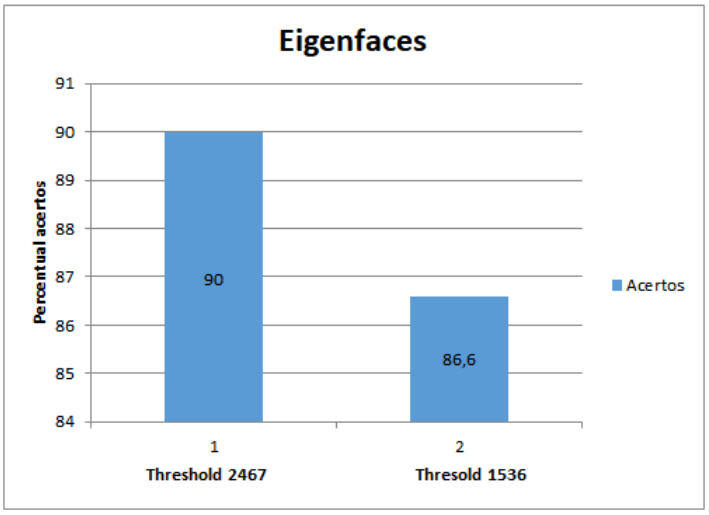
Os procedimentos e técnicas aprendidos ao longo do estudo foram utilizados para desenvolver o software de reconhecimento, os testes para verificar a funcionalidade do mesmo foram realizados nos laboratórios da universidade do curso de ciência da computação com os estudantes de ciência da computação.

Para o realizar os testes, o computador foi deixado no local que seria monitorado e os estudantes entravam no mesmo para que o software realizasse o procedimento de detecção e reconhecimento da facial.

Para escolher qual algoritmo de reconhecimento seria utilizado no *software* foram realizados testes com os algoritmos citados e descritos anteriormente *Eigenfaces*, *Fisherfaces* e *Local Binary Patterns Histograms*. Foi medido a eficácia de cada um através de testes com variações na luz do ambiente, posicionamento da pessoa diante da câmera, também foi utilizada a base de dados *yalefaces* que disponibiliza o conjunto de 133 imagens de treinamento e 30 imagens de teste para realizar a avaliação dos algoritmos de reconhecimento facial. Após esse processo foram obtidos os resultados abaixo.

O algoritmo *Eigenfaces* por ser muito sensível a variação de luz no ambiente foi descartado para o *software* final, tendo em vista que essa variação na luminosidade afeta diretamente a eficácia no processo de reconhecimento realizado por este algoritmo. Além do *threshold*(limiar de segurança) ter um valor muito alto que não é bom para a eficácia do software, quanto menor o *threshold* mais eficaz é o reconhecimento. Eficácia do *Eigenfaces*(ver figura 20).

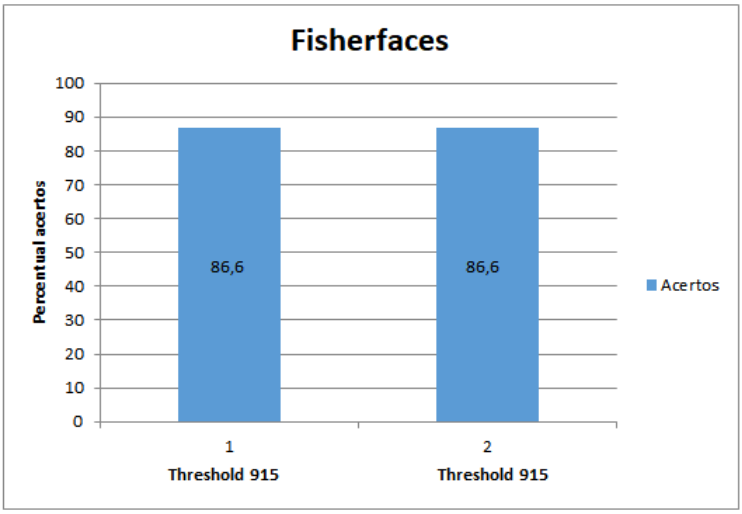
Figura 20 – Eigenfaces



O percentual de acertos referente ao número 1 no gráfico se dá quando utilizamos o padrão de 50 componentes de características faciais e o no número 2 quando utilizamos somente 30 componentes.

O algoritmo *Fisherfaces* por ser análogo ao *Eigenfaces* também foi descartado mas realizando uma comparação entre os resultados obtidos pelos dois, o *Fisherfaces* tem um rendimento melhor, isto é, realiza o processo de reconhecimento de forma mais eficaz e com um *threshold* menor que o *Eigenfaces* (ver figura 21), porém ao ser comparado com o *Local Binary Patterns Histograms* ambos tem um desempenho inferior e por esse motivo o algoritmo de reconhecimento utilizado no software final foi o LBPH.

Figura 21 – Fisherfaces

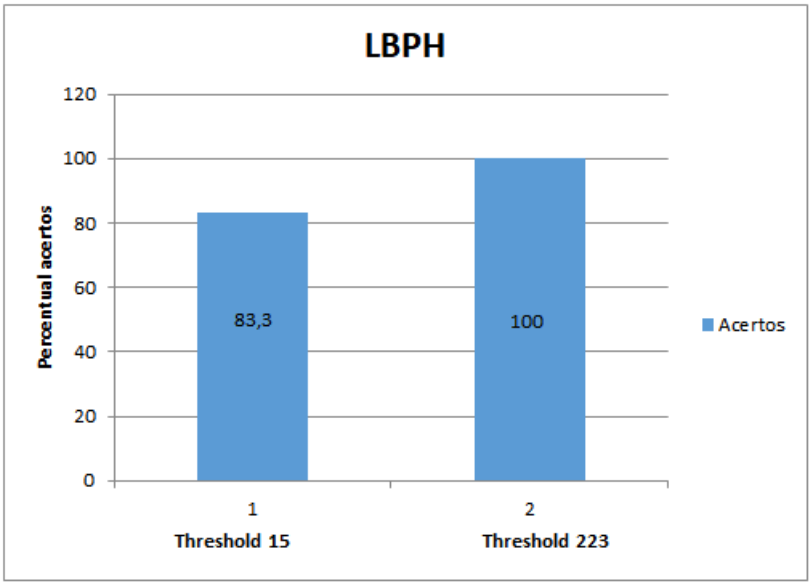


O percentual de acertos referente ao número 1 no gráfico se dá quando utilizamos o padrão de 50 componentes de características faciais e o no número 2 quando utilizamos

somente 30 componentes. Por utilizar a LDA o resultado nesse algoritmo foi o mesmo em ambos os casos, porém diferentemente do *Eigenfaces* o *threshold* já é melhor.

O LBPH por utilizar uma matriz binária para representar as imagens tem uma maior eficácia quando utilizado os seus parâmetros padrões e a variação da luminosidade não afeta o processo de reconhecimento assim como afeta os demais algoritmos, porque ao modificar a variação da luz no ambiente todos os elementos da matriz serão alterados na mesma proporção o que não afetará no resultado final do reconhecimento. Eficácia do LBPH (ver figura 22).

Figura 22 – LBPH



A quantia de acertos referente ao número 1 no gráfico se dá quando utilizamos o padrão de reconhecimento do LBPH, que são raio, *neighbors* (vizinhos), grid-x que é o número de células na horizontal, grid-y que é o número de células na vertical e o *threshold*(limiar de segurança), nessa ordem (2, 9, 9, 9, 1) e no número 2 quando utilizamos (12, 10, 15 15, 1). Com isso observa-se que ao aumentar o número de vizinhos e do raio, diminui o *threshold* do algoritmo no processo de reconhecimento.

O algoritmo teve uma média de 83,3 acertos a cada 100 imagens detectadas, dentro dessas imagens estão presentes alguns falsos positivos,falsos negativos e os satisfatórios.

Falsos positivos, isto é a pessoa estava cadastrada no banco de dados mas o programa não a identificou e marcou como desconhecido. Falsos positivos são aceitos uma vez que o programa não identificou uma pessoa que está cadastrada no banco e realizou a captura de uma foto da mesma, com isso após a verificação das imagens pode-se confirmar que a pessoa acessou o local.

Falsos negativos, o programa reconheceu de forma equivocada uma pessoa que não está cadastrada no banco e a marcou como cadastrada, gerando assim um problema grave pois uma foto dessa pessoa deveria ser armazenada e não foi. A ocorrência desses é inevitável pois a eficiência no reconhecimento depende da qualidade da imagem que está sendo obtida através da *webcam* e a mesma não fornece uma imagem com bons padrões de qualidade, tendo em vista que a pessoa estará em movimento durante o processo de detecção e reconhecimento.

Os resultados satisfatórios foram aqueles em que o programa funcionou de maneira correta, reconhecendo as pessoas cadastradas no banco e enviando o *log* da mesma para o banco de dados de *log* e as não cadastradas colocando como desconhecido e enviando um *log* de texto e uma imagem em binário para o banco de dados.

6 CONCLUSÃO

Fazer com que um computador realize a tarefa de detectar, reconhecer e nomear uma pessoa como nós humanos fazemos é uma tarefa complexa, o que para nós é algo simples para a máquina envolvem centenas de cálculos e comparações matemáticas. Com o software de reconhecimento implementado ele pode ser aplicado a várias situações cotidianas, para o desbloqueio da tela do celular ou computador, para o monitoramento de salas, controle de acesso, entre outros.

Durante a implementação do software, realizar os processos de detecção e reconhecimento facial foi um processo problemático, pois muitos dos resultados obtidos não tinham a precisão que se desejava ter. Os resultados não eram precisos e o software identificava de forma equivocada as pessoas em frente a câmera, o que não era o esperado, sendo assim o software ainda não estava identificando com a precisão desejada, isto é, de forma correta.

O processo de detecção facial apesar de ter um ótimo funcionamento para identificar uma face diante da câmera pode apresentar falhas dependendo dos parâmetros passado para realizar a detecção, caso o tamanho mínimo da imagem que deseja-se detectar seja muito pequeno o software pode detectar faces onde na verdade não há nada, gerando assim falsos positivos na detecção que são diferentes dos falsos positivos no reconhecimento.

A qualidade das imagens geradas pela câmera é um fator de grande importância, juntamente com essa qualidade a interferência de luz no ambiente são fatores que interferem diretamente no processo de reconhecimento, uma vez que ao captar imagens com baixa qualidade ou forte incidência de luz o resultado final não terá a precisão desejada.

No processo de reconhecimento as fotos utilizadas no treinamento são fotos frontais com algumas variações nas expressões faciais. Para o programa realizar a tarefa de reconhecer uma pessoa, a imagem obtida através da *webcam* tem de ser semelhante as imagens cadastradas no processo de treinamento, por isso a qualidade dessas imagens é tão importante, para que o algoritmo trabalhe de maneira eficaz. Para evitar os falsos negativos, isto é, o reconhecimento de forma equivocada de uma pessoa que não está cadastrada no banco, o fator de segurança(threshold) tem de estar configurado de forma correta.

Mesmo configurando todos os padrões do algoritmo e de segurança, controlando a luminosidade do local e utilizando câmeras que capturem imagens com melhor qualidade o

software é sujeito a falhas e pode identificar de forma incorreta a face que está diante da câmera.

Para trabalhos futuros podem ser alterados os parâmetros dos algoritmos, fator de segurança, trabalhar com a luminosidade. Podem ser alterados os algoritmos utilizados no software, as imagens obtidas através da câmera podem ser tratadas previamente antes de passar pelo processo de reconhecimento, com isso melhorando ainda mais a eficácia do programa tornando-o mais seguro.

REFERÊNCIAS

CICHOKI, A.; AMARI, S. ichi. *Adaptive blind signal and image processing: Learning algorithms and application*. [S.l.]: John Wiley Sons LTD, 2002.

CORDEIRO, M. I. *Código aberto e livre acesso: uma nova cultura na gestão de recursos?* [S.l.]: Biblioteca Nacional de Portugal, Lisboa, 2007.

DINIZ, F. A. *Um estudo empírico de um sistema de reconhecimento facial utilizando o classificador KNN*. 2016. Revista Brasileira de Computação Aplicada. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/5227>. Acesso em: 28 Abr 2018.

Equipe Data Science Academy. *O que é visão computacional?* 2017. Data Science Academy. Publicado em: 24 Jan 2017. Disponível em: <http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>. Acesso em: 28 Abr 2018.

FLANAGAN, D. *JAVA: O guia essencial*. 5. ed. [S.l.]: Bookman, 2006.

FONTANIVA, G. A. R. *Monitoramento de ambientes e controle de frequência acadêmica através de visão computacional*. 2014. Repositório Digital UFFS. Publicado em: 2014. Disponível em: <https://rd.uffs.edu.br/handle/prefix/1076>. Acesso em: 28 Abr 2018.

JESUS, L. et al. *Analise de Métodos de Processamento de Imagens para Reconhecimento Facial utilizando Fisherfaces em Imagens sob Condições Desfavoráveis*. 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wticgerbase/2015/009.pdf>. Acesso em: 10 Out 2018.

JUNIOR, F. das C. L. et al. *RedFace: um sistema de reconhecimento facial baseado em técnicas de análise de componentes principais e autofaces: comparação com diferentes classificadores*. 2013. Revista Brasileira de Computação Aplicada. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/2627/2188>. Acesso em: 28 Abr 2018.

KUROIWA, B. T.; CARRO, S. A. *Detecção de intrusão com reconhecimento facial em imagens geradas por câmeras de segurança*. 2015. Colloquium Exactarum. Disponível em: <http://revistas.unoeste.br/revistas/ojs/index.php/ce/article/viewArticle/1424>. Acesso em: 28 Abr 2018.

LOPES, E. C. *Detecção de Faces e características faciais*. 2016. Disponível em: <http://www.pucrs.br/facin-prov/wp-content/uploads/sites/19/2016/03/tr045.pdf>. Acesso em: 04 Out 2018.

LóPEZ, L. S. *Local Binary Patterns applied to Face Detection and Recognition*. 2010. Disponível em: <https://upcommons.upc.edu/bitstream/handle/2099.1/10772/PFC_LauraSanchez_%28LBP_applied_to_FaceDetection%26Recognition%29.pdf?sequence=1&isAllowed=y>. Acesso em: 08 Out 2018.

MARENGONI, M.; STRINGHINI, D. *Tutorial: Introdução à Visão Computacional usando OpenCV*. 2009. RITA: Revista de Informática Teórica e Aplicada. Disponível em: <http://seer.ufrgs.br/rita/article/view/rita_v16_n1_p125/7289>. Acesso em: 1 Mai 2018.

NETO, E. L. A. *Sistemas de indentificação pessoal utilizando tecnicas de reconhecimento e verificação facial automáticas*. 1997. Repositório da Produção Científica e Intelectual da Unicamp. Publicado em: 1997. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/259421>. Acesso em: 28 Abr 2018.

OpenCV. *Face Detection using Haar Cascades*. 2018. OpenCV. Disponível em: <https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html>. Acesso em: 19 Mai 2018.

OpenCV team. *Eigenfaces for Recognition*. 2018. OpenCV. Disponível em: <https://opencv.org/>. Acesso em: 1 Mai 2018.

PARRA, D. E. *Comparação entre algoritmos de reconhecimento de face no contexto de acessibilidade*. 2013. Repositório da Produção Científica e Intelectual da Unicamp. Publicado em: 2013. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/275561>. Acesso em: 28 Abr 2018.

PostgreSQL. *What is postgresql ?* 1986. PostgreSQL. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 08 Out 2018.

POSTGRESQL tutorial. 2015. DevMedia. Disponível em: <https://www.devmedia.com.br/postgresql-tutorial/33025>. Acesso em: 08 Out 2018.

P.QUINTILIANO; GUADAGNI, R.; ROSA, A. S. *Practical Procedures to Improve Face Recognition Based on Eigenfaces and Principal Component Analysis*. 2001. Disponível em: <http://ijofcs.org/quintiliano-papers-practical-proc-fr.PDF>. Acesso em: 15 Abr 2018.

RIBEIRO, I.; CHICHIA, G.; MARANA, A. N. *Reconhecimento de Faces Utilizando Análise de Componentes Principais e Transformada Census*. 2015. Disponível em: <https://www.researchgate.net/profile/Aparecido_Marana/publication/268379344_Reconhecimento_de_Faces_Utilizando_Analise_de_Componentes_Principais_e_a_Transformada_Census/links/54c238e50cf256ed5a8c64cf/Reconhecimento-de-Faces-Utilizando-Analise-de-Componentes-Principais-e-a-Transformada-Census.pdf>. Acesso em: 28 Abr 2018.

scikit-learn developers. *Comparison of LDA and PCA 2D projection of Iris dataset*. 2017. Disponível em: <https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html>. Acesso em: 17 Nov 2018.

Significados. *O que é SQL:*. 2012. Significados. Disponível em: <https://www.significados.com.br/sql/>. Acesso em: 08 Out 2018.

SILVA, A. F. da. *Reconhecimento de Faces via PCA: Análise de desempenho*. 2006. Repositório UFU. Publicado em: 18 Ago 2006. Disponível em: <https://repositorio.ufu.br/handle/123456789/12588>. Acesso em: 30 Abr 2018.

SODHI, K. S. *Face Recognition Using PCA, LDA and Various Distance Classifiers*. 2013. JOURNAL OF GLOBAL RESEARCH IN COMPUTER SCIENCE. Disponível em: <http://www.rroij.com/open-access/face-recognition-using-pca-lda-and-various-distance-classifiers-30-35.pdf>. Acesso em: 17 Nov 2018.

SOUZA, F. L. de. *Classificador Fisherface Fuzzy para o Reconhecimento de Faces*. 2014. Repositório Unesp. Disponível em: <https://repositorio.unesp.br/bitstream/handle/11449/110600/000791000.pdf;sequence=1>. Acesso em: 10 Out 2018.

The Linux Information Project. *BSD License Definition*. 2005. Disponível em: <http://www.linfo.org/bsdlicense.html>. Acesso em: 1 Mai 2018.

TURK, M.; PENTLAND, A. *Eigenfaces for Recognition*. 1991. Disponível em: <http://www.face-rec.org/algorithms/pca/jcn.pdf>. Acesso em: 1 Mai 2018.

VIOLA, P.; JONES, M. *Robust Real-Time Face Detection*. 2003. Computational Vision. Disponível em: <http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf>. Acesso em: 19 Mai 2018.

Vocal Tecnologies. *Face Detection using Viola-Jones Algorithm*. 2017. VOCAL. Disponível em: <https://www.vocal.com/video/face-detection-using-viola-jones-algorithm/>. Acesso em: 19 Mai 2018.