

DETECÇÃO E RECONHECIMENTO FACIAL UTILIZANDO JAVACV E O ALGORITMO LBPH

Wellington Sousa Aguiar - Centro Universitário Estácio Do Ceará - wellington@tecsist.com

Eliseu Lucena Barros - Centro Universitário Estácio Do Ceará - eliseu42lucena@gmail.com

Daniel Nascimento Barroso - Centro Universitário Estácio do Ceará - barrozim@gmail.com

Alexandre Bonadiman Angeli - Faculdade Estácio de Sá de Vitória, Espírito Santo - alexandre.angeli@outlook.com

O reconhecimento facial ou de objetos é um campo de estudo do processamento de imagens que vem crescendo a cada dia, principalmente na área de segurança. Através de técnicas para reconhecimento pode-se prever possíveis ações maliciosas onde dificilmente seriam observadas e/ou analisadas pelo olho humano. O objetivo deste estudo é apresentar um software e o algoritmo utilizado para realizar a detecção e o reconhecimento facial, sendo essa uma técnica que pode ser utilizada na segurança e em várias outras aplicações. O método de pesquisa utilizado foi o qualitativo para avaliar a utilização da solução através da aplicação do software. A pesquisa bibliográfica também foi usada, constituída por pesquisas em livros, artigos científicos, teses e dissertações, para melhor compreender o funcionamento no algoritmo de reconhecimento LBPH (*Local Binary Patterns Histograms*). Foi desenvolvido um software utilizando a linguagem de programação Java versão 8 update 151, capaz de capturar as imagens, realizar o treinamento e fazer o reconhecimento de faces através da webcam. Através da análise dos resultados foi possível entender o funcionamento, analisar os pontos fracos do algoritmo e entender a melhor forma de elaborar o treinamento para reconhecimento facial. Com este estudo podemos concluir e entender que o modelo LBPH, assim como outros algoritmos, possuem pontos fortes e fracos no reconhecimento de imagens, apresentam resultados satisfatórios, mas ainda necessitam de estudos para a evolução destas técnicas.

Palavras-chaves: Reconhecimento facial, Detecção facial, Processamento de imagem, algoritmo LBPH.

FACIAL DETECTION AND RECOGNITION USING JAVACV AND LBPH ALGORITHM

Facial or object recognition is an image processing field of study that is growing every day, especially in the area of security. Through recognition techniques one can predict possible malicious actions where they would hardly be observed and / or analyzed by the human eye. The objective of this study is to present a software and the algorithm used to perform the facial detection and recognition, which is a technique that can be used in security and in several other applications. The research method used was the quantitative to evaluate the use of the solution through the application of the software. The bibliographic research was also used, consisting of researches in books, scientific articles, thesis and dissertations, to better understand the operation in the algorithm of recognition LBPH (*Local Binary Patterns Histograms*). Software was developed using the Java version 8 update 151 programming language, capable of capturing images, performing training and doing face recognition through the webcam. Through the analysis of the results it was possible to understand the functioning, to analyze the weak points of the algorithm and to understand the best way to elaborate the training for facial recognition. With this study we can conclude and understand that the LBPH model, as well as other algorithms, have strong and weak points in image recognition, present satisfactory results, but still need studies for the evolution of these techniques.

Keywords: Face recognition, Face detection, Image processing, LBPH algorithm.

INTRODUÇÃO

Os casos de fraudes, acidentes, roubos e outras ações maliciosas vem crescendo cada vez mais em nosso país, muitas pessoas sentem medo de andar de ônibus ou de caminhar a noite em uma praça, pois não sabem quem está ao seu lado, podendo ser este sujeito um assaltante ou alguém querendo aplicar um golpe. Essa mesma suspeita também se dá na área da tecnologia da informação que através da sua evolução permite que pessoas não autorizadas realizem operações fraudulentas sem grandes dificuldades, como pagar contas, inserir crédito no telefone pré-pago, realizar transações financeiras, entre outras atividades. Essas comodidades são muito boas e facilitam a vida de muitas pessoas na vida profissional ou privada, ganhando tempo em evitar filas. Com o aumento dos casos de crimes virtuais a área da segurança da informação precisa ser cada vez mais reforçada, pois indivíduos maliciosos começaram a se aproveitar das oportunidades e invadir os canais virtuais para desvio de dinheiros e pagamentos ilícitos. Os ataques de *crackers* vem crescendo e segundo o site <https://breachlevelindex.com/>, especializado em estatísticas de violações de dados, de 2013 até a metade de 2018, já tinha ocorrido mais de 9,7 bilhões de casos de violação de dados no mundo, mostrando assim um dos números absurdos de ocorrências de crimes cibernéticos.

Essa pesquisa tem por objetivo apresentar um campo da ciência da computação conhecido como “processamento de imagens”, essa técnica pode ser utilizada de várias formas e este trabalho visa o desenvolvimento de um software capaz de utilizar a imagem da webcam para capturar e estudar rostos, desta forma o sistema também será capaz de detectar uma face e realizar o reconhecimento, apresentando quem é a pessoa que aparece na câmera ou se ela é desconhecida.

O estudo deste artigo é importante para a segurança em geral, não se limitando somente a área da tecnologia. A análise de imagens permite uma aplicação muito ampla, sendo utilizada de várias formas diferentes, permitindo assim o crescimento do uso dessa tecnologia para nossa segurança virtual e pública, visando evitar crimes ou qualquer ação maliciosa. Segundo o relatório anual Norton Cyber Security Insights de 2017, o crescimento de empresas adotando o reconhecimento facial para aumentar a segurança, foi de 13% conforme mostra o Gráfico 1 abaixo.

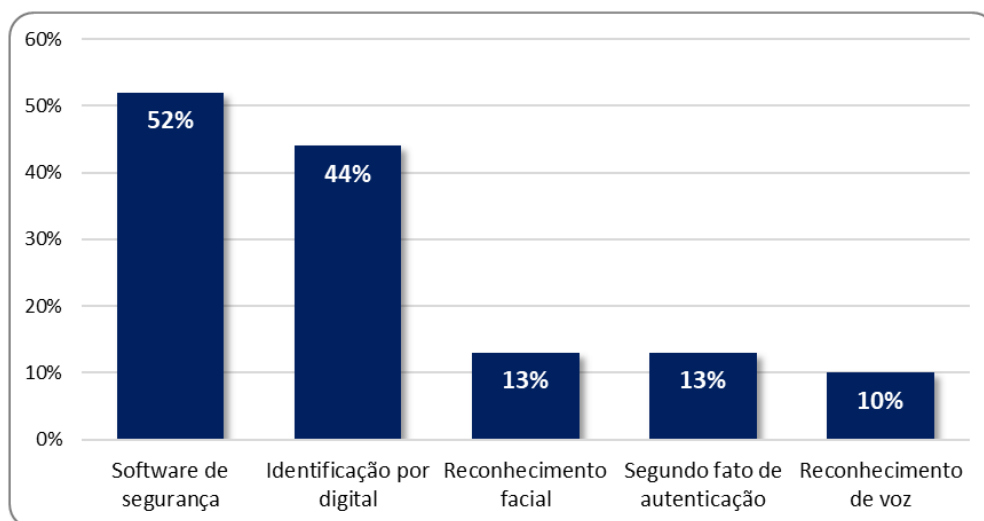


Gráfico 1 – Crescimento de novas tecnologias de segurança em 2016.

Fonte – <https://www.symantec.com/content/dam/symantec/docs/about/2017-ncsir-global-results-en.pdf>

Segundo a pesquisa da empresa Symantec conforme apresentada na Figura 1, somente em 2016, cerca de 1,1 bilhões de dados foram roubados e em um período de 8 anos mais de 7,1 bilhões de dados foram expostos devido a violação de dados, por mais assustadores que sejam estes números, podemos verificar que a quantidade de sucesso nos ataques que ocorreram nos anos de 2014, 2015 e 2016 vem aos poucos reduzindo, pois muitas empresas tem se atentado à necessidade de investimento na área de segurança da informação.




2014	2015	2016	
1.2B 	564M 	1.1B 	Total de identidades expostas
1.523	1.211	1.209	Total de violações

Figura 1 – Quantidade de dados roubados nos anos de 2014, 2015 e 2016.

Fonte – <https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf>

Um gráfico liberado pelo Fórum Brasileiro de Segurança Pública, Anuário 2017, mostra o número de pessoas desaparecidas no Brasil no período de 2007 até 2016, apontando um número superior a 600 mil pessoas registradas como desaparecidas, podendo ser ainda maior devido as subnotificações. Em 2014 foram mais de 94 mil pessoas desaparecidas, conforme mostra o Gráfico 2 abaixo.



Gráfico 2 – Pessoas desaparecidas no BRASIL de 2007 a 2016.

Fonte – <http://www.forumseguranca.org.br/publicacoes/11o-anuario-brasileiro-de-seguranca-publica/>

Conforme mostram os dados apresentados acima, mesmo com tanta evolução tecnológica, existe muito a pesquisar, descobrir e aprimorar, visando a segurança pública e virtual das pessoas, este artigo se propõe a apresentar uma tecnologia ainda em desenvolvimento, mas que pode ser aplicada em vários meios de segurança.

REFERENCIAL TEÓRICO

Para melhor explicar a pesquisa proposta, algumas técnicas e conceitos precisam ser descritos e detalhados para facilitar a leitura do texto e o melhor entendimento do projeto.

Java

Para o melhor desenvolvimento e compreensão deste estudo, foi utilizado um software desenvolvido na linguagem JAVA para executar a função de detectar e reconhecer rostos.

Java é uma linguagem de programação orientada a objetos, que teve seu início do desenvolvimento em 1991 na Sun Microsystems. A linguagem passou a ser utilizada na web, desktop, servidores, jogos, mobile, sendo uma das linguagens mais utilizadas no mundo, podendo ser aplicada em diversos tipos de ambientes e aplicações.

A linguagem é considerada multiplataforma, pois suas aplicações podem atuar em qualquer sistema operacional, isso ocorre pelo fato de trabalhar com uma máquina virtual, não sendo executada diretamente no sistema operacional em que está instalado, essa máquina virtual própria do Java é chamada de Java Virtual Machine (JVM), que tem por função traduzir o *bytecode* para que seja executado em cada sistema.

O *bytecode* é a forma intermediária do código, sendo interpretado pela JVM e se tornando transparente ao desenvolvedor. Não se faz necessário o entendimento da tradução do *bytecode* para o código fonte do sistema operacional, pois a JVM se encarrega deste processo, facilitando o trabalho do programador, pois o sistema desenvolvido para um ambiente poderá ser utilizado em qualquer outro desde que tenha a máquina virtual instalada. A Figura 2 apresenta a estrutura da linguagem Java.

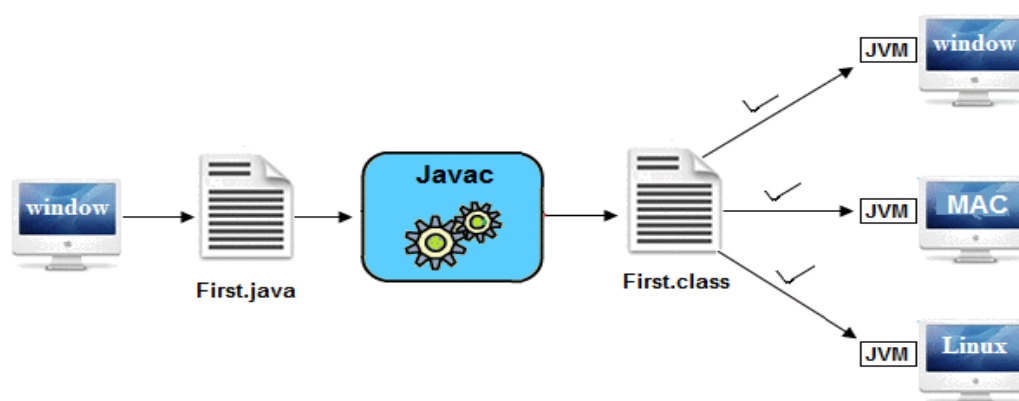


Figura 2 – Funcionamento dos códigos executados na linguagem JAVA.

Fonte – <https://www.sitesbay.com/java/features-of-java>

Outra facilidade em se utilizar o Java são as ferramentas de desenvolvimentos gratuitas fornecidas, como por exemplo a IDE NetBeans, ferramenta utilizada no desenvolvimento do código de reconhecimento facial.

Processamento de imagem

Processamento de imagem pode ser resumido com o seguinte conceito: é a manipulação de uma imagem por computador onde a entrada e saída sejam imagens (GONZALEZ, 2000).

Essa área de estudo se baseia em dois aspectos: a melhoria das informações visuais à interpretação humana e o processamento dos dados para percepção via computadores.

A área de processamento de imagem digitais vem ganhando grande relevância e obteve uma evolução considerável, despertando cada vez mais interesse por parte dos pesquisadores de várias áreas, entre as quais: medicina, criptografia, geoprocessamento e etc. São várias as aplicações: reconhecimento de padrões, interpretação de imagem médica, reconhecimento facial, entre outros.

Segundo Gonzalez (2000, p.2):

“[...] Técnicas de processamento de imagens digitais são atualmente utilizadas para resolver uma grande variedade de problemas. Embora frequentemente não relacionados, esses problemas comumente requerem métodos capazes de melhorar a informação visual para análise e interpretação humana.”

Deteção facial

A visão computacional é uma subárea da inteligência artificial e tem como principal objetivo criar um sistema que seja capaz de simular a visão humana, sendo possível a detecção e reconhecimento de objetos, faces, formas e as mais diversas imagens.

Com relação a visão computacional, graças aos hardwares atuais, as pesquisas no campo de detecção facial apresentaram grandes avanços, despertando cada vez mais o interesse dos pesquisadores. Atualmente esses sistemas de detecção são utilizados em vários campos, sendo possível analisar imagens estáticas ou mesmo diretamente em uma câmera em tempo real. Suas aplicações vão desde reconhecimento de portaria até visão de reconhecimento facial em robôs.

A detecção facial é o primeiro passo para a construção de um sistema de reconhecimento facial, sendo essa etapa muito importante, pois elimina informações desnecessárias que estão junto ao rosto de uma pessoa, facilitando assim o procedimento de reconhecimento.

Para o ser humano, o ato de detectar um objeto ou rostos é muito simples, mas quando tratamos isso no computador, existem diversos desafios que precisam ser superadores, tornando assim uma tarefa complexa, que parece simples quando analisado de forma humana. Dentre os desafios incluem: variação de luminosidade, faces em diferentes tamanhos, faces não frontais e etc.

Reconhecimento Facial

Segundo Russel (2003, p.855):

“A visão nos permite reconhecer pessoas, animais e objetos inanimados de maneira confiável. Em IA ou no estudo de visão computacional é habitual utilizar-se a expressão reconhecimento de objetos como referência a todas essas habilidades.”

Muitas vezes a área de reconhecimento facial é tratada como a de detecção, mas vale ressaltar que são áreas distintas, pois a detecção vai ter por objetivo somente encontrar a face, sendo ela enviada por meio de uma imagem estática ou através de um vídeo e ele irá detectar e informar o rosto. O reconhecimento facial, vai trabalhar além do processo de detecção, ele vai capturar a face detectada e dizer se aquela imagem representa a pessoa $p1$ ou $p2$, realizando o processo de identificar que indivíduo é esse que está na imagem ou na câmera.

Para realizar o reconhecimento de uma face é necessário que o sistema realize algumas etapas conforme mostra a Figura 3. Essa imagem representa o digrama do sistema de reconhecimento facial, onde mostra o processo de detecção e além dele, outros processos como: extração de características, identificação/reconhecimento e a resposta do sistema.

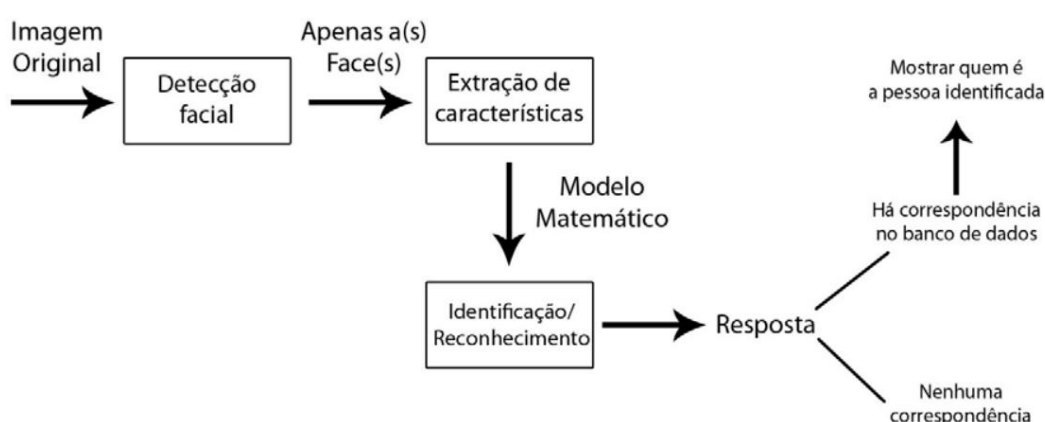


Figura 3 – Diagrama do sistema de reconhecimento facial

Fonte – Site <http://opencv.org/>

Algoritmo LBPH

O Padrão Binário Local (*Local Binary Pattern*, ou LBP) é um operador de textura simples, porém eficiente, tem por objetivo rotular os *pixels* de uma imagem ao limitar os vizinhos de cada *pixel* da imagem, utilizando o resultado como um número binário.

LBP foi descrito em 1994, desde então foi descoberto ser uma poderosa característica para classificação de textura, melhorando ainda mais seu desempenho da detecção, quando somado com o classificador Histograma de Gradientes Orientados (*Histogram of Oriented Gradients* ou HOG), por este motivo é chamado de LBPH, sendo a soma do algoritmo LBP com a técnica estatística de histograma.

O LBP analisa uma matriz de *pixels* definida no desenvolvimento, selecionando um ponto de *pixel* e avaliando o seu valor com os demais da vizinhança, realizando assim um comparativo com os demais *pixels* definidos em um raio. Considerando um *pixel* $P1$, caso o valor do *pixel* vizinho definido no raio R , seja maior ou igual ao valor de $P1$, então o *pixel* vizinho será rotulado com o valor de 1, caso seja menor então recebe o valor de 0, com o resultado da matriz obtida após essa operação é realizada a junção de cada *pixel* da matriz e transformado para o valor decimal que eles representam. A Figura 3 apresenta o processo realizado.

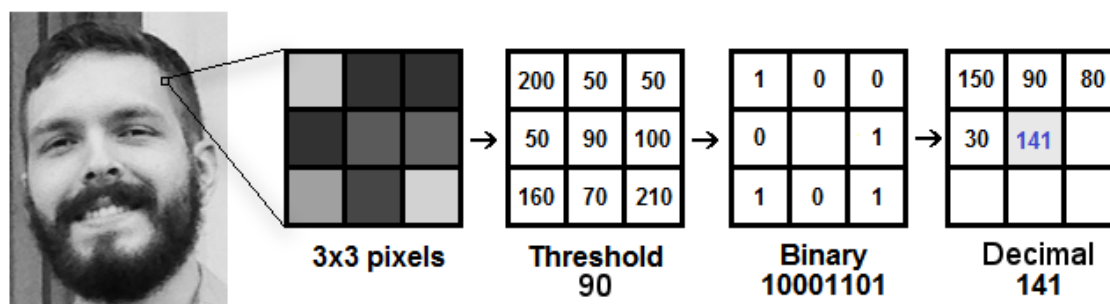


Figura 3 – Matriz capturada pelo algoritmo LBP.

Fonte - <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

Conforme mostrado na Figura 3, é extraída uma matriz de *pixels* no tamanho 3x3. Analisando a imagem, são capturados os valores dos *pixels* e realizado a comparação do *pixel* central (neste caso possui o valor de 90) com os demais, conforme descrito acima, os *pixels* vizinhos do raio que possuem um valor maior ou igual ao do ponto central recebem o valor de 1 e os que são menores, recebem o valor de zero. Após o cálculo da nova matriz (neste exemplo), foi obtido o valor binário de 10001101, que quando calculado para decimal o valor é de 141.

A Figura 4 abaixo, demonstra a utilização do raio de *pixels* e a definição de *pixels* vizinhos, também a possibilidade de alteração do tamanho deste raio, enfatizando que, o procedimento LBP de expandir para usar um número diferente de raio e vizinhos, é chamado de Circular LBP.

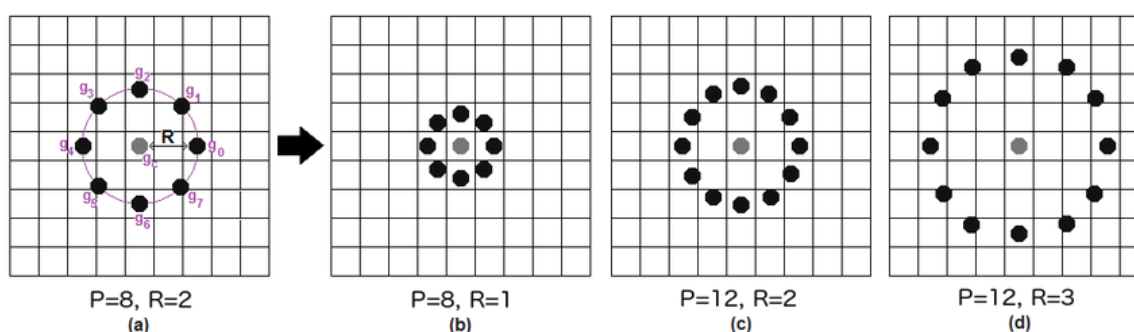


Figura 4 – Processo de expandir o tamanho do raio.

Fonte - <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

Diferente de alguns outros algoritmos, o LBP possui grande resistência com a iluminação, já que caso uma imagem seja iluminada a alteração dos valores irão mudar de forma igual, por exemplo: baseando-se na Figura 3, caso a imagem sofra uma alteração de luminosidade, o *pixel* da posição 3x2, que atualmente possui o valor de 70, teria seu novo valor alterado para 100, o que não iria alterar seu valor de 0 para 1, já que o *pixel* central também teria seu valor aumentado, portanto, aumentando o valor de *Threshold*, não ocorrendo alteração no valor final da operação.

Após a transformação do valor de cada *pixel* utilizando o LBP, o sistema irá construir um histograma que conta quantos elementos existem de cada valor, como temos uma imagem em escala de cinza, cada histograma (de cada grade) conterá apenas 256 posições (0 ~ 255) que representam as ocorrências de cada intensidade de pixel. Conforme o Gráfico 3 abaixo, ele apresenta um histograma utilizado no LBPH.

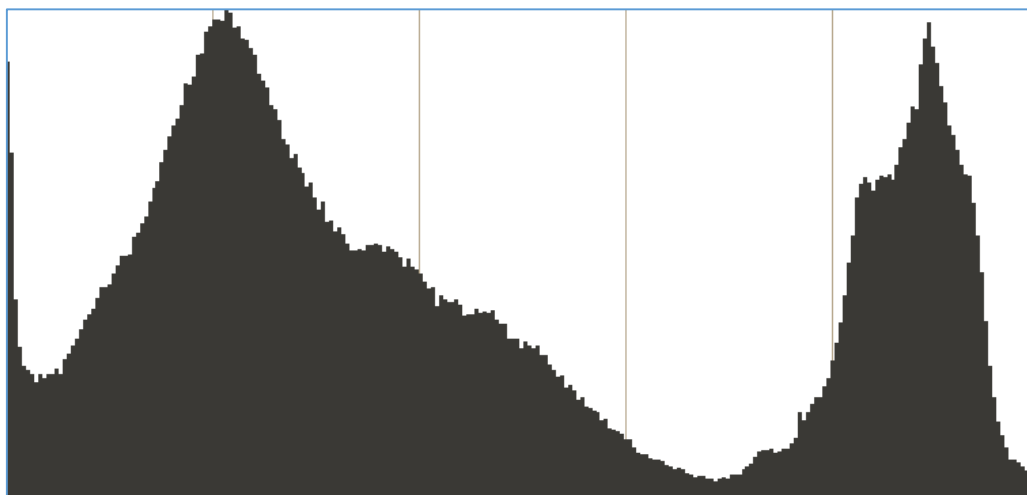


Gráfico 3 – Modelo de histograma.

Fonte – <https://blog.emania.com.br/saiba-usar-os-histogramas-a-seu-favor/>

METODOLOGIA

Esta é uma pesquisa aplicada, onde os conceitos estudados são aplicados visando a utilização no mundo real e de caráter qualitativo através das avaliações do autor quanto a aplicação das técnicas e algoritmos.

A abordagem qualitativa pesquisa detalhadamente os fenômenos do ambiente estudado, o pesquisador vive e conhece a realidade deste grupo ou ambiente. Na pesquisa qualitativa, o pesquisador participa, compreende e interpreta (MICHEL, 2009).

Também foi utilizado como método a pesquisa bibliográfica, composta de estudos através dos livros, artigos, teses, dissertações, anais de congressos e etc. Visando criar um conteúdo completo e seguro sobre o assunto estudado.

Para elaboração deste trabalho, foi desenvolvido um sistema protótipo para melhor compreender o processo e as funcionalidades de um software de reconhecimento facial, este software foi criado utilizando a linguagem de programação Java, devido maior proximidade do autor com esta linguagem.

O sistema de reconhecimento facial foi desenvolvido utilizando a biblioteca JavaCV, contendo o OpenCV e outras bibliotecas. As classes foram divididas em três partes, explicado e detalhado em fases: processo para captura dos dados; treinamento; detecção e reconhecimento facial.

Captura

Essa é a primeira parte de execução do software, a classe “captura” é responsável por realizar a detecção da câmera, seja ela webcam de um computador ou um periférico de entrada inserido *off board*, a imagem da câmera é aberta em um *Frame* do Java, retornando uma tela com a captura da imagem da câmera utilizada.

Após realizar a entrada da imagem, é necessário a utilização de um método para que seja feito a conversão do formato da imagem para o *Mat*, que é um formato de matriz, para esta função o JavaCV possui um método pronto codificado na Figura 5.


```
OpenCVFrameConverter.ToMat converter = new OpenCVFrameConverter.ToMat()
```

Figura 5 – Declaração da classe OpenCVFrameConverter.

Fonte – Autoria própria.

Também realizamos o processo de detecção de face, utilizando o arquivo *haarcascade-frontalface-alt.xml*, que já é um arquivo classificador treinado específico para detecção de faces, desta forma, o sistema é capaz de detectar o rosto de uma pessoa, sendo passada por uma foto ou direto através da câmera.

A partir da imagem colorida do rosto detectado e transformada em uma imagem preto e branco é recortada somente a parte da face, para que seja salva em uma pasta separada, posteriormente essas imagens serão utilizadas para realizar o processo de treinamento, como descrito da subseção “Treinamento”.

Algumas variáveis são utilizadas para definição de configuração durante o processo de captura, para definir limite de imagens, teclas utilizadas para maior controle das imagens capturadas e definição de valor de ID das pessoas do treinamento.

Treinamento

Essa classe do sistema é responsável por aplicar o algoritmo LBPH em cada uma das imagens dos rostos que foram capturadas no processo anterior, para isso é atribuído um rótulo para cada uma das imagens que estão salvas para treinamento, e ao final são utilizadas funções do JavaCV para que possam ser declaradas as funções do algoritmo LBPH e a função de treinamento, conforme codificado na Figura 6.

```
FaceRecognizer lbph = LBPHFaceRecognizer.create(1, 15, 15, 15, 1);
lbph.train(fotos, rotulos);
lbph.save("src\\recursos\\classificadorLBPH.yml");
```

Figura 6 – Algoritmo de geração do arquivo de treinamento LBPH.

Fonte – Autoria própria.

A Figura 6 mostra a instância do *FaceRecognizer* sendo declarada, passando alguns valores de parâmetros, no exemplo são passados os parâmetros: 1, 15, 15, 15, 1, estes valores são utilizados para configuração de: *radius*, *neighbors* (número dos vizinhos do pixel limiar), *gridX*, *gridY* e *Threshold*.

Após realizar o treinamento, é executado o comando “save” para guardar um arquivo com as informações do classificador.

Reconhecimento

Este é o último passo do processo de detecção e reconhecimento facial através de uma webcam, é muito parecido com o procedimento de captura, pois o reconhecimento será realizado através da captura da nova imagem de entrada, executado novamente o algoritmo LBPH para realizar o procedimento já discutido em sessões anteriores que é a comparação de histogramas, gerando um valor de confiança.

Nessa classe foi inserido um valor de *Threshold* para delimitar o valor máximo de confiança para que seja reconhecido como a pessoa do treinamento e impresso na tela o nome da pessoa que foi definido através de um vetor de *strings*. Caso o sistema não encontre a pessoa do treinamento, ele retorna um valor de -1, valor este que é padrão da função *predict*, e neste caso, foi solicitado que o software apresentasse o nome “Desconhecido”, caso contrário ele irá retornar o valor de rótulo (definido anteriormente) e baseado neste valor é impresso o valor do vetor de *string*, se for a pessoa de rótulo 1 ele retorna o nome da pessoa que está no vetor na posição 1, e assim por diante.

LEVANTAMENTO DOS DADOS E RESULTADOS

Foi elaborado um protótipo do sistema utilizando a linguagem Java, o sistema executado realiza a detecção da face e através de um treinamento é capaz de fazer o reconhecimento, informando quem é a pessoa, podendo essa imagem ser capturada através de uma câmera em tempo real ou uma fotografia armazenada anteriormente.

O sistema utiliza o pacote JavaCV que é um *wrapper* do OpenCV, contendo toda essa biblioteca do C/C++ utilizada para a área de processamento de imagem, o JavaCV apenas está “puxando” toda a biblioteca OpenCV em uma interface diferente, além do OpenCV são utilizadas várias outras bibliotecas, incluindo FFmpeg, OpenKinect e outras.

Para realizar o processo de reconhecimento facial foi necessário acompanhar cada uma das etapas conforme descrito na Figura 2. Para melhor compreensão de cada processo executado, os processos serão descritos em etapas.

Etapa 1 – Imagem de entrada

O sistema precisa receber uma imagem original, normalmente colorida, essa imagem pode ser enviada em formato de arquivo ou uma captura diretamente em vídeo, que é o caso deste trabalho, as imagens foram capturadas diretamente da webcam tanto para fazer o treinamento quanto o reconhecimento.

A imagem abaixo, é uma imagem real da execução do software, essa imagem foi enviada e lida pelo sistema, em seguida foi realizado o segundo passo que é a detecção de uma face, conforme mostra a Figura 7.

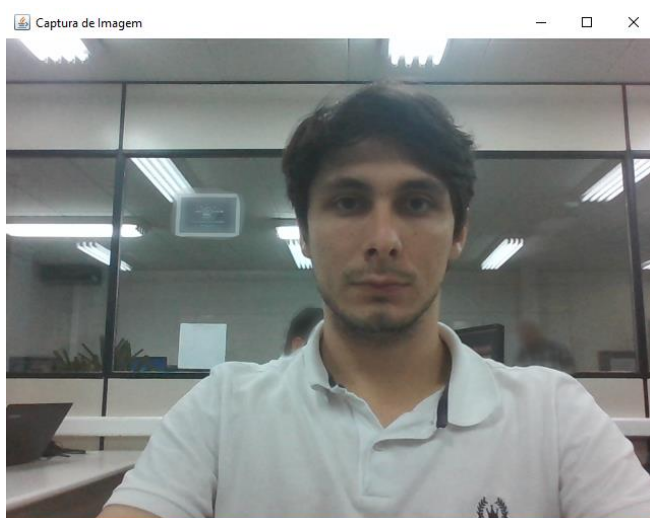


Figura 7 – Captura da imagem original pelo sistema.
Fonte – Autoria própria.

Etapa 2 – Detecção e captura

Após o recebimento de uma imagem, o sistema realizou o processamento através da biblioteca OpenCV para ler e detectar a face. Para comprovação, é inserido uma codificação com o objetivo de desenhar um retângulo em volta do rosto detectado. A Figura 8 mostra exatamente o desenho feito pelo sistema.

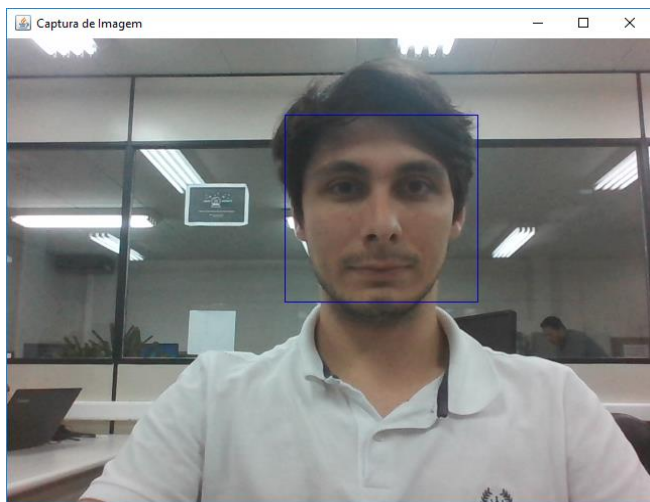


Figura 8 – Utilização do OpenCV para detectar e demarcar uma face.
Fonte – Autoria própria.

Para que o sistema realizasse o treinamento com o processo de extração das características, a imagem colorida foi transformada em uma imagem preto e branco, também foi recortado somente o conteúdo da imagem que está dentro do retângulo, dessa forma, após remover toda informação desnecessária, mantendo apenas a imagem do rosto.

A Figura 9 abaixo, apresenta as imagens que foram salvas em uma pasta para posteriormente serem utilizadas no treinamento.



Figura 9 – Faces utilizadas para treinamento.
Fonte – Autoria própria

A Figura 9 apresenta um exemplo de alguns modelos de imagens que foram capturadas e recortadas para treinamento, como pode ser observado é recomendado uma boa quantidade de imagens, a recomendação é de pelo menos 25 imagens, com variação

de posições, expressões e luminosidade, ressaltando que dependendo do algoritmo de reconhecimento a variação de luz pode ser o fator mais crucial para o reconhecimento. Como neste artigo foi utilizado o algoritmo LBPH, sendo este resistente a luz conforme já descrito em sessões anteriores, não será abordado em detalhes a relação da luz no processo de reconhecimento facial.

A Figura 10 abaixo, mostra o código utilizado para detectar o rosto, realizar a captura da imagem do retângulo e salvar em uma pasta do sistema.

```
for (long i = 0; i < faces.size(); i++) {
    Rect dadosFace = faces.get(i);
    rectangle(imagemColorida, dadosFace, new Scalar(203, 0, 0, 0));
    Mat faceCapturada = new Mat(imagemCinza, dadosFace);
    resize(faceCapturada, faceCapturada, new Size(160, 160));

    if (tecla == null) {
        tecla = frame.waitKey(5);
    }
    if (tecla != null) {
        if (tecla.getKeyChar() == 'q') {
            if (amostra <= numeroAmostras) {
                imwrite("src\\fotos\\pessoa." + idPessoa + "." + amostra + ".jpg", faceCapturada);
                System.out.println("Foto " + amostra + " capturada\n");
                amostra++;
            }
        }

        tecla = null;
    }
}
```

Figura 10 – Código que realiza o recorte e captura das imagens.

Fonte – Autoria própria.

Etapas 3 – Treinamento

Nessa etapa, o sistema já realizou a detecção do rosto e a captura das imagens, após essas etapas, será executado o treinamento das imagens, sendo retirada as características através dos dados que são passados no código. O treinamento será executado através do algoritmo LBPH já descrito anteriormente no referencial teórico, onde será realizado todo o processo de seleção da matriz de pixels, e o cálculo com os vizinhos.

Com o objetivo de reforçar o conhecimento da execução do LBPH, segue abaixo uma lista dos passos realizados supondo que tenhamos uma imagem facial em escala de cinza como mostrado na Figura 9.

1. É obtido parte desta imagem como uma janela de 3x3 pixels ou maior, podendo ser controlado pelo programador.
2. Então, precisamos tomar o valor central da matriz para ser usado como limiar.
3. Esse valor será usado para definir os novos valores dos 8 vizinhos.
4. Para cada vizinho do valor central (limiar), estabelecemos um novo valor binário recebendo 1 se o valor for igual ou superior ao limiar e 0 para os valores menores.
5. Agora, a matriz conterá apenas valores binários (ignorando o valor central).
6. Precisamos concatenar cada valor binário de cada posição da matriz linha por linha para um novo valor binário (por exemplo, 10001101).
7. Então, será convertido esse valor binário em um decimal e colocado no lugar do valor do pixel central.
8. No final deste procedimento (chamado LBP), temos uma nova imagem que representa melhor as características da imagem original.

Por fim, serão retirados os valores e criado um histograma, conforme pode ser visto na Figura 11:

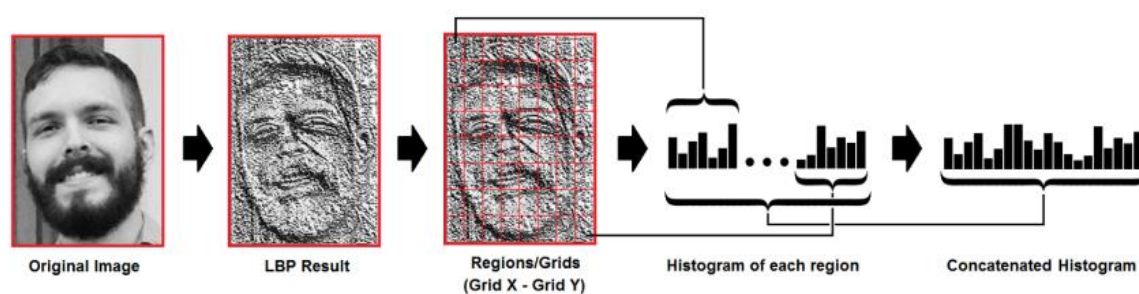


Figura 11 – Processo com imagem realizado pelo LBPH.

Fonte – <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

Após a execução do algoritmo de treinamento, foi gerado um arquivo com a extensão “.yml”, este arquivo contém as informações para o próximo passo de reconhecimento facial.

Etapa 4 – Reconhecimento

Agora que o algoritmo já está treinado, os histogramas são utilizados para representar cada imagem do conjunto de dados do treinamento. Assim, para realizar a comparação entre imagens é executada a etapa novamente e comparado os dois histogramas, retornando então o histograma mais próximo.

É possível usar algumas abordagens para comparar dois histogramas, que no caso é chamado de distância ou a confiança do sistema, por exemplo: distância euclidiana, qui-quadrado, valor absoluto e etc. Neste exemplo, é utilizado a distância euclidiana, baseada na fórmula apresentada na Figura 12:

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

Figura 12 – Fórmula da distância euclidiana para calcular a confiança dos histogramas.

fonte – <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>

O sistema após fazer o reconhecimento da imagem de entrada com as imagens que foram utilizadas no treinamento, retorna um valor de confiança, porém vale ressaltar que para este valor, quanto menor melhor, isso porque o valor de confiança é o valor referente a distância de igualdade entre os dois histogramas, quanto mais próximo, significa que as imagens são mais parecidas. Por exemplo, se for realizado o treinamento com a imagem X e essa mesma imagem for utilizada para comparação, então o resultado de confiança será 0, ou seja, diferença zero, o que significa que as imagens são iguais.

Durante o desenvolvimento podem ser realizadas várias configurações, como o tamanho mínimo para detecção, valores de *Threshold* e outros campos que podem auxiliar no processo de leitura. A Figura 13 é o resultado do sistema no reconhecimento facial, onde apresenta a leitura de rostos em diferentes tamanhos, mostrando o retângulo de detecção facial e o valor da confiança do retorno.

Para mostrar de forma visual a leitura e apresentação dos valores de distância, foi permitido que o software reconheça a pessoa “Alexandre” com qualquer valor de confiança, meramente para ilustrar o funcionamento.

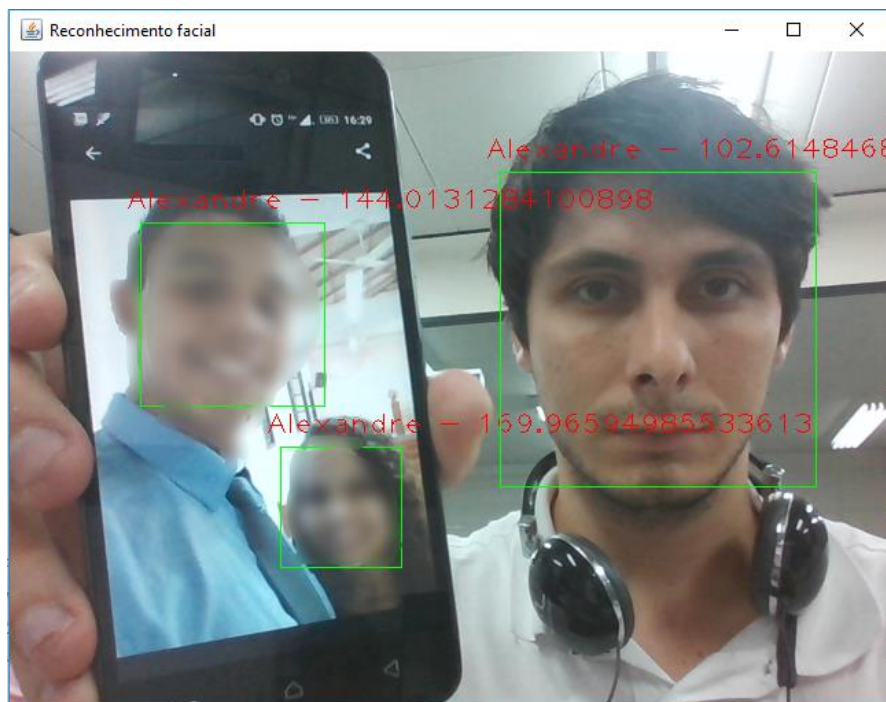


Figura 13 – Reconhecimento facial com *threshold* alto.

Fonte – Autoria própria.

Como pode ser observado na Figura 13, foi realizada a detecção de cada face, o sistema também comparou cada rosto capturado e gerou um valor de confiança, sendo essa a distância entre o rosto atual e a imagem de treinamento. A pessoa 1 recebeu o valor de 144, a pessoa 2, 159 e a pessoa 3, que neste exemplo é a mesma pessoa do treinamento, recebeu o valor de 102, portanto, o sistema definiu que a pessoa mais parecida é a terceira.

O valor de *Threshold* que é o que define o limite para ser considerado reconhecido ou não, é um parâmetro que precisa ser analisado para cada ambiente, tendo conhecimento dos valores que são distribuídos é possível configurá-lo para realizar o reconhecimento facial correto. A imagem abaixo é a representação real do funcionamento do sistema, dessa vez com o parâmetro definido de forma correta e assim realizando o reconhecimento dos rostos com no máximo 105 de *Threshold*.

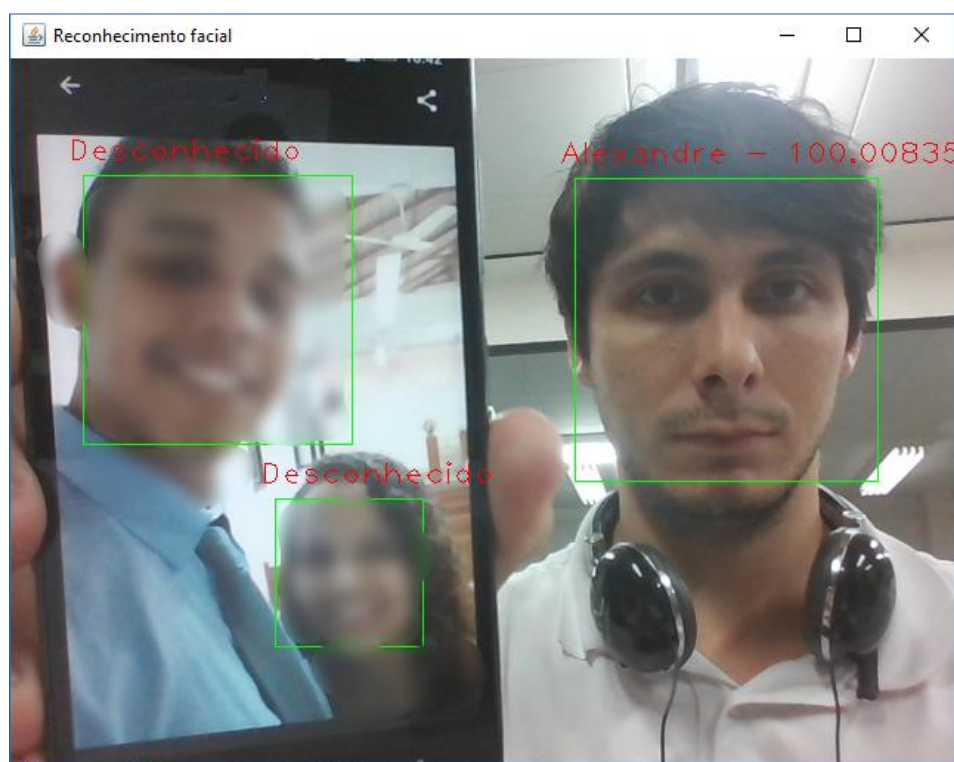


Figura 14 – Reconhecimento facial com valor de *threshold* regulado.

Fonte – Autoria própria.

Agora que foi definido um valor máximo de distância para ser considerado a pessoa do treinamento, o sistema consegue reconhecer a pessoa própria do treinamento e outras pessoas são consideradas desconhecidas, pois o valor de distância entre os histogramas é muito grande conforme apresenta a Figura 14.

CONCLUSÃO

O algoritmo LBPH teve um bom desempenho no reconhecimento facial, mostrando ser robusto com a transformação monotônica em escala de cinza. Foram testados outros dois algoritmos: o Engefases e o Fisherfaces, porém, nos testes realizados o LBPH obteve o melhor resultado com relação ao reconhecimento, apresentando uma distância pequena no cálculo dos histogramas, os testes realizados não foram divulgados por não ser o foco deste trabalho.

Com o desenvolvimento deste artigo foi possível analisar a execução dos algoritmos e entender quais os pontos importantes de atenção, como é o caso das imagens para treinamento, a luminosidade e valores de *threshold*. Conhecer o código e entender cada parte de funcionamento são necessários para realizar melhorias, buscando alcançar maior precisão na detecção e no reconhecimento facial.

Durante o tempo de desenvolvimento e testes com outros algoritmos, se tornou visível os pontos fortes e fracos que cada um apresenta, portanto conhecer outros códigos e aplicá-los é necessário para que se possa compreender onde e quando utilizar cada um.

CONSIDERAÇÕES FINAIS

Durante os testes realizados com o sistema de reconhecimento utilizando o algoritmo LBPH, ele apresentou uma variação no valor de confiança arredondado de 92 até 124 em variações de luz, testados no período da manhã, tarde e noite. Essa variação pode ocorrer devido a vários fatores. Vale enfatizar a importância que tem as imagens de treinamento, a recomendação é que sejam capturadas de 25 a 50 imagens para o treinamento, pois conforme explicado, são dessas imagens que será retirado o histograma para comparação com as outras.

As imagens de treinamento precisam ter diferentes posições, não sendo estática, pois a análise ocorre em tempo real através de uma câmera, ou seja, a imagem analisada nunca será em uma mesma posição fixa, ocorrendo variações, desta forma é necessário que as imagens treinadas possam ter leitura de cada lado do rosto e com várias expressões. Também é recomendado a variação de luz para que o software não sofra com as diferenças de dia e noite, aumentando ainda mais a capacidade de reconhecimento.

Outra consideração necessária é o tratamento do valor de *threshold*, este valor é o que determina qual a distância máxima entre dois histogramas para que a imagem capturada seja reconhecida como a pessoa das fotos treinadas. Caso o valor do *threshold* seja muito baixo, pode ocorrer problemas no reconhecimento da pessoa treinada quando ocorre variações de posição e/ou luz, o que são fatores que podem distanciar os histogramas, porém se o valor definido no *threshold* for muito alto, poderá ocorrer o mesmo problema da figura 12, pessoas que não são rostos treinados são reconhecidos como a mesma pessoa.

A fim de evitar tais problemas é necessário que o desenvolvedor tenha conhecimento dos valores de confiança alcançados, desta forma poderá definir o valor necessário para que sejam feitos os reconhecimentos de forma correta.

REFERÊNCIAS

- AHONEN, Timo, ABDENOUR Hadid, and MATTI Pietikainen. "Face description with local binary patterns: Application to face recognition." *IEEE transactions on pattern analysis and machine intelligence* 28.12 (2006): 2037–2041.
- ALMEIDA, Osvaldo Cesar Pinheiro de. Técnicas de processamento de imagens para localização e reconhecimento de faces. São Carlos. SP, 2006. Dissertação (mestrado em Ciência da Computação). Universidade de São Paulo, USP.
- AMARAL, Vagner, Gilson A. Giralddi, and CARLOS E. Thomaz. "LBP estatístico aplicado ao reconhecimento de expressões faciais." (2013).
- AMAMRAL, Vagner do and CARLOS Eduardo Thomaz. Extração e Comparação de Características Locais e Globais para o Reconhecimento Automático de Imagens de Faces. Diss. Dissertação de Mestrado, Centro Universitário da FEI, SP, Brasil, 2011.
- ARAUJO, Gabriel Matos. Algoritmo para reconhecimento de características faciais baseado em filtros de correlação. Rio de janeiro, RJ, 2010. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Rio de Janeiro, UFRJ.
- BRAGA, Luiz Filipe Zenicola. Sistemas de Reconhecimento Facial. Diss. UNIVERSIDADE DE SÃO PAULO, 2013.
- BROOKSHEAR, J. Glenn. Ciência da Computação: Uma visão abrangente. 7. Ed. São Paulo, Editora Artmed, 2003.
- CONCI, Aura, Eduardo Azevedo, and FABIANA R. Leta. Computação gráfica. Elsevier, 2008.
- GAZUHIRO Okabe, Rogerio, and SILVIO Antonio Carro. "RECONHECIMENTO FACIAL EM IMAGENS CAPTURADAS POR CÂMERAS DIGITAIS DE REDE." *Colloquium Exactarum*. Vol. 7. No. 1. 2015.
- GONZALEZ, R. C., and R. E. WOODS. "Processamento de Imagens Digitais, tradução do original Digital Image Processing." Edgard Blucher, São Paulo (2000).
- MICHEL, Maria Helena. Metodologia e Pesquisa Científica em Ciências Sociais: Um guia prático para acompanhamento da disciplina e elaboração de trabalhos monográficos. 2ª ed. São Paulo: Ed. Atlas, 2009.
- NORVIG, Peter, and STUART Russell. Inteligência Artificial: Tradução da 3a Edição. Vol. 1. Elsevier Brasil, 2014.
- SILVA, Alex Lima, and MARCOS Evandro Cintra. "Reconhecimento de padrões faciais: Um estudo." Encontro Nacional de Inteligência Artificial e Computacional, 2015, Proceedings ENIAC. 2015.
- SILVA, Antônio Machado e. Curso Processamento digital de imagens de satélite. Centro de Eventos da PUCRS - de 07 a 12 de outubro de 2001. Porto Alegre - RS. Disponível em www.cartografia.org.br. Acesso em: 19 fev. 2007.
- SPRING: Integrating remote sensing and GIS by object-oriented data modelling. Camara G, Souza RCM, Freitas UM, Garrido J. *Computers & Graphics*, 20: (3) 395-403, May-Jun 1996.