Detecção e Reconhecimento Facial em vídeo *Real-Time*

Bruno Carvalho, Byron Kamal, Gustavo Monteiro, João Vitor Baptista, Pedro Helias Carlos

Universidade de Brasília - UnB/FGA

Resumo. Este artigo visa concretizar o conteúdo teórico e prático aprendido na disciplina Tópicos Especiais em Engenharia de Software: Visão Computacional, trazendo o contexto histórico, aplicação, metodologia e requisitos necessários para implementar a detecção e reconhecimento facial, técnica de processamento de imagens e visão computacional que tem várias aplicações (principalmente na área da ciência das informações) e com o avanço da tecnologia, o treinamento dos computadores pretende chegar ao nível de complexidade e precisão da visão humana.

1. INTRODUÇÃO

O desenvolvimento da tecnologia que conhecemos atualmente como processamento digital de imagens, se deu no início da década de 50, em que teve como precursores empresas e entidades acadêmicas, como Jet Propulsion Laboratory, MIT, Bell Labs, University of Maryland, entre outros, que buscavam lidar com as imagens dos satélites, além de uso para imagens médicas, detecção de caracteres alfanuméricos.

Durante as décadas de 50 e 60 houve uma expansão e melhoria do uso de algoritmos de reconhecimento de padrões de imagens, utilizando técnicas como aplicação de um operador laplaciano na imagem do objeto em nível-de-cinza, para gerar um gradiente da imagem (detector de borda em regiões com mudança significativa de brilho), e assim que os padrões (em formato retilíneo) são determinados, era possível determinar a classe que o objeto faz parte, com os programas aplicando um método de detecção-pós-previsão (basicamente uma tentativa e erro) [1].

O reconhecimento de padrões que ocorriam em câmaras de bolhas (mecanismo para detectar partículas eletricamente carregadas), caracteres alfanuméricos, até mesmo em uma colônia de bactérias tem uma característica em comum que é uma das primeiras barreiras que a tecnologia do processamento digital teve que resolver, elas atendem padrões com formatos

retilíneos, ou seja, qualquer formato que fugisse dessa restrição não era devidamente reconhecido e classificado[2].

No caso do reconhecimento facial, essa metodologia simples encontrava muitas restrições, as fotografias tinham que ser tiradas do rosto completo da pessoa, sem óculos, ou barba, e com restrição de angulação da pose da foto, e do fundo da fotografia, e assim determinada a posição dos olhos, boca, nariz, bochecha com os programas aplicando o método de detecção-pós-previsão, caso houvesse ruídos, uso de parâmetros que não fossem as partes desejadas, algum contorno do rosto que não fosse tabelado, geraria como resultado uma detecção inadequada do padrão do rosto .

A detecção e reconhecimento de faces por meio de processamento de imagens vem desde então avançando, com a melhora da tecnologia dos computadores no processamento de grandes conjuntos de dados, uma nova ciência surgiu, a visão computacional, tecnologia que visa treinar o computador para extrair, processar e analisar imagens ou qualquer dado multidimensional, de modo a criar um alto nível de entendimento sobre os objetos para automatizar o computador a realizar uma ação equiparada a complexidade do sistema visual humano por meio de inteligência artificial.

A restrição tecnológica de capacidade de processamento dos computadores ainda não permitiu que tal nível tenha sido alcançado, porém mais que apenas imagens, o desenvolvimento desse campo já permite extração de informações de vídeos, *real-time*, e é usado em vários âmbitos da sociedade, dos meios acadêmicos e científicos de maior complexidade para produção de tecnologia e conhecimento de ponta, até usos para entretenimento de pessoas comuns, o que faz o seu estudo importante, visto que, ainda há muitos campos não explorados e capacidades não atingidas.

2. TRABALHOS CORRELATOS

Atualmente existe um grande número de trabalhos realizados na área de reconhecimento facial que utiliza o algoritmo de histogramas de padrões binários locais ou LBPH. Três desses trabalhos serão apresentados a seguir.

O primeiro deles é o *OpenFace*, uma biblioteca com o propósito geral de reconhecimento facial com aplicações em celulares, criada por alunos da Universidade Carnegie Mellon em Pittsburgh, nos Estados Unidos. Eles partem do princípio de que a identidade de uma pessoa é uma grande parte do contexto em humanos e modula o que as

pessoas dizem e como elas agem [3]. Da mesma forma, o reconhecimento de pessoas é uma operação primitiva na computação móvel que adiciona contexto a aplicativos como assistência cognitiva, eventos sociais, anotação de alto-falante em reuniões e identificação de pessoas de interesse por meio dos dispositivos móveis. Os experimentos dos alunos mostraram que o *OpenFace* oferece maior precisão do que os projetos de código aberto anteriores e é bem adequado para cenários de dispositivos móveis.

Outro importante trabalho correlato é o de Ranganatha S et al. [4] que descreve um modelo para detecção e rastreamento de rostos humanos em diferentes sequências de vídeo de fundo usando a plataforma OpenCV. Ambas as amostras de imagens positivas e negativas são treinadas e salvas como arquivo xml. Com a ajuda de amostras treinadas, o algoritmo LBPH esclarece se o quadro de vídeo contém faces ou não.

Há também o trabalho de Shireesha C et al. [5] que propõe um sistema de gerenciamento de atendimento automatizado. Este sistema, que é baseado em algoritmos de detecção e reconhecimento de faces, detecta automaticamente o aluno quando ele entra na sala de aula e marca o atendimento reconhecendo-o. Diferentes cenários em tempo real são considerados para avaliar o desempenho de vários sistemas de reconhecimento facial. Quando comparado com a marcação de atendimento tradicional, este sistema economiza tempo e também ajuda a monitorar os alunos.

3. MODELO PROPOSTO

Cores formam o espectro contínuo, são percepções visuais provocadas pela ação de um feixe de fótons sobre células especializadas na retina, a visão humana é restrita ao espectro visível, faixa de comprimento de ondas que vai de 400*nm* (Violeta) até 700*nm* (Vermelho), uma imagem colorida pode ser representada por um modelo de cores: *RGB* (Red- Green – Blue) , *HSV*(Hue – Saturation - Value), *CMYK*(Cian – Magenta – Yellow – black Key), entre outros, que variam de acordo com o tipo de processamento que pretende-se aplicar, uma vez que cores podem ser descritores da imagem, seja pela cor (*RGB*), intensidade e brilho (*HSV*), meio-tom (*CMYK*).

O modelo mais comumente usado é o *RGB*, uma imagem colorida neste modelo é representada por 3 matrizes, cada matriz individualmente do R, G e B é dita como

nível-de-cinza, e possuem valores de pixel de 0 a 255 cada, onde 0 representa a cor preta, e 255 a cor branca, e uma imagem também pode ser do tipo binária, na qual cada pixel só pode ter 2 valores, geralmente essa imagem é resultado do pós-processamento de uma imagem colorida.

É possível processar a imagem colorida, porém requer maior poder de processamento do dispositivo, uma vez que para poder extrair alguma informação na imagem colorida é necessário primeiro extrair a informação em cada um dos níveis de cinza e depois sobrepô-los para gerar o resultado final, para isso, um método para facilitar o processamento da imagem é transformar a imagem colorida para nível-de-cinza, assim tendo que executar as operações desejadas apenas em relação a uma matriz.

Uma vez que a imagem já esteja em nível de cinza, existem muitos métodos para operar sobre essa matriz, manualmente, uma vez que pode-se determinar especificamente técnicas para cada caso, porém também já existem muitas funções prontas que realizam satisfatoriamente a aplicação das operações sobre imagens.

Neste artigo a implementação das técnicas serão realizadas por meio da linguagem de programação Python, e especificamente algumas de suas bibliotecas como Numpy (Numeric Python), Matplotlib, e a biblioteca Opencv, que é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de visão computacional (onde existem muitas aplicações já desenvolvidas para o processamento de imagens).

Para criação de um sistema de detecção e reconhecimento facial, é necessário usar muitas imagens para treinamento da máquina para extração dos padrões necessários, para isso, será usado neste trabalho o SQLite que é uma biblioteca que implementa um banco de dados.

Detecção facial é o processo realizado pela integração dos pixels da imagem e implementação das funções de classificadores em cascata que gerarão os padrões que serão usados como base para o procedimento de Reconhecimento facial, uma vez que por meio da entrada das imagens, ou banco de dados das imagens que deseja-se executar esse reconhecimento, precisam desse padrão já determinado para fazer a análise.

O processo resumidamente para executar a detecção e o reconhecimento facial dá-se por: entrada da imagem/banco de imagens, leitura e conversão da imagem para nível de cinza, criação do detector de faces, aplicação dos descritores para detectar o padrão desejado

na imagem, implementação de um quadro para identificar onde está o resultado da detecção da característica desejada (a cor pode ser determinar como a pessoa quiser, uma vez que na implementação a cor do quadro também deve ser especificada por meio do código RGB), cria-se uma lista com esses resultados, gerando um array com essas informações (ou salva em uma banco de dados) e por fim o uso da função de reconhecimento detectMultiscale que vai identificar na base de dados quais imagens fazem parte do conjunto de padrões extraídos e classificados no processo de de extração [6].

3.1 - Algoritmo de detecção facial

Em 2001, dois pesquisadores Paul Viola e Michael Jones propuseram um algoritmo com o objetivo de resolver os problemas de detecção de faces, contudo o algoritmo serve para a detecção de objetos em tempo real em geral.

Identificar rostos é uma tarefa simples para os seres humanos, porém são necessárias instruções específicas para o computador executar determinada tarefa. O objetivo central do algoritmo é simplesmente distinguir rostos de não-rostos, tal metodologia é robusta pois possui uma grande taxa de acerto e taxas de falso positivos baixa, pode ser aplicada em tempo real pois em comparação com outras metodologias possui um custo computacional vantajoso. O algoritmo consiste em selecionar características específicas do rosto, fazer uma operação para minimizar o custo computacional, em seguida são feitas classificações das janelas e por fim é feita a cascata de classificadores. [7]

Em geral, os rostos humanos compartilham algumas características similares, tais características podem ser identificadas através de *features* retangulares que são unidades básicas de do algoritmo Viola-Jones representada na figura 1, consiste em retângulos com faixas pretas e brancas em diversos sentidos e tamanhos.

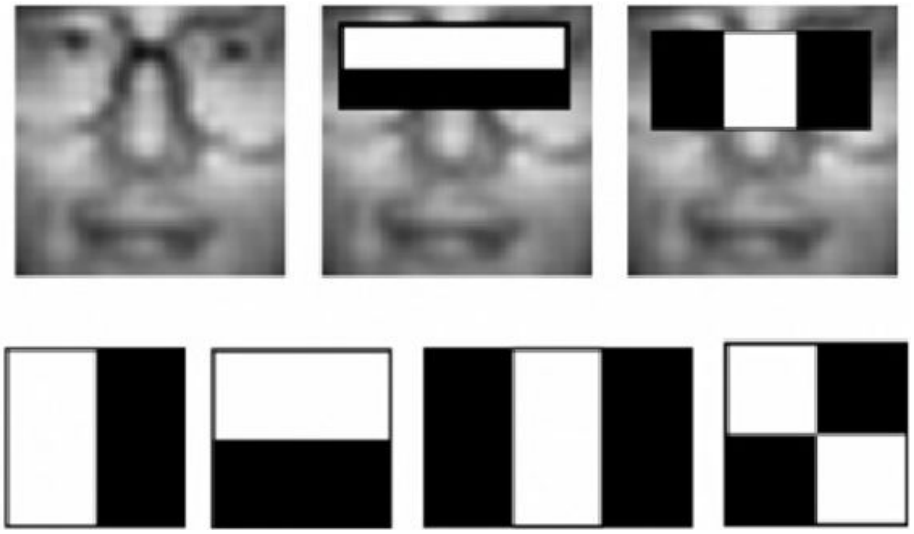


Figura 1 – Unidades básicas denominadas *Features* retangulares.

O valor de cada *feature* retangular sobre uma imagem deve ser calculada usando a equação 1.

Equação 1 - Equação para computar a diferença entre os pixels da faixa preta e faixa branca

$$f(w) = \frac{1}{w}\Sigma^{w}P_{preto} - \frac{1}{w}\Sigma^{w}P_{branco}$$

$$f(w): \text{Valor do feature na janela } w:$$

$$\frac{1}{w}\Sigma^{w}P_{preto}: \text{Média dos pixels na região preta}$$

$$\frac{1}{w}\Sigma^{w}P_{branco}: \text{Média dos pixels na região branca}$$

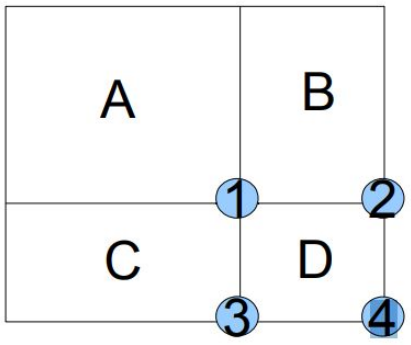
Quanto mais f(w) se aproximar de 1 mais provável é a chance de do *feature* retangulares ser um Haar-Feature. Porém existe o problema de calcular a média de um somatório muito grande de pixels e isso leva a um custo computacional de O(n²) que não é viável para aplicações em tempo real. Para acelerar o cálculo do valor de um feature, é utilizada a representação da integral da imagem , definida pela equação 2 e exemplificando na figura 2:

Equação 2 - Indica como é feito a integral da imagem para melhorar o desempenho.

$$ii(x,y) = \Sigma_{x\leq x', x\leq y}\, i(x',y') \quad (2)$$

$$i(x',y'): \text{Valor do pixel da imagem na coluna } x' \text{ e linha } y'$$

$$ii(x,y): \text{Valor da integral da imagem nas suas colunas } x \text{ e linhas } y$$



$$\sum^{D} pixels = ii(4) + ii(1) - (ii(2) + ii(3))$$

Figura 2 - Avaliação da soma dos valores dos pixels na região D utilizando a integral da imagem.

O início do processo de classificação de uma janela contendo um possível rosto ocorre através dos denominados classificadores fracos que é definido pela função:

Equação 3 - Classificador fraco usado para ser somado e se tornar um classificador forte.

$$h(w,p,f,\theta) = 1, \; se\; pf(w) < p\theta \qquad (3)$$

$$= 0, caso\ contrário$$

$$w : subjanela\ de\ 24 \times 24\ pixels;$$

$$f : feature;$$

$$p : polaridade;$$

$$\theta : treshold;$$

Em um nível acima, se encontram os classificadores fortes. Estes, compostos por diversos classificadores fracos, são definidos pela função:

Equação 4 - Classificadores fortes ponderados pelos classificadores fracos

$$C(w) = \left\{ 1, se \sum_{t=1}^{T} \alpha_t h_t(w) \leq \frac{1}{2} \sum_{t=1}^{T} h_t(w) \right\}$$

$$= \{0, caso\ contrário\}$$

$$\alpha_t : constante\ calculada\ durante\ o\ treinamento.$$

$$h_t(w) : valor\ do\ t-ésimo\ classificador\ fraco;$$

$$T : número\ de\ classificadores\ fracos$$

Por fim, constrói-se uma cascata de classificadores onde, cada camada/nível possui um conjunto distinto de classificadores fortes. Uma representação do funcionamento da cascata de classificadores pode ser vista na Figura 3. Foi projetada para a detecção de casos negativos ser mais rápida, acelerando a execução do algoritmo. De fato, casos negativos não passam por todos os níveis da cascata, sendo rejeitados antes do último nível da cascata. De modo contrário, para uma face ser detectada é preciso que se percorra todas as camadas para confirmar e diminuir o erro de falso positivos no algoritmo. [8]

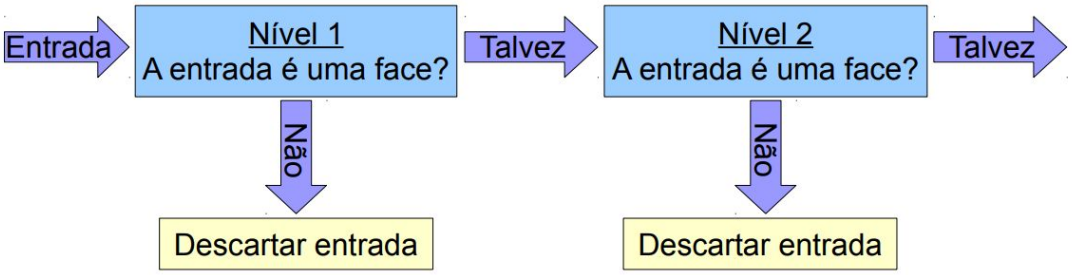


Figura 3 – Diagrama de representação do funcionamento da cascata de classificadores.

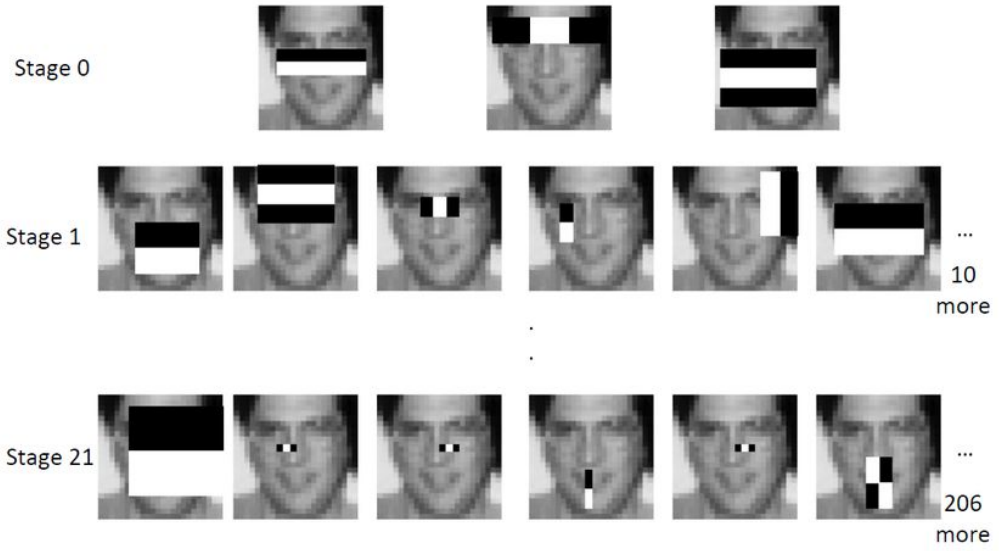


Figura 4 – Representação dos níveis/estágios de cada classificador.

O treinamento de um classificador forte que contenha T classificadores fracos é realizado pelo algoritmo AdaBoost que de forma geral classificações subsequentes feitas são

ajustadas a favor das instâncias classificadas negativamente por classificações anteriores mostrada na figura 5.[9]

Para realizar esse treinamento são necessários os conjuntos de imagens negativas e positivas. No caso do treinamento implementado no OpenCV foram utilizadas 1192 imagens negativas aleatoriamente retiradas do google imagens, já os exemplos positivos foram obtidos no Face Detection Data Set and Benchmark (FDDB) e fornece um arquivo em XML.

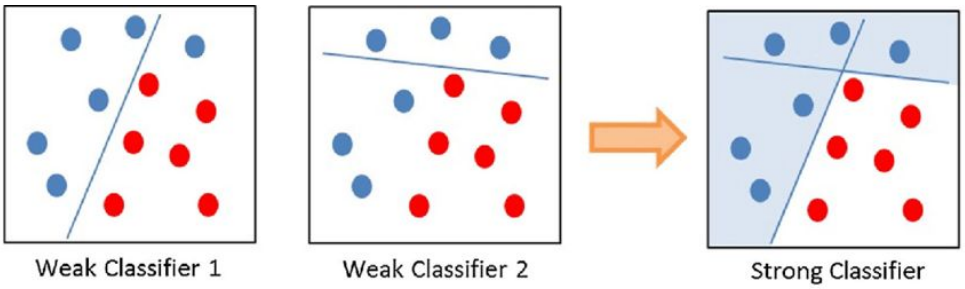


Figura 5 – Sequência de passos do algoritmo AdaBoost.

A figura 6 mostra os resultados obtidos usando 200 *features* de classificação usando Viola-Jones algoritmo para detecção de faces.

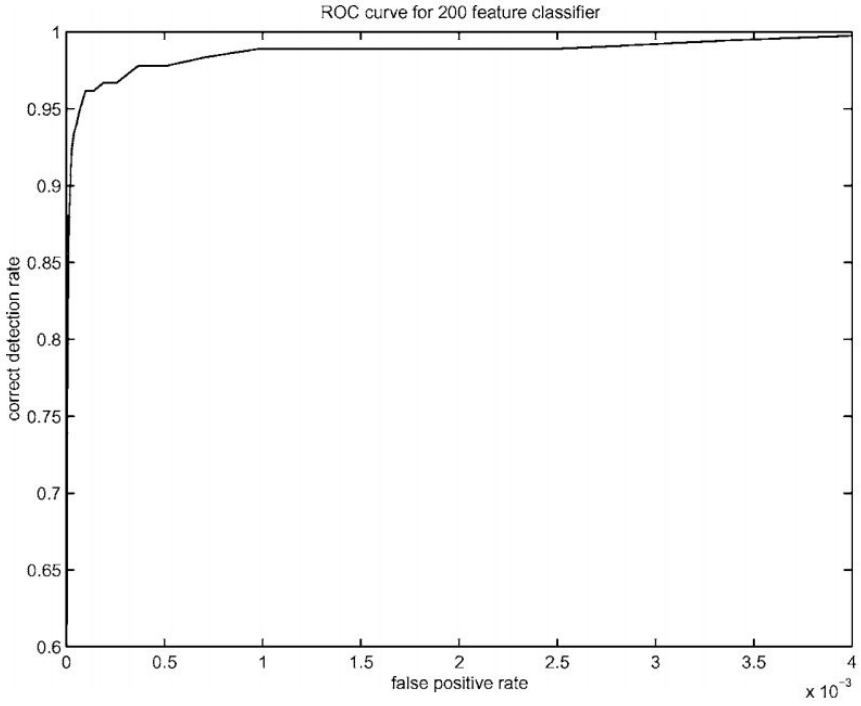


Figura 6 – Curva da taxa de falsos positivos pela taxa de acertos de detecção para um classificador de 200 *features*.

3.2 - Algoritmo de Reconhecimento Facial

O algoritmo de detecção facial recebe como classificador o haarcascade_frontalface_default.xml, disponibilizado pelo OpenCV. Após a detecção, o software captura as imagens do usuário e faz o recorte na imagem fazendo o enquadramento do rosto. O passo seguinte é a aplicação do LBPH. [10]

Originalmente o LBP foi proposto por Ojala, Pietikäinen e Harwood (1996) como um método para análise de texturas, mas com o tempo passou a ser utilizado como para extração de características no processo de reconhecimento e classificação de imagens. Ele leva rotula os pixels de uma imagem de forma binária, levando em consideração uma matriz 3x3, sendo o pixel central o da matriz o limiar para definir o valor dos pixels vizinhos.[6]

O OpenCV disponibiliza a função ***LBPHFaceRecognizer_create*** para a utilização do LBPH. Dentre os parâmetros utilizados por essa função, pode-se destacar 4:

- **Raio(R):** representa o raio ao redor do pixel central, geralmente definido como 1. No openCV este parâmetro recebe o nome de neighbors.
- **Vizinhos(P):** número de pontos para construir o padrão circular local, que será colocado na nova matriz. Normalmente é definido como 8. No openCV este parâmetro recebe o nome de radius.
- **Grade X:** número de células na direção horizontal. Geralmente definido como 8. No openCV este parâmetro recebe o nome de *grid_x*.
- **Grade Y:** número de células na direção vertical. Geralmente definido como 8. No openCV este parâmetro recebe o nome de *grid_y*. [11.]

Existe um procedimento chamado de LBP circular(Figura 7), no qual os valores de R e P podem variar, mas o mais usual é serem utilizados os valores padrões, já citados. A utilização dos parâmetros Grade X e Grade Y terão seus usos explicados mais à frente.[12]

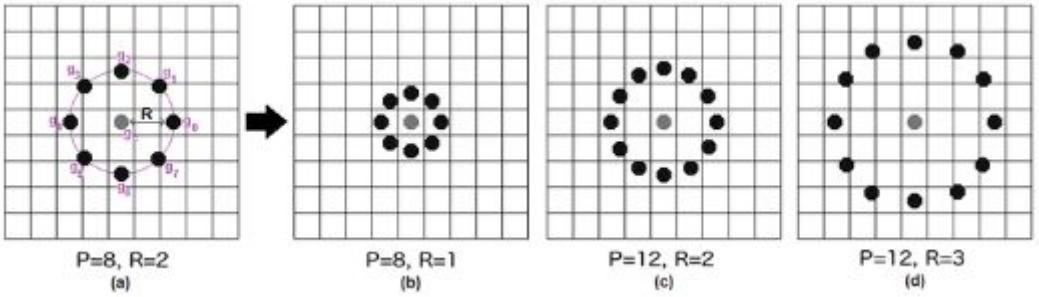
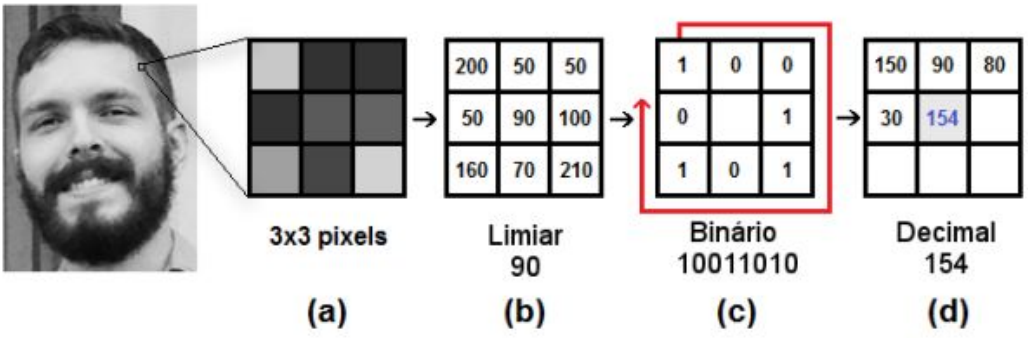


Figura 7 - LBP Circular

Para melhor entendimento do funcionamento do LPBH, considera-se uma parte da imagem, em escala de cinza, como uma matriz 3x3 (gerou-se essa matriz pelos valores padrões de R e P), sendo que esta pode ser representada como uma matriz de inteiros que representam a intensidade de cada pixel. É criada uma nova matriz, onde o pixel central é

utilizado para como limiar para definir os pixels vizinhos, sendo atribuído valor 1 para o pixel que possui valor igual ou superior ao limiar e valor 0 para o pixel que for menor que o limiar. Os binários gerados nessa nova matriz são concatenados em um binário de 8 bits e, a partir deste valor binário, é feita a conversão para o sistema decimal e este valor decimal assume a posição na imagem do antigo valor central. Este processo é aplicado a todos o pixels da imagem.[6]

É possível ver de forma ilustrativa o processo supracitado na imagem a seguir:



Fonte: Kelvin Salton do Prado, 2016.

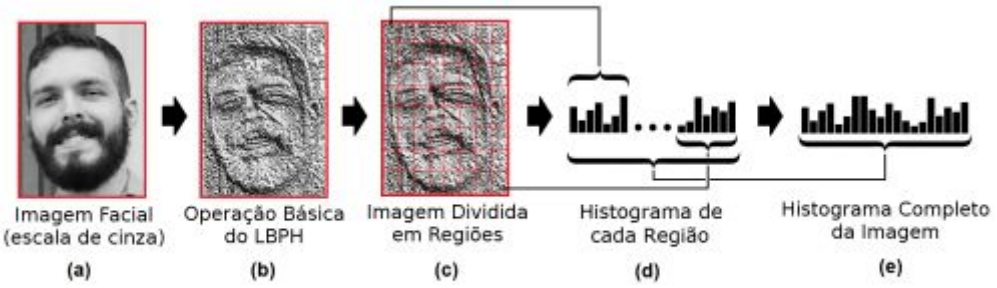
Figura 8 - Processo de aplicação do LBP

Até aqui foi realizado somente o procedimento do Local Binary Pattern (LBP). Após realizado o processo de aplicação do LBP, pode-se observar o resultado:

Fonte: Kelvin Salton do Prado, 2016

Figura 9 - Exemplo de resultado de imagem após aplicação do LBP

A nova imagem gerada é dividida em regiões, normalmente em 8X8 (parâmetros da GradeX e da GradeY), e a partir de cada região são extraídos os padrões. Ao final do processo de extração, os padrões são concatenados e forma-se assim um padrão global da imagem, isto é, o histograma da imagem.[6]



Fonte: Kelvin Salton do Prado, 2016.

Figura 10 - Exemplo de extração do histograma das imagens

Após a extração dos dados das imagens, é realizado o treinamento do algoritmo de reconhecimento e o software gera um arquivo no formato yml que serve de base para o reconhecimento facial.

Ao utilizar o software, este detecta o rosto do usuário pela imagem de vídeo, faz o processamento desta imagem e compara com o arquivo já treinado, retornando a porcentagem de confiança de reconhecimento facial para a imagem de entrada.

4. RESULTADOS OBTIDOS

Inicialmente, os resultados demonstrados neste artigo indicam o decorrer do desenvolvimento do projeto após execução do algoritmo estudado, para ambos, o detector e reconhecedor. Testou-se o algoritmo construído a partir de um sequenciamento de imagens advindos da própria execução do software, obtendo imagens do usuário.Após a captura e o processamento, a imagem do usuário é exibida juntamente da indicação do registro do usuário(no caso, o nome) e a porcentagem de confiabilidade do reconhecimento.

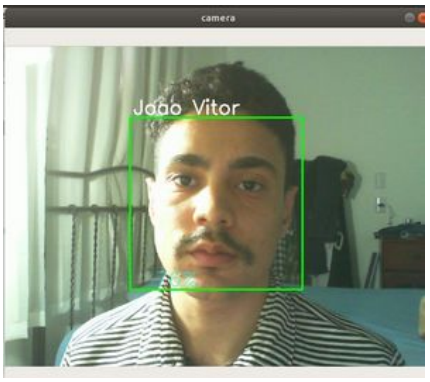


Figura 11 - Resultado final do LBPH aplicado.

É possível observar a partir da figura 11 a validação do usuário. É necessário citar fatores que afetam o processo descrito, como iluminação e interferências externas. A iluminação muito baixa torna difícil a detecção de detalhes na face, assim como o excesso da mesma. Apesar de ser algo bastante configurável e sempre apto a melhorias(utilizando influências externas, como sombreamento)este é um problema bastante recorrente na maioria de sistemas de reconhecimento facial, e com o passar do tempo vem sendo estudado para propor melhorias quanto a algoritmos mais precisos e com boa acurácia. O Plano em que se encontra o usuário deve ser bem definido, pois o algoritmo, ocasionalmente, busca informações externas, como mostrado na figura 12, o que impossibilita de se utilizar o sistema para mais de um usuário por ciclo de operação.

Figura 12 - Interferências Externas.

É interessante descrever as vantagens observadas na utilização do LBPH durante a realização do artigo, em comparação a outros métodos discutidos neste estudo.

- Eigenfaces

Esse algoritmo envolve a projeção de faces em altas e baixas dimensões, um espaço de características. Tais características são conhecidas como Eigenfaces. O algoritmo leva em conta a variação global da imagem da face e, como desvantagem, não consegue lidar com várias condições de iluminação frequentes na imagem, o que, por sua vez, prejudica a estruturação do projeto em basear-se em vídeo em tempo real.[13]

- Fisherfaces

Esse algoritmo, mesmo sob condições variantes de luminosidade, apresenta bons resultados. Caracteriza-se, além da insensitividade a variação luminosa, pela redução de dimensionalidade no espaço de características, porém contrapondo-se pela maior eficácia revelada por taxas de erros menores, quando se há a necessidade de projetar imagens em tempo real. O algoritmo não tem a proposta de lidar com a variações de pose, o que pode ser crucial para a execução do reconhecimento facial. [14]

Os métodos Eigenfaces, Fisherfaces e LBPH, ambos, realizam reconhecimento facial a partir da comparação das faces com um dataset treinado. Tal dataset supre o algoritmo de comparação e indica a qual endereçamento(neste caso, o nome do usuário) o rosto pertence. Há diferenciações fundamentais a respeito do treino do dataset, com relação a cada método de reconhecimento. Eigenfaces e Fisherfaces admitem-se por descrições matemáticas das características mais marcantes do dataset como um todo, ou seja, advém a partir de algoritmos matemáticos a fim de treinar o conjunto. Já o LBPH analisa cada face contida no dataset separadamente e de forma independente, o que compactua com os processos adotados no código (o mesmo cria um dataset quando é inicializado, e graças a análise LBPH isso é facilitado).

O LBPH, em comparação a estes, é bastante simples e eficaz, visto que a caracterização e reconhecimento percorre cada imagem do dataset. Sendo assim, na ocorrência de um novo cadastro, o algoritmo faz a mesma análise, checando apenas os padrões inerentes na nova imagem, ao invés de utilizar recursos extremamente matemáticos, mas sim poucas quantidades de análises. Tal processo reflete a melhor aplicação em detrimento dos outros métodos.

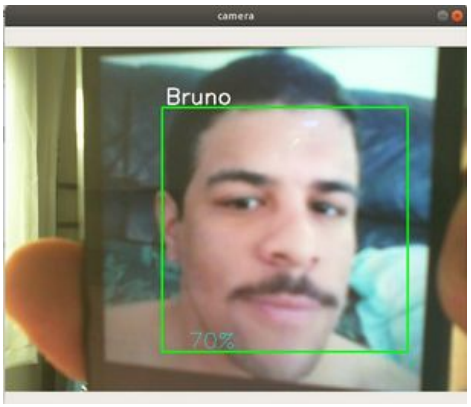


Figura 13 - Aplicação da adição de um novo usuário.

5. CONCLUSÃO

A metodologia LBPH levou a um progresso significativo na análise de textura, sendo um dos algoritmos de reconhecimento facial mais fáceis de compreender. Com o trabalho realizado, foram percebidos a boa representação de características locais nas imagens e a alta probabilidade de obter excelentes resultados, principalmente em um ambiente controlado. É amplamente utilizado em todo o mundo, tanto em pesquisa quanto em aplicações. É robusto contra transformações monotônicas em escala de cinza. Devido ao seu poder discriminativo e simplicidade computacional, o método tem sido muito bem sucedido em muitos desses problemas de visão computacional que não eram antes considerados como problemas de textura, como análise facial e análise de movimento.

Referências

[1]T. Sakai, M. Nagao,T. Kanade. **Computer analysis and classification of photographs of human faces.** First USA-JAPAN Computer Conference, Kyoto University, p.55-62 (1972).

[2]R. Narashiman. **Labeling Schemata and Syntactic Descriptions of Pictures**. Information And Control 7, p.151-179 (1964).

[3]A. Brandon, L. Bartosz, S. Mahadev. **OpenFace: A general-purpose face recognition library with mobile applications.** p.1-2 (2016)

[4]Ranganatha S, Y P Gowramma. **Image Training and LBPH Based Algorithm for Face Tracking in Different Background Video Sequence.** p. 1-2 (2018)

[5]Shireesha C, M. V. Raghunadh. **Automated attendance management system based on face recognition algorithms.** p.26-28 (2013)

[6]SALTON DO PRADO, Kelvin. **Comparação de técnicas de reconhecimento facial para identificação de presença em um ambiente real e semicontrolado**. Programa de Pós-Graduação em Sistemas da Informação, São Paulo, n. CRB-8 4625, p. 41-49, 14 nov. 2017.v

[7]**CV::CASCADECLASSIFIER Class Reference.** [S. l.], 2017. Disponível em: https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html. Acesso em: 4 fev. 2019.

[8]SANTOS, Túlio Ligneul. **Detecção de faces através do algoritmo de Viola-Jones.** Disponível em: https://www.lcg.ufrj.br/marroquim/courses/cos756/trabalhos/2011/tulio-ligneul/tulio-ligneul-report.pdf. p. 1-8, 1 jul. 2011. Acesso em: 4 fev. 2019.

[9]VIOLA, PAUL ; JONES, MICHAEL J. **Robust Real-Time Face Detection. In: INTERNATIONAL JOURNAL OF COMPUTER VISION, 2001.** Disponível em: http://www.face-rec.org/Algorithms/Boosting-Ensemble/16981346.pdf. Acesso em: 5 fev. 2019.

[10]Tsutsumi Kuroiwa, B. ; Carro, S. **DETECÇÃO DE INTRUSÃO COM RECONHECIMENTO FACIAL EM IMAGENS GERADAS POR CÂMERAS DE SEGURANÇA.** São Paulo: Colloq Exactarum, 2015.

[11.]cv::face::LBPHFaceRecognizer Class Reference**.** 2019. Disponível em: <https://docs.opencv.org/3.4/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html>. Acesso em: 2 fev. 2019.

[12]**Reconhecimento Facial: Como Funciona o LBPH.** 2017. Disponível em: <https://updatedcode.wordpress.com/author/kelvinsalton/>. Acesso em: 3 fev. 2019.

[13]**AdaBoost**. 2016. Disponível em: https://pt.wikipedia.org/wiki/AdaBoost. Acesso em: 5 fev. 2019.

[14]**Viola–Jones object detection framework.** 2018. Disponível em: https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework. Acesso em: 5 fev. 2019.