CENTRO UNIVERSITÁRIO UNIAMÉRICA
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JACK CHEN GOMEZ

RECONHCIMENTO FACIAL NO CONTROLE DE ACESSO RESIDENCIAL

FOZ DO IGUAÇU – PR
2019

JACK CHEN GOMEZ

RECONHCIMENTO FACIAL NO CONTROLE DE ACESSO RESIDENCIAL

Trabalho de Conclusão de Curso – TCC apresentado ao Curso de Engenharia Elétrica do Centro Universitário Uniamérica, em cumprimento às exigências para obtenção do grau de Bacharelado em Engenharia Elétrica.

FOZ DO IGUACÇU – PR

2019

JACK CHEN GOMEZ

RECONHCIMENTO FACIAL NO CONTROLE DE ACESSO RESIDENCIAL

Trabalho de Conclusão de Curso – TCC examinado e aprovado pelo Curso de Engenharia Elétrica do Centro Universitário Uniamérica, em cumprimento às exigências para obtenção do grau de Bacharelado em Engenharia Elétrica.

Aprovada em ___/___/_____

Nota:_____

BANCA EXAMINADORA

_____

Orientador (a): Prof(a). Titulação. Nome do (a) Docente
Centro Universitário Uniamérica

_____

Examinador (a): Prof.(a). Titulação. Nome do Membro da Banca Examinadora
Centro Universitário Uniamérica

_____

Examinador (a): Prof.(a). Titulação. Nome do Membro da Banca Examinadora
Centro Universitário Uniamérica

DEDICATÓRIA...

Dedico este projeto à minha família e amigos que sempre estiveram presentes direta ou indiretamente em todos os momentos de minha formação.

Dedico este trabalho a todos aqueles que de alguma forma contribuíram com meu desenvolvimento. Principalmente aos meus pais e irmãos que são sempre presentes.

RESUMO

Este trabalho apresenta o estudo, desenvolvimento e avaliação de um sistema de reconhecimento facial automatizado, que recebe como entrada uma imagem e como saída retorna uma resposta positiva ou negativa no caso de haver ou não, na imagem, faces previamente cadastradas em um banco de dados.

O projeto do sistema consiste em três partes: detecção facial, extração de características e reconhecimento. A detecção facial foi implementada usando-se o algoritmo de Viola-Jones, que tem como base o treinamento de classificadores fortes agregando-se diversos classificadores fracos. Para a extração de características, foi utilizada a análise discriminante, que reduz a dimensionalidade do espaço enquanto preserva o máximo possível de informação discriminatória.

Palavras-chave: Viola-Jones, Reconhecimento Facial, Detecção Facial, Classificador em Cascadas.

ABSTRACT

This work presents the study, development and evaluation of an automatic facial recognition system, that receives an image as input and returns a positive or negative response as output, depending on whether or not there are known faces, previously added to the database, on the image.

The system design consists of three parts: face detection, feature extraction and recognition. Face detection has been implemented using the Viola-Jones algorithm, which is based on training strong classifiers by aggregating various weak classifiers. Feature extraction has been conducted by using discriminant analysis, which reduces the dimensionality of space while preserving the discriminatory information as much as possible.

Keywords: Viola-Jones, Facial Recognition, Face Detection, Haar Cascade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

(Apenas se for necessária. As siglas devem ser organizadas em ordem alfabética. O significado das siglas deve ser redigido com espaçamento simples e justificado. Para facilitar o preenchimento, basta revelar todas as bordas da tabela e ocultá-las ao terminar.)

ARM - *Advanced RISC Machine* (Maquina RISC Avançada);

CISC - *Complex Instruction Set Computer* (Computador com um Conjunto Complexo de Instruções);

RISC - *Reduced Instruction Set Computer* (Computador com um conjunto reduzido de instruções);

KLT - *Karhunen-Loève Transform* (Trasnformada de Karhunen-Loève);

PCA - *Principal Component Analysis* (Analise de Componente Principal);

1. INTRODUÇÃO

Nos últimos anos, debate-se sobre a segurança no sistema de controle de acesso, nos mais diversos métodos empregados, desde documentos de identidade até senhas de autenticação, no entanto documentos e senhas não se tornam tão eficazes, pois a facilidade de falsificação de documentos e o roubo de senha acarretam a necessidade de aprimoramento no sistema de segurança.

A demanda para acessibilidade e segurança do usuário fez com que haja necessidade de pesquisa na área de biometria. Tais quais não dependem de características que são suscetíveis a falsificações ou roubos. Desta forma a utilização de mecanismos biométricos como a face humana para a identificação é de grande valia, o qual não requer de dispositivos especiais para captura, e torna-se mais eficiente e segura se comparadas com o uso dos métodos tradicionais, tais como cartões magnéticos e documentos de identidade [1].

Com o a evolução da microeletrônica nas ultimas décadas a escala/tamanho dos dispositivos diminuíram radicalmente, por exemplo, hoje há microcomputadores do tamanho de um *flash drive*[1] o qual permite os desenvolvimentos de produtos com baixo custo e miniaturizados. Assim, surgiu uma série de sistemas embarcados [2].

Por volta do ano de 1960 iniciou-se a inserção das primeiras pesquisas de reconhecimento facial, na área de engenharia elétrica [4], porém foi na década seguinte que surgiu a necessidade de automatizar o sistema de reconhecimento facial [5]. Desde então, iniciou-se varias pesquisas sobre o assunto nos mais diversos campos da ciência.

As novas tecnologias como o processo evolutivo se tornaram robustas e eficazes [6]. Entretanto, com mais de cinco décadas de pesquisa, ainda há barreiras cientificas e tecnológicas que dificultam o reconhecimento facial automatizado. Um dos exemplos, é que dependendo do sistema empregado, o aumento da taxa de erro aumenta significativamente, isso pode ocorrer quando a luminosidade do ambiente não é muito favorável, quando a leitura da face é captada de perfil e não frontal ou até mesmo se as expressões faciais não forem majoritariamente neutras.

Deste modo, a proposta deste projeto é desenvolver um sistema para detecção de faces de pessoas, com algoritmo, aplica-se em equipamentos que se assemelham aos de ponta hoje em dia. Portanto tem-se um produto com baixo valor, entre tanto possui grande

[1] É um dispositivo de memória constituído por memória flash (EEPROM), capaz de fazer a gravação de dados com uma ligação USB.

portabilidade, eficiência e segurança. Otimizando por exemplo, a entrada e saída de pessoas na residência.

1.1. Motivação

Diversos algoritmos de reconhecimento facial apresentaram-se no decorrer dos anos, porém, nenhum que faça o trabalho com a mesma eficiência que o ser humano isso em qualquer ambiente, vista de qualquer ângulo e sem importância da expressão seja ela neutra, ou não. Entre tanto em condições controladas, existem sistemas que possam ultrapassar a percepção de reconhecimento facial humana, sendo até capazes de diferenciar gêmeos monozigóticos [7].

A vantagem sobre outros métodos de reconhecimento facial via biometria é que a aplicação de seu método não é intrusiva, ou seja, pode ser feita sem que o individuo interaja diretamente com sistema.

1.2. Objetivos gerais

O objetivo almejado neste projeto é a busca do conhecimento sobre o aperfeiçoamento de aplicações para sistemas embarcados. Entre umas das funcionalidades é o processamento de vídeos, imagens, uso dos princípios de visão computacional e o aprimoramento de algoritmos de extração de características faciais e detecção de faces.

1.3. Objetivos específicos

Neste projeto tratará de trazer um sistema de detecção e extração de características faciais, para possíveis aplicações em um sistema de segurança. Esta pesquisa terá como principal foco o alto nível de confiabilidade para que possa em sínteses ser aplicado em sistemas de segurança. Assim com o nível alto de certeza, não ocorram falhas nas detecções de faces e caraterísticas faciais.

1.4. Justificativa

A elaboração de um sistema de reconhecimento facial, empregando uma plataforma embarcada e baixo custo. Utilizando uma webcam integrada é uma excelente escolha para realizar o registro e controle de acesso viabilizando uma possível solução. Além disso, o projeto visa melhorar o discernimento em *Linu*x embarcado [8] assim como o uso da biblioteca *Pillow* e *OpenCV* tais ferramentas que auxiliam no tratamento de imagens e gravações de vídeos.

Em sínteses o projeto desenvolvido, tem a finalidade de reproduzir o reconhecimento facial para possivelmente facilitar o controle de acesso de uma determinada residência. Com tudo utilizando o *softwar*e *Python* no qual é uma linguagem de programação de alto nível [9] facilitará essa comunicação entre os dispositivos físicos e algoritmos.

1.5. Organização do trabalho

Este projeto esta subdividido em 6 capítulos

1. Introdução: Apresenta a historia sobre sistemas de reconhecimento facial, bem como vantagens, processo de aplicações e dificuldade tecnológicas. Além disso, são apresentados os objetivos e organização deste trabalho.
2. Revisão Bibliográfica: Neste capitulo, apresenta-se o estado da arte em relação ao uso de sistemas de reconhecimento e detecção de faces. Serão abordadas as publicações sobre reconhecimento e detecção de faces na literatura especializada.
3. Embasamento Teórico: Neste capitulo, o sistema de reconhecimento facial será abordado com aspecto teórico e são exibidos diferentes métodos e conceitos.
4. Metodologia do Sistema: Explica todos os passos que foram necessitários o desenvolvimento do sistema de reconhecimento facial automatizado. No qual detecta e reconhece a face do cidadão.
5. Resultados: Descreve os resultados obtidos durante o desenvolvimento do sistema de reconhecimento do capitulo 4.
6. Conclusões: Encerra apresentando as conclusões deste projeto.

2. ESTADO DA ARTE

Neste capitulo, apresenta-se o estado da arte em relação ao uso de sistemas de reconhecimento e detecção de faces para obter-se alta taxa de acerto. Neste contexto, são abordadas as publicações sobre reconhecimento e detecção de faces na literatura especializada, utilizadas nesse trabalho.

2.1.Definição do Problema

Em função do processo de globalização mundial é fundamental um mecanismo de controle acesso de pessoas mais sofisticado. Nós últimos anos há grande preocupação do ser humano poder confiar em sistemas de controle de acesso em residências. Através de um sistema de detecção de faces e possível reconhecer a face de pessoa cadastrada no banco de dados e assim liberar o acesso. Por isso à necessidade de um sistema confiável e de alto nível de precisão.

2.2.Análise das principais referencias bibliográficas

Neste trabalho serão analisados os modelos de detecção de faces como método de *Knowledge-Based*, *Template-Based* e *Appearance-Based*. Estas análises restringirão às principais referências que fazem uso desses conceitos e métodos associados a estes temas.

Em 2002 [25] apresentou o método *Knowlegde Based*, que consistia em algoritmos baseados em regras derivadas do conhecimento do pesquisador sobre a face humana. Essas eram geralmente denotados os relacionamentos entre as características faciais, como por exemplo, o formato do rosto, e as distancias entres os olhos, bocas e nariz.

Um pouco mais tarde no mesmo ano, [26] desenvolveu o método *Template Based*, no qual o algoritmo compara uma imagem à outra, em outras palavras, as imagens de teste são comparadas a padrões faciais pré-definidos, que em sua maior parte são gerados como base de uma face frontal. A presença de um rosto na imagem é determinada através de valores de correlação entre ele e o *Template* construído.

A partir de 2002, os estudos sobre métodos de reconhecimento facial passaram a receber mais destaques e muitas pesquisas na área vem sendo desenvolvidas e que associadas

ao uso do conjunto de diferentes algoritmos e hardwares são capazes de obterem ótimos resultados.

Um dos mais importantes métodos de reconhecimento facial refere-se à capacidade de treinamento de imagens. O *Appearance Based* foi desenvolvido em 2005 [28] e as principais diferenças entre os métodos é que este método utiliza padrões faciais os quais são gerados por meio de um grupo de imagens de treinamento, capturando as variações presentes em todas as faces do conjunto. Posteriormente, eles são utilizados na comparação com a imagem de teste que se deseja classificar. Além disso, existe a possibilidade de que técnicas como essas sejam combinadas a métodos de redução de dimensionalidade, com o objetivo de minimizar o custo computacional.

3. FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento deste capítulo, adotou-se um estudo para todos os componentes necessários. Cada hardware e software possuem características que adequa com o sistema de reconhecimento facial. Entre tudo, o sistema deve ter alta confiabilidade, alto rendimento e custo baixo, para que possa facilmente ser implantado.

3.1.Sistemas Embarcados

Conforme dados estimados, por pesquisadores na área de alta tecnologia, aproximadamente 90% dos microprocessadores fabricados basicamente são destinadas a maquinas que regularmente não são chamadas de computadores. Entre algumas destas maquinas estão os aparelhos celulares, eletrodomésticos, calculadoras, impressoras e videogames [10].

A diferença do conjunto de dispositivos citados acima e do computador convencional (PC-*Desktop*/*Notebook*), é o conjunto dedicado e especial de hardwares, softwares e placas de expansão [10].

Diferente do computador convencional, o sistema embarcado realiza tarefas pré-definidas, as quais possuem seus requisitos específicos [11]. Visto que, o sistema é dedicado a determinadas funções. Contudo através da engenharia pode-se automatizar projeto, reduzindo o seu tamanho, o custo do produto e recursos computacionais [12].

De modo geral, tais projetos não dispõem maneiras acessíveis de alterar suas funcionalidades durante o uso. Neste caso, se houver a necessidade de modificar o proposito, há de reprogramar o sistema para a nova tarefa [12].

Entre as diversas arquiteturas apresentadas nos sistemas embarcados, o que lhe define é o tipo de microcontrolador que é utilizado [13]. Exemplo do INTEL 8051, lançado em 1977 pertencente à família *MSC-5*, dispõe de 8 *bits* e é considerado um dos mais populares do mundo, pelo fato de possuir um conjunto de normas *CISC* lhe oferecendo um vasto conjunto de instruções [14].

Existe outra gama de microcontroladores, além destes citados acima, exemplo do *ARM*, o qual é amplamente utilizado em *tables*, celulares, *drones* e *dongles*. Possui uma arquitetura de 32 *bits* e conjunto de normas *RISC*. Quando comparada com a arquitetura *CISC*, tem-se a diferença de numero de instruções, portanto reduzem os custo e energia, tornando estas as principais vantagens dos microcontroladores *ARM* [15].

3.2. Sistema operacional

O sistema operacional é um conjunto de programas, dos quais a função é administrar os recursos do sistema (controle de memoria, criação de arquivos e administração de programas por meio do processador) estabelecendo uma interface entre o computador e o usuário.

A maioria dos usuários de computador já teve experiências com algum tipo de sistema operacional, embora seja difícil expressar diretamente o que é um, entretanto o sistema operacional basicamente possui duas funções que não são relacionadas.

Existem dois modelos de conceituar um sistema operacional, para usuário ou programador, sendo elas: (*top-down*) na qual é uma abstração do hardware, tem a função de ser intermediário entre programas (*softwares*) e os componentes físicos de um computador (*hardware*) e também existe o (*bottom-up*) o qual gerencia recursos, dispositivos de entrada e saída, refere-se ao controle de aplicações e processo de executar, como, a utilização de recursos (memoria, disco e periféricos) [16].

Para [17] um sistema operacional é desenhado para ocultar as características de *hardware* (ditas "de baixo nível") e, de acordo com seu desempenho há a necessidade de desenvolver uma máquina abstrata a qual proporciona às aplicações e serviços compreensíveis ao usuário (ditas "de alto nível").

3.2.1. Linux

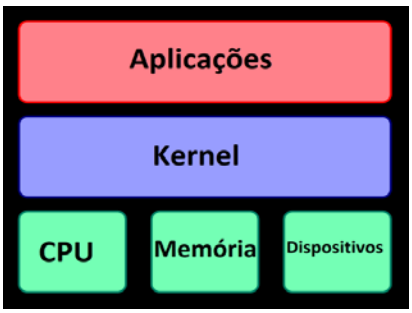
O sistema operacional *Linux* mudou os padrões de sistemas operacionais proprietários [18]. O sistema proprietário é um *software* para computadores o qual é licenciado com direitos exclusivos apenas para o produtor [19].

A diferença dos sistemas operacionais proprietários é que apresentam um ambiente fixo, onde as aplicações são limitadas e não proporciona flexibilidade para customiza-la, melhorar o desempenho ou até mesmo corrigir *bugs*. São denominados de sistemas *opens sources*, Entre eles, está o sistema operativo *Linux*, o qual possibilita a customização e redistribuição com acesso completo ao código fonte do sistema operativo [18].

Em sínteses, o sistema operativo *Linux* tornou-se popular, na grande parte por sua eficiência, qualidade e gratuidade. A sua utilização é ampla, quando se trata de supercomputadores e sistemas embarcados.

O termo *Linux* trata-se do próprio *Kernel*[2] desenvolvido por Linus Torvalds, o qual pode ser rodado pelo sistema operacional GNU/*Linux*. O *Kernel* é a base com um núcleo de ferramentas responsáveis pelos serviços básicos do sistema operacional GNU/*Linux*, dentre eles: a administração de recursos, abstração para aplicações do usuário e gerenciamento de sistema e arquivos [20] [21]. Na figura 1 pode-se observar a junção do *Kernel*, com o *software* e *hardwares*.

Figura 1 - Núcleo de sistema conecta o software aplicativo ao hardware de um computador.



Fonte: Autoria própria (2019) – Wulf 74 pp.337345.

[2] *É o componente central do sistema operativo da maioria dos computadores.*

No entanto, o *Kernel* por si só não oferece utilidade para o usuário, não é capaz de ter controles de terminais de comandos básicos, realizar edições de arquivos ou comunicar-se via rede.

3.3 Detecções de faces

A detecção de face depende do uso de métodos computacionais que confirmam a existência de uma face, em uma imagem digital, de vídeo ou fotografia [22].

Apesar de que a tarefa de detectar faces seja um exercício simples para os humanos, o desenvolvimento de sistemas computacionais que façam esse tipo de ocupação ainda é complexo, isto se deve as barreiras impostas pelos recursos que serão utilizados, para realização da detecção de faces.

Segundo [23], existem várias dificuldades que podem ser vistas no processo computacional para detecção de face, dentre elas é possível citar: a variação no posicionamento da face existente na imagem, a presença e/ou ausência de características especifica estruturais, como a como barba a que pode modificar características relacionadas à forma, tamanho e cor da face, as expressões faciais, a obstrução da face por determinado objeto do ambiente e condições da imagem referente à: iluminação (espectro e intensidade) e a características técnicas da câmera.

Entretanto há algumas abordagens que aprimoram o processo de detecção de face, melhorando os resultados obtidos. Uma destas é o algoritmo de detecção que se baseia em cores, consiste em verificar a cor da pele, realizando a detecção da cor em cada *pixel*[3] e o classificando de acordo com os parâmetros de cor pré-determinados [24].

No sistema de extração por cores, o algoritmo deve ser suficientemente capaz, de detectar a face, independentemente da cor da pele e das variações nas condições de iluminação [22].

O algoritmo faz a leitura da segmentação tomando por base a informação relativa à cor. Depois deste processo, sobra uma camada de *pixels* a qual não é considerada pele. Entre tanto estes *pixels*, mesmo que de maneira esparsa, ainda são visíveis dificultando a detecção da face.

[3] É o menor elemento em um dispositivo de exibição (por exemplo, um monitor)

Na sequencia é realizado um processamento destas falhas, denominado processamento de imagem, usando a morfologia matemática. Este meio apresenta uma solução para descartar os *pixels* que foram indevidamente selecionados como pele.

O processamento morfológico ocorre da seguinte maneira: é feito a abertura (erosão seguida de dilatação) para remoção dos *pixels* imperfeitos, que foram classificados como pele, posteriormente é realizada uma reagrupação (através do fechamento, ou seja: dilatação seguida de erosão) dos *pixel*s considerados como pele [24].

O algoritmo de detecção de cores, citado a cima, esta baseado apenas em características invariantes. No caso, este método se baseia em técnicas que possuem susceptibilidades a erros, quando há necessidade de identificar faces em movimento. Além disso, esta abordagem apresenta dificuldade em lidar com condições adversas (baixa iluminação e ruídos) [25].

Tal método realiza classificação através de padrões (conhecimentos) pré-estabelecidos, realizando, desta forma, a detecção da face dos indivíduos. Por exemplo, calcula-se que a os seres humanos, em sua condição classificadas normais, possuem determinadas características invariáveis, como: nariz, um par de olhos e boca, distribuídos de maneira especifica sobre a face.

Através destas condições são estabelecidas regras para a análise de uma face humana. No entanto este método está ligado às regras que lhe são necessárias através do seu algoritmo, se as regras não são bem especificadas o resultado do reconhecimento da face será afetado, podendo não condizer com a realidade, porém se as regras impostas forem muito especificas qualquer deformidade existente na face causará o não reconhecimento da face [25].

Existe também o processo de reconhecimento de face baseado na aparência, este processo é o oposto ao mencionado anteriormente, pois não está associado às regras determinadas a priori e nem à propriedades que podem estar submetidos aos padrões externos, os quais podem desgastar os recursos utilizados para a detecção da face.

O *Eigenfaces* apresentado por [26] em 1991, é baseado em PCA ou KLT, a qual se fundamentou no trabalho de Sirovich e Kirby que comprovou a eficiência na representação de figuras.

Segundo [26], as imagens que são encontradas, não estão distribuídas de forma aleatória em um espaço de alta dimensionalidade, logo elas podem ser descritas em um espaço de menor dimensionalidade. A transformada de KLT é utilizada, para encontrar os vetores que

descrevem as imagens dentro deste espaço. Tais vetores são chamados de *EigenFaces*, pela semelhança a que eles representam nas imagens.

3.4.Algoritmo Viola-Jones

Em 2001, Paul Viola e Michael Jones, desenvolveram um algoritmo de identificação de imagem denominado de algoritmo de Viola-Jones. Esse algoritmo era capaz identificar faces utilizando os padrões que existem no rosto humano, também pode diferenciar qualquer outro objeto, desde que seja aprendido [28].

A abordagem de *Viola* e *Jones* baseavam-se em três conceitos: integral de imagem, treinamento de classificadores usando *boosting* e o uso de classificadores em cascata. Embora o algoritmo quando treinado pode reconhecer qualquer objeto, a principal motivação de *Viola* e *Jones* era a de reconhecimento facial. A vantagem deste algoritmo é extrema rapidez com que é executado [29].

A integral de imagem ou popularmente chamada tabela de soma de áreas, é um algoritmo, proposto por [30] em 1984, consiste em avaliar eficientemente a soma dos *pixels* (intensidade dos níveis de cinza) de uma área retangular nas regiões da imagem. A equação 1 mostra como calcular a integral de imagem em uma determinada coordenada.

Equação 1 – Calcula Integral de Imagem

$$ii(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

Onde $ii(x,y)$ é a integral da imagem nas coordenadas do *pixel* (x,y) e $i(x,y)$ é a imagem original. Observa-se que a integral da imagem na coordenada (x,y) é a soma dos *pixels* citados acima, de y e à esquerda de x, inclusive x e y (considerando que a origem do sistema de coordenadas está localizada no canto superior esquerdo da imagem). A tabela de soma pode então ser calculada para todos os *pixel*s somente em uma varredura como mostra na equação 2.

Equação 2 – Calcula todos os pixels em uma varredura.

$$ii(x, y) = i(x, y) + ii(x - 1, y) + ii(x, y - 1) - ii(x - 1, y - 1)$$

Então, $(x, -1) = 0$ e $(-1, y) = 0$ serve para contornar em casos que as coordenadas dos *pixels* estão fora dos limites da imagem.

Utiliza-se esse modo para encontra com maior facilidade a soma de área em qualquer região retangular da imagem. Então, em um região retangular ABCD de uma imagem Figura 2, a soma das identidades dos *pixels* podem ser calculadas usando a equação 3.

Equação 3 – Soma das identidades dos *pixels*

$$\sum_{(x,y)\in ABCD} i(x, y) = ii(A) + ii(D) - ii(B) - ii(C)$$

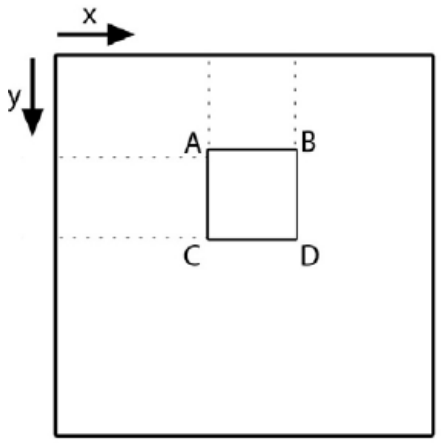


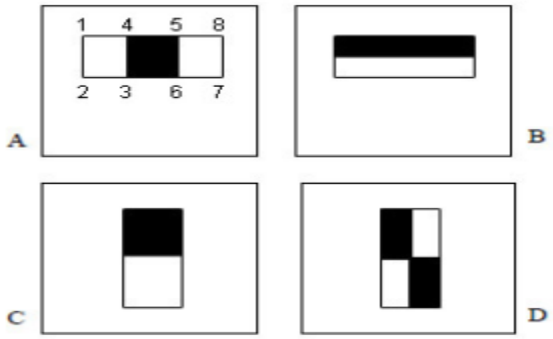
Figura 2 – Região ABCD em uma matriz de pixels.

Calculando a integral de imagem é possível identificar padrões utilizando as características *Haar-like*[4]. A figura 3 demostra quatro possíveis tipos de características-base possíveis, e que são calculadas, subtraindo a soma dos valores dos *pixels* da região branca, do valor somado dos *pixels* da regia preta.

[4] É um mecanismo que é utilizado para subtração de pixels de uma região determinada, a outra região.

É necessário, para o cálculo das características A, Figura 3, oito consultas à tabela de soma de áreas (integral de imagem), respectivamente para os pontos indicados.

Figura 3 – Representação visual de quatro tipos de características *Haar-like*



3.4.1. Outras abordagens

Existem inúmeras abordagens que tratam de solucionar os problemas de detecção facial. Algumas aplicam variações nas ideias propostas de *Viola* e *Jones.* De modo geral, as abordagens são compostas de três métodos: método baseados em conhecimento (*Knowledge-Based*), métodos com base em modelos (*Template-Based*) e métodos que se baseiam em aparência (*Appearance-Based*).

Métodos Knowledge-Based procuram descrever os padrões da face, utilizando regras caracterizadas na fisionomia humana, como por exemplo: na face humana observam-se dois olhos, uma boca e um nariz. A partir destas regras, são determinadas relações entre as características que as definem, o contraste, distâncias relativas e posições [30].

Os processos do *Template-Based* se diferenciam do *Knowledge-Based,* pois não procuram padrões que se baseiam em um conjunto de regras manuais, mas buscam representar a face parametricamente por meio de pontos de controle, que se deformam a fim de encontrar na imagem um padrão, com isso alinha-se o modelo numa possível face. Esses pontos de controle do modelo são definidos de modo a apresentar as principais características da face [31].

Por último, o método *Appearance-Based* tem como base, entender características sobre a face humana, aplicando algoritmos de aprendizagem de máquina que são treinados

com uma enorme quantidade de imagens, tanto de faces quanto de outros objetos. De forma resumida, esses métodos aplicam análises estatísticas a fim de encontrar características discriminantes entre faces e não faces. As características aprendidas são modelos de classificação. O algoritmo de *Viola Jones* se enquadra nesta categoria.

4. MATERIAL E MÉTODOS

O objetivo desta parte do trabalho é implementar um sistema de reconhecimento facial totalmente automatizado. O projeto do sistema completo envolve as três etapas descritas no capítulo anterior: Detecção facial, Extração de Características e Reconhecimento. Neste capítulo serão detalhados os métodos escolhidos e as ferramentas usadas para o desenvolvimento de cada uma das etapas.

4.1. Plataforma de Desenvolvimento

A plataforma de desenvolvimento escolhida para elaborar este projeto foi o *Python*. A principal razão que motivou tal escolha foi pelo fato do software em questão oferecer uma vasta biblioteca de funções matemáticas e algoritmos numéricos, além de um toolbox próprio para processar imagens (disponível somente em algumas versões). Outro ponto forte é a simplicidade da linguagem, que faz com que seja possível desenvolver programas complexos em um curto espaço de tempo e de forma intuitiva. Além disso, oferece boa portabilidade, sendo compatível com Windows, Linux.

4.2. Banco de Faces

Para testar e avaliar um sistema de reconhecimento facial é imprescindível que se tenha um banco de dados com imagens de várias pessoas, não apenas sozinhas, mas em grupos e em diferentes configurações de iluminação e pose. É recomendável que cada pessoa cadastrada no sistema, possua pelo menos duas imagens, pois não pode haver interseção entre as imagens-teste (usadas para avaliar a qualidade do sistema) e a imagens usadas para o cadastro. E a imagem-teste deve ser obviamente, diferente da imagem cadastrada, seja pelo

ângulo da câmera em relação à face, ou pela iluminação ambiente, ou expressão facial, etc. Dependendo do tipo de extrator de características usado, pode ser desejada mais de duas imagens diferentes por pessoa para o cadastro no banco de dados, além da imagem-teste.

4.2.Detecção Facial

Este módulo do sistema recebe como entrada uma imagem que pode ou não conter faces humanas, e como saída retorna imagens somente das faces, caso estas existam. O algoritmo escolhido foi o de Viola-Jones por sua eficiência e baixa taxa de erros. O maior desafio deste método é treinar o sistema exaustivamente com uma vasta compilação de exemplos positivos (faces) e exemplos negativos (não-faces). É importante que essa coleção de imagens de faces abranja pessoas de diferentes idades, etnias, com barba, sem barba, com diferentes estilos de cabelo e diferentes expressões faciais.

A implementação desta parte do projeto foi divida em três submódulos: cálculo das características *Haar-like*, treinamento dos classificadores usando *AdaBoost* e Varredura da imagem de entrada em busca dos padrões treinados. Os códigos desenvolvidos nesta parte projeto podem ser encontrados no Apêndice A.

4.3.1. Catacteristicas *Haar-like*

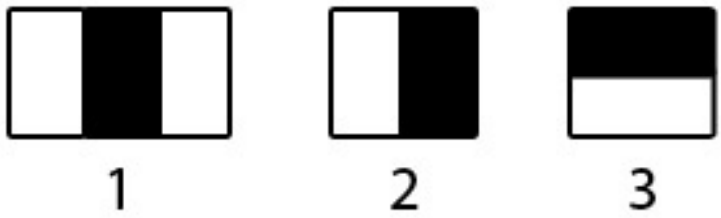
Cada característica (ou padrão) representa uma diferença de intensidades entre áreas da imagem. O número total de características possíveis para uma dada janela depende de quantas bases são usadas. Foram escolhidas três bases (figura 1) que, de acordo com relatórios de pesquisa, obtiveram um bom desempenho, e a janela usada foi de 24x24 pixels (dimensão das imagens de treinamento). Portanto existem 114.000 diferentes características nessa janela considerando-se as três bases escolhidas. Isso porque as características podem sofrer alterações de escala e posição dentro da janela. É importante notar a diferença entre a característica em si e seu respectivo valor quando computado em uma região da imagem: A característica é apenas uma máscara que pode sofrer alteração de posição e escala dentro de uma região; quando essa máscara é aplicada em uma área da imagem, computa-se então seu valor, e é o valor associado à característica que permite diferenciar regiões. O algoritmo (1) para o cálculo dos valores de todas as características *Haar-like*, para as bases escolhidas, em

uma dada imagem, pode ser descrito da seguinte maneira:

Algoritmo 1 - Cálculo das características Haar-like

- Calcular a Integral da Imagem para todos os pixels, onde a Integral no ponto (x, y) é igual à soma dos valores dos pixels menores ou iguais a x e y.
- Definir a base da característica *Haar-like* em altura e largura, onde estas definem quantas divisões existem em cada eixo. Por exemplo, no caso da característica 1 da figura 4, seria largura 3 e altura 1, pois existem duas retas verticais dividindo o retângulo, e nenhuma reta horizontal.
- Para cada mudança de escala da base, deslocar a característica ao longo de toda a imagem.
- Para cada posição da característica dentro da imagem, calcula-se o valor das diferenças de intensidades baseado no padrão da base.
- Repetir o processo, a partir do segundo passo, para cada base escolhida.
- Repetir o processo para todos os exemplos positivos e negativos.

Figura 4 – Bases *Haar-like* usadas no projeto.



A partir desse algoritmo, derivaram-se todas as características possíveis numa imagem de 24x24 pixels, usando-se essas três bases, e cada uma dessas características foi então calculada para cada um dos exemplos positivos e negativos. Neste passo, não foi escolhida nenhum característica ótima, portanto todas são consideradas classificadores fracos. O próximo passo é analisar o conjunto de todas as características e escolher aquelas que possuem o menor erro de classificação para então combiná-las em um classificador forte.

4.3.2. Treinamento usando *AdaBoost*

O algoritmo de aprendizagem de máquina desenvolvido neste projeto foi baseado em [29], e tem como premissa usar uma série de classificadores fracos para formar um classificador forte, e então cascatear vários estágios de classificadores fortes. O algoritmo abaixo (2) descreve o processo de formação do classificador forte:

Algoritmo 2 - Treinamento de classificadores utilizando AdaBoost

- Seja um conjunto de exemplos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, onde x_i são as imagens, y_i é 0 para exemplos negativos e 1 para exemplos positivos, e n é o número total de exemplos

- Inicializar os pesos $w_{1,i} = \frac{1}{2m}$ onde m é o número total de exemplos da Categoria. Por exemplo, se tiverem 100 exemplos positivos e 50 negativos, os pesos iniciais dos exemplos positivos e negativos serão Respectivamente $\frac{1}{2\times100}$ e $\frac{1}{2\times50}$.

- Para t=1 até T (número de classificadores fracos) faça:

  1. Normalizar os pesos $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

  2. Selecionar o melhor classificador fraco de acordo com o seguinte erro ponderado:

$$c_t = \min_{f,p,\theta}\left(\sum_i w_i |h(x_i, f, p, \theta) - y_i|\right)$$

  3. Definir $h_t(x) = h(x, f_t, p_t, \theta_t)$, onde f_t, p_t e θ_t são minimizadores de c_t. f_t é uma característica *Haar-like*.

  4. Atualizar os pesos:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

  caso contrário, e $\beta_t = \frac{c_t}{}$

- O classificador forte é então dado por:

$$C(x) = \begin{cases} 1 \text{ se } \sum \alpha_t h_t(x) \leq \frac{1}{2}\sum \alpha_t \end{cases}$$

onde $\alpha_t = \log\frac{1}{}$

A partir desse algoritmo cria-se um classificador forte combinando-se *T* classificadores fracos. Para criar a cascata de classificadores fortes basta repetir o processo para uma diferente combinação de imagens (ou pelo menos mudando parte dos exemplos negativos) até obter o número de estágios desejado.

4.4. Detecção em múltipla escala

Com os classificadores já treinados, o próximo passo é implementar um módulo que busque na imagem de entrada os padrões desejados e identifique as regiões classificadas corretamente.

Para que a detecção facial seja feita em múltipla escala, é necessário redimensionar a imagem ou a janela de detecção iterativamente. Como a janela base dos padrões é de 24x24 pixels, qualquer face na imagem que seja muito maior do que esse tamanho, não será detectada corretamente. Optou-se por reduzir a escala da imagem ao invés de ampliar a escala da janela, porque este último acaba tornando o algoritmo mais lento. Foi usada uma técnica chamada Pirâmide de Imagens (*Image Pyramid*) na qual, a partir da imagem original, é criada uma coleção de imagens com escala reduzida. Para cada escala, a janela de detecção varre a imagem inteira (figura 4). O algoritmo abaixo (3) detalha o processo de detecção:

Algoritmo 3 - Detecção em múltipla escala

- Escolher um fator de escala para criar a pirâmide e os incrementos de posição nos dois eixos para o deslocamento da janela.
- Enquanto a área da imagem for maior do que a área da janela de detecção, reduzir a escala.
- Para cada escala da imagem, calcular a integral de imagem para todos os pixels.
- Deslocar a janela de detecção ao longo de toda a imagem.
- Para cada posição da janela na sub-região da imagem:

1. Para cada estágio do classificador forte aplicar todos os classificadores fracos.
2. Se a classificação foi positiva ao final de um estágio, prosseguir para o próximo, caso contrário, terminar a classificação para essa posição e deslocar a janela para a próxima

posição.

3. Se a classificação foi positiva para todos os estágios, salvar as coordenadas da região.

- Ao final do processo de varredura em todas as escalas, analisar as regiões classificadas corretamente e agrupar as sobreposições.

O último passo do algoritmo é importante porque como a varredura é feita em múltiplas escalas, frequentemente quando há faces na imagem, a mesma região classifica positivamente várias vezes criando sobreposições. Dependendo de quão pequeno seja essas sobreposições há incremento do deslocamento, uma mesma face pode estar presente em mais de uma janela ao longo da detecção. O método usado para agrupar as sobreposições foi encontrar as intersecções das regiões classificadas positivamente (duas de cada vez) e caso a área de intersecção seja maior do que 1/3 do menor retângulo, então o menor retângulo é descartado e o maior permanece.

4.5. Extração de Características de Reconhecimento

Neste módulo do projeto, o objetivo é desenvolver um algoritmo capaz de identificar com confiabilidade, nas imagens de entrada, faces cadastradas no banco de dados. É importante que o sistema seja o máximo possível invariante às condições ambientes (e.g: iluminação) e às expressões faciais. Os códigos em *Python* desenvolvidos neste módulo podem ser encontrados no Apêndice A.

Este módulo foi divido em duas funções: Criar e Testar. A primeira função cria um conjunto de parâmetros que definem matematicamente as imagens do banco de dados. A segunda função recebe como entrada uma imagem qualquer (não pertencente ao banco de dados) e analisa a proximidade desta com as do banco de dados através dos parâmetros matemáticos.

O método escolhido para a extração de característica foi o *LDA* porque este oferece um bom poder discriminatório e apresenta um bom desempenho em termos de tempo de execução. A ideia central é reduzir a dimensionalidade do espaço, dada pelas dimensões da mesma, para o número de indivíduos cadastrados no banco de dados. O algoritmo (4) a seguir indica quais passos foram usados neste projeto para o desenvolvimento de um extrator de características baseado em *LDA* a partir de um banco de imagens de faces.

Algoritmo 4 - Análise Linear Discriminante

- Seja N o número total de imagens a serem analisadas e M o número de indivíduos;
- Normalizar as N imagens para que tenham a mesma largura W e altura H;
- Transformar as imagens em um vetor-coluna de dimensões $(WH) \times 1$;
- Seja x_i a i-ésima imagem vetorizada e C o número de classes (cada classe representa um indivíduo);
- Calcular a média de todas as imagens (μ) e as médias das classes $(\mu_1, \ldots, \mu_M)$;
- Criar uma matriz A de ordem $WH \times M$, na qual cada coluna é dada por $\mu_j - \mu$, para j de 1 até M;
- Cria-se outra matriz B de ordem $WH \times N$ na qual cada coluna é dada por $x_i - \mu_j$, para i de 1 até M, onde μ_j é a média da classe a qual pertence a imagem vetorizada x_i;
- Sejam S_b e S_w respectivamente as matrizes de dispersão entre classes e dentro da própria classe. Então $Sb = A \times A^T$ e $S_w = B \times B^T$, ambas de ordem $WH \times WH$;
- Definir as matrizes auxiliares $C = A^T \times A$ e $D = B^T \times B$, ambas de ordem $M \times M$;
- Seja V a matriz dos autovetores de C;
- Ordenar em ordem decrescente os autovetoers da matriz V de acordo com os autovalores;
- Encontra-se então M autovetores de S_b, que são dados por $A \times V$;
- Cria-se uma matriz diagonal D com os autovalores de $A \times V$;
- Calcular a matriz $Z = (A \times V) \times D^{-1/2}$;
- Definir a matriz auxiliar $E = Z^T \times B$;
- Calcular os autovetores de $E \times E^T$ e formar uma matriz U com esses vetores;
- O subespaço final é então dado por $T = Z \times U$;
- Projeta-se a média de cada classe (cada coluna da matriz A) nesse subespaço.

Depois que todas as variáveis forem calculadas e as imagens do banco de dados forem projetadas no subespaço dos autovetores, são salvos em um arquivo a matriz de projeção, as matrizes T e A, e os escalares W, H, N. Com isso, as imagens do banco de dados não são mais necessárias no processo de reconhecimento. Porém, cada vez que forem adicionados novos indivíduos ao banco de dados, é necessário computar novamente todas as matrizes.

Para o reconhecimento, foi utilizada uma métrica baseada na distância euclianda. Primeiro, normaliza-se a imagem-teste para que tenha as mesmas dimensões das imagens que foram usadas para criar o subespaço. Depois se transforma a imagem-teste em um vetor coluna, subtrai-se o vetor A e então se projeta a mesma no subespaço T. Calcula-se a norma da diferença entre a projeção da imagem-teste e cada projeção salva no banco de dados.

A projeção que originou a menor norma é a mais próxima da imagem- teste, mas não quer dizer necessariamente que há uma correspondência positiva, uma vez que só por ser a imagem mais próxima não significa ser próximo do suficiente, daí a necessidade de se estabelecer um critério de aceitação.

O critério de aceitação adotado neste projeto baseia-se em analisar a razão entre a menor distância e a segunda menor distância. Se esta razão for muito pequena, a chance da imagem-teste corresponder a mesma classe da imagem de menor distância é elevada.

5. RESULTADOS E DISCUSSÃO

Esta parte do trabalho apresenta os resultados obtidos nos testes dos módulos do sistema de reconhecimento e também descreve como os testes foram realizados. Os códigos desenvolvidos para realizar os testes podem ser encontrados no Apêndice A.

5.1. Resultados Obtidos

Depois que o algoritmo do detector facial foi desenvolvido, foi utilizado a minha imagem para testar a eficiência e a acurácia do detector. Estas imagens foram inseridas no momento do funcionamento do algoritmo no Anaconda *Python*. O módulo de detector recebe como parâmetros: a imagem de entrada, uma escala inicial, um fator de escala e incrementos de posição da janela de detecção (de quanto em quanto a janela se desloca nos eixos x e y).

Para que fosse possível visualizar as detecções, foi desenvolvida uma pequena rotina que desenha na imagem os retângulos dados pelas coordenadas da saída do detector.

O primeiro passo é colar o diretório no Anaconda *Prompt* com o comando “cd” conforme mostra a figura abaixo.

Figura 5 – Comando “cd” para execução direta do diretório

Depois de inserido o diretório no comando “cd” o algoritmo python pode ser executado diretamente da pasta em que se encontra todos os algoritmos necessários para a detecção de faces.

O segundo passo é executar o comando “python Face_Capture _With_Rotate.py” o qual faz o cadastro dentro da base de dados (nome e quantidades de pessoas cadastradas).

Após executar o comando, e cadastrar a pessoa na base de dados, a webcam fotografara 50 fotos para cadastrar na base de dados, conforme a figura 6 e 7.

Figura 6 – Comando para o cadastramento da pessoa

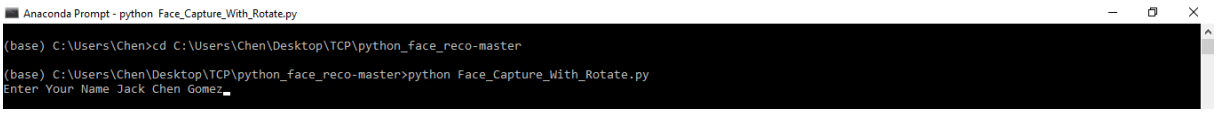
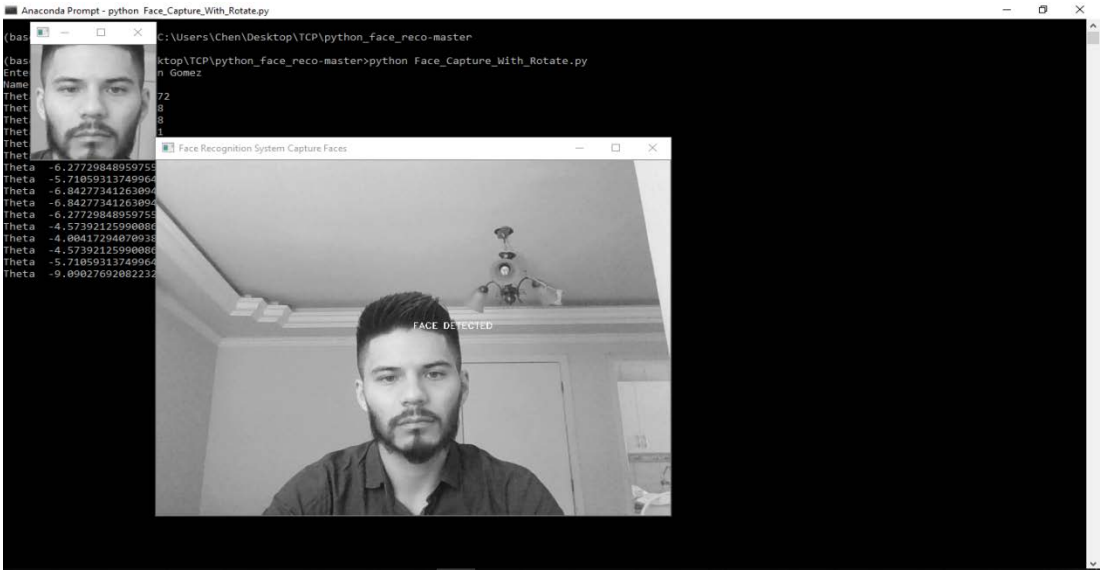


Figura 7 – Tomada de imagens da pessoa para o cadastramento no banco de dados

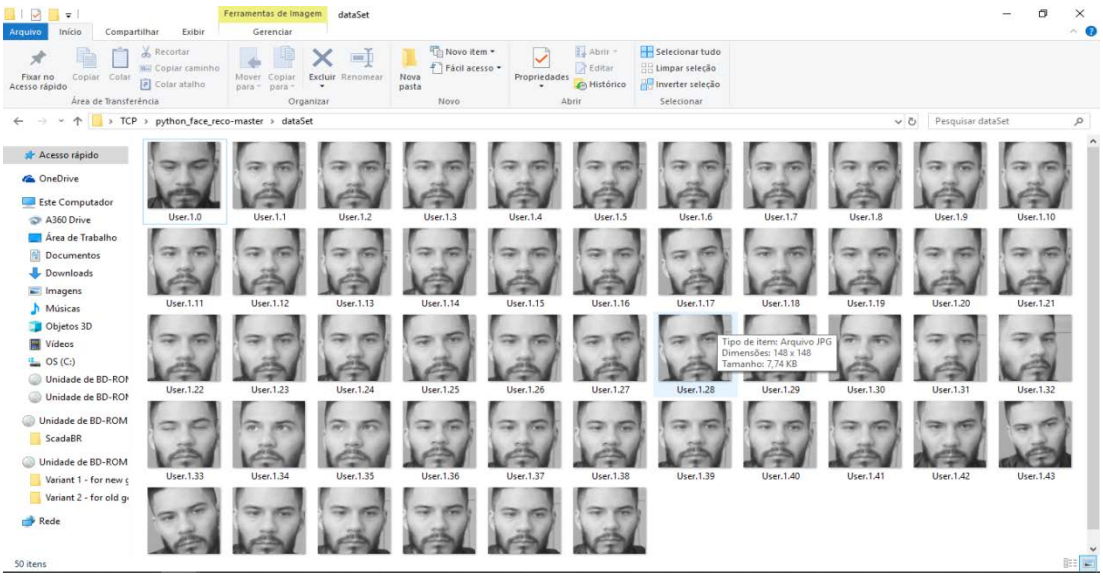


Verifica-se no diretório, no documento de texto “Names” se o nome este cadastrado conforme solicitado pelo algoritmo.

Figura 8 – documento de texto atualizado após a inserção do nome.

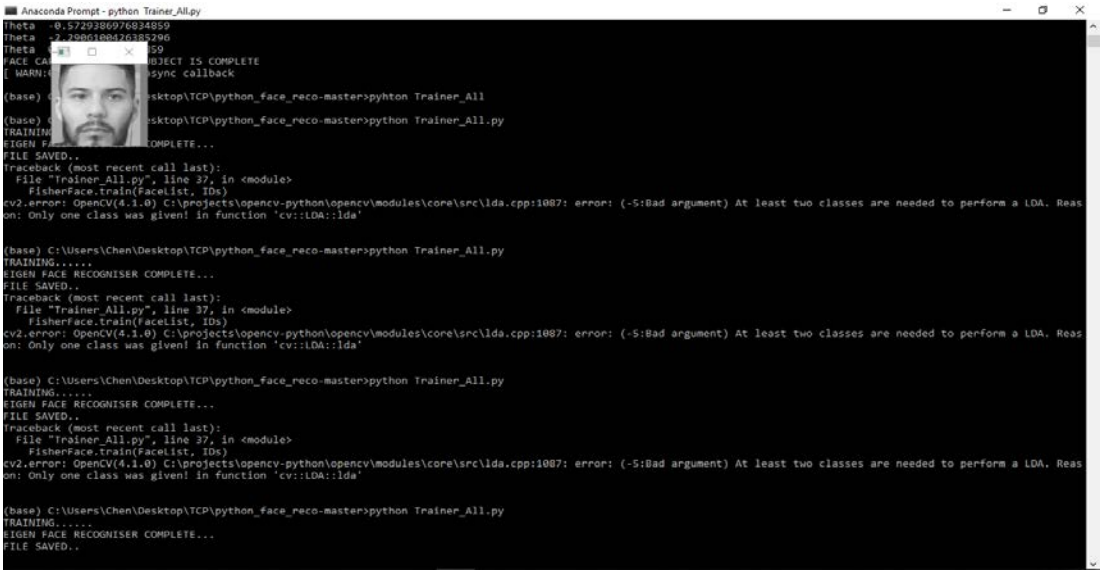
Também se verifica na pasta “dataSet” se foi inserido as 50 imagens tomadas após executar o comando “python Face_Capture _With_Rotate.py”.

Figura 9 – Imagens tomadas para o reconhecimento facial na pasta “dataSet”



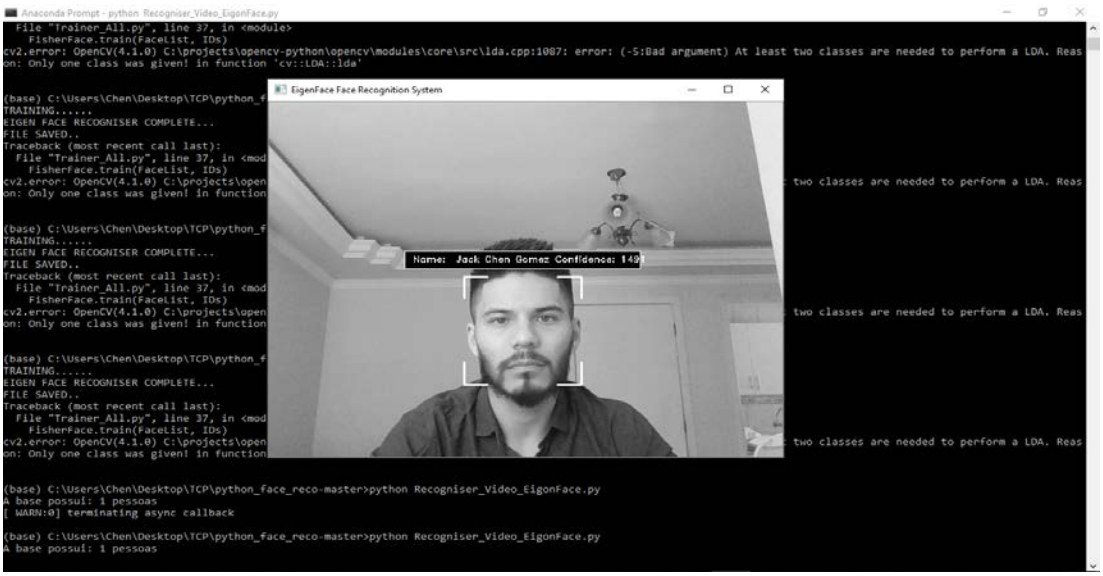
Depois de executar estes comando, está pronto para salvar o a fotos e nomes em arquivos de “xml” para o treinamento. Efetua-se o treinamento da base “dataSet” e “Names” com o comando “python Trainer_All.py”.

Figura 10 – execução do algoritmo de treinamento



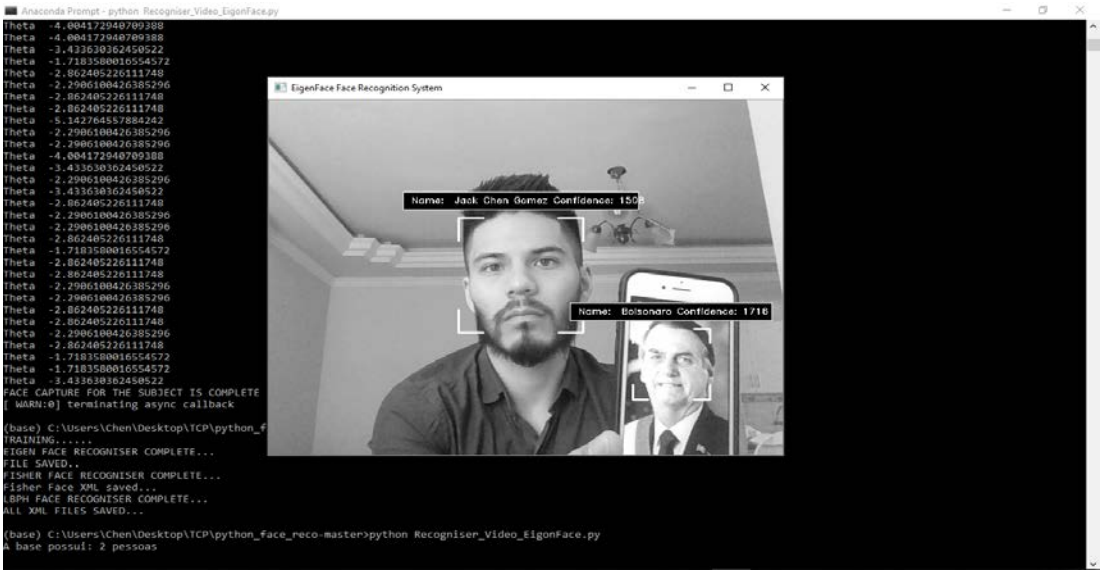
Logo após de inserção da imagem na base e o treinamento executa-se o comando “python Recogniser_Video_EigonFace.py” o qual ligará a webcam para o reconhecimento da imagem.

Figura 11 – Reconhecimento facial da pessoa cadastrada no banco de dados



No caso sé há necessidade de processamento de mais de uma imagem na webcam em conjunto á essa possibilidade, abaixo demostra-se a implementação de uma imagem de outra pessoa e o algoritmo fazendo a leitura de dois rostos ao mesmo tempo.

Figura 12 – Reconhecimento facial das pessoas cadastradas no banco de dados



Com base nos resultados dos quatro testes realizados, pode-se concluir que o sistema de reconhecimento, em geral, tem uma baixa taxa de erros, e na maioria das condições testadas, apresentou resultados satisfatórios, reconhecendo indivíduos mesmo com pouca iluminação e baixa qualidade da imagem (smartphones).

6. CONCLUSÃO

Este trabalho apresentou um estudo sobre sistemas de reconhecimento facial automatizados, cobrindo desde a parte teórica até os passos necessários para o desenvolvimento de um programa computacional baseado nos métodos estudados. Foi verificado que a implementação de um sistema de reconhecimento facial exige algoritmos robustos, capazes de lidar com variações de pose do indivíduo, ruídos na imagem de entrada e diferentes configurações de iluminação ambiente. O sistema desenvolvido consistiu de um módulo de detecção facial, outro de extração de características e outro de reconhecimento. O módulo de detecção facial foi baseado no algoritmo de Viola-Jones e apresentou resultado satisfatório. O módulo de extração de característica, baseado na análise linear discriminante, mostrou-se eficiente ao calcular o subespaço de características, levando em media 1,0 segundo para analisar 100 imagens. Das 100 imagens-teste, o módulo de reconhecimento, identificou todas as imagens corretamente. Além disso, mostrou-se tolerante a ruído e

conseguiu identificar indivíduos mesmo com pouca iluminação e baixa qualidade da imagem (smartphone).

7. REFERÊNCIAS

[1] NASCIMENTO, V. **Implementação de um sistema de identificação facial utilizando Linux Embarcado. 2015.** 1f. Monografia (Especialização em Engenharia de Computação com ênfase em eletrônica) - Escola de Engenharia de São Carlos, Universidade São Paulo.

[2] Embedded System Design: **A Unified Hardware/Software Introduction**–Frank Vahid and Tony Givargis, John Wiley and Sons, 2002

[4] M. BLEDSOE. **The model method in facial recognition. Technical Report** PRI 15, Panoramic Research Inc., Palo Alto, CA, 1964.

[5] M. Kelly. **Visual identification of people by computer. Technical Report** AI 130,Stanford, CA, 1970.

[6] BRAGA L. F. Z. **Sistemas de Reconhecimento Facial. 2013**. 23f. Monografia (Especialização em Engenharia de Elétrica com ênfase em eletrônica) São Carlos, Universidade de São Paulo.

[7] W. Zhao; R. Chellappa. **Face Processing: Advanced Modeling and Methods.** Academic Press, Inc., Orlando, FL, USA, 2005

[8] USA TODAY FBI uses facial-recognition technology on DMV photos Usa Today. 2009. Disponível em: <http://usatoday30.usatoday.com/tech/news/2009-10-13-fbi-dmv- facial-recognition_N.html>. Acesso em: 01 mar. 2019.

[9] THE MAKING OF PYTHON. **Artima Developer.** 2003. Disponível em: <https://www.artima.com/intv/python.html>. Acesso em: 01 mar. 2019.

[10] Reis, Claiton – **"Sistemas Operacionais para Sistemas Embarcados", Tutorial**, Editora: EDUFBA, BRASIL, 2004.

[11] Steve Heath, **Embedded Systems Design** , Newnes, 2002 ISBN 0-080-47756-9

[12] Shibu **Intro To Embedded Systems** 1E . Tata McGraw-Hill Education ISBN 0-070-14589-X

[13] E. BARROS E S. CAVALCANTI **Introdução a os Sistemas Embarcados.** 2019. Disponível: <http://www.cin.ufpe.br/~vba/periodos/8th/s.e/aulas/STP%20%20Intro%20Sist-%20Embarcados.pdf>. Acesso em: 10 mar. 2019.

[14] INTEL.COM, '**Intel® Embedded Microcontrollers**'. 2019. Disponível: <http://www.i-ntel.com/design/embcontrol/>. Acesso em: 10 mar. 2019.

[15] ARM.COM, '**ARM Processor Architecture - ARM**'. 2019. Disponível: <http://www.arm.com/producs/processors/instruction-set-architectures/index.php>. Acesso em: 10 mar. 2019.

[16] TANENBAUM, Andrew S. (2006). **Operating systems: design and implementation**. USA: Prentice Hall. 6 páginas

[17] COMER, Douglas (2012). **Operating system design: the XINU approach**, Linksys version. New York: CRC Press. 2 páginas

[18] M. Mitchell, J. Oldham and A. Samuel, **Advanced Linux programming**. Indianapolis, Ind.: New Riders Pub., 2001.

[19] Eric S. Raymond (29 de dezembro de 2003). **Software proprietário deve ser distinguido de software comercial**. Acesso em: 15 mar. 2019.

[20] GNU/LINUX, '**FAQ**'. 2019. Disponível: <https://www.gnu.org/gnu/gnu-linux-faq.pt-br.html>. Acesso em: 10 mar. 2019.

[21] STALLMAN, RICHARD. 'Linux e o Sistema GNU'. 2019. Disponível: <https://www.gnu.org/>. Acesso em: 30 mai. 2019.

[22] KASHEM, Mohammod Abul et al. **Face Recognition System Based on Principal Component Analysis (PCA) with Back Propagation Neural Networks (BPNN)**. Canadian Journal On Image Processing And Computer Vision, Canadá, n.p.36-45, 04 abr. 2011. 2
.
[23] GOUVEIA, Wellington da Rocha; PAIVA, Maria Stela Veludo de. **Detecção de Faces Humanas em Imagens Coloridas Utilizando Redes Neurais Artificiais**. In: VWORKSHOP DE VISÃO COMPUTACIONA, 5., 2009, São Paulo.

[24] TYAGI, Rahul Kumar; SINGH, Neha; CHAUDHARY, Piyush. **Analysis of Facial Ggesture Recognition Using Eigen Faces. International Journal Of Computer Science And Communication**, Neemrana, India., n. , p.465-468, 2 jul. 2011.

[25] YANG, Ming-hsuan; KRIEGMAN, David J.; AHUJA, Narenda. **Detecting Faces in Images: A survey. Pattern Analysis And Machine Intelligence**, IEEE Transactions On, n. , p.34-58, 07 ago. 2002.

[26] TURK, M.A.; PENTLAND, A.P**. Face Recognition Using Eigenfaces. Proceedings of IternationalConference on Pattern Recognition**. , n. , p.586-591, 06 jun. 1992.

[27] HAYKIN, Simon. **Redes Neurais: Princípio e Prática. 2. ed**. Porto Alegre: Bookman, 1999. 896 p.

[28] VIOLA, P; JONES, M. **Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition (CVPR)** v. 1, p. I--511--I--518 , 2001.0-7695-1272- 0.

[29] VIOLA; M. JONES. **Rapid object detection using a boosted cascade of simple features. In Proc. Of CVPR**, 2001

[30] G. Yang; T. S. Huang, "**Human Face Detection in Complex Background**". Pattern Recognition, vol. 27, no. 1, 1994.

[31] T.F.Cootes, C.J.Taylor, A.Lanitis, **Active Shape Models: Evaluation of a Multi-Resolution Method for Improving Image Search**, in Proc. British Machine Vision Conference, 1994

APÊNDICE A – Códigos em *Python* (abrir em *Sublime Text* 3)

Face_Capture_With_Rotate

```
import cv2
# Importando o opencv
import numpy as np
# Importando o numero python
import NameFind
 # Importando função NameFind
WHITE = [255, 255, 255]
#   importando o classificador Haar cascades para a detecção de faces
face_cascade = cv2.CascadeClassifier('Haar/haarcascade_frontalcatface.xml')
# Classificador "Face de frente" Haar Cascade
eye_cascade = cv2.CascadeClassifier('Haar/haarcascade_eye.xml')
# Classificador "olho" Haar Cascade
ID = NameFind.AddName()
Count = 0
cap = cv2.VideoCapture(0)
# Captura camera
while Count < 50:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
 # Convertendo a camera para grayScale
```

```
   if np.average(gray) > 110:
 # Testando o brilho da imagem
       faces = face_cascade.detectMultiScale(gray, 1.3, 5)
# Detectando as faces nas posições
       for (x, y, w, h) in faces:  # Lugar das frames X, Y  WIDTH, HEIGHT
          FaceImage = gray[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
# Copiando a face e colando
          Img = (NameFind.DetectEyes(FaceImage))
          cv2.putText(gray, "FACE DETECTED", (x+int((w/2)), y-5),
cv2.FONT_HERSHEY_DUPLEX, .4, WHITE)
          if Img is not None:
             frame = Img
# Mostrando a face dectectada:
             frame = gray[y: y+h, x: x+w]
          cv2.imwrite("dataSet/User." + str(ID) + "." + str(Count) + ".jpg", frame)
          cv2.waitKey(300)
          cv2.imshow("CAPTURED PHOTO", frame)
# Mostrando a imagem capturada
          Count = Count + 1
   cv2.imshow('Face Recognition System Capture Faces', gray)
# Mostrando video
   if cv2.waitKey(1) & 0xFF == ord('q'):
      break
print (Face capturada com sucesso)
cap.release()
cv2.destroyAllWindows()
```

Trainer_All

```
import os
# Importando o pacote OS
import cv2
```

```
# Impostando a livraria opencv
import numpy as np
# Importando livraria Numpy
from PIL import Image
# Importando libraria de imagem
EigenFace = cv2.face.EigenFaceRecognizer_create(15)
# Criando EIGEN FACE RECOGNISER
FisherFace = cv2.face.FisherFaceRecognizer_create(2)
# Criando FISHER FACE RECOGNISER
LBPHFace = cv2.face.LBPHFaceRecognizer_create(1, 1, 7,7)
# Criando LBPH FACE RECOGNISER
path = 'dataSet'
# Pacotes de fotos
def getImageWithID (path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    FaceList = []
    IDs = []
    for imagePath in imagePaths:
        faceImage = Image.open(imagePath).convert('L')
# Abrir imagem e converter para cinza
        faceImage = faceImage.resize((110,110))
# Reajustar tamanho da imagem para que o EIGEN consiga treinar
faceNP = np.array(faceImage, 'uint8')
# Convertendo a imagem para uma matriz Numpy
ID = int(os.path.split(imagePath)[-1].split('.')[1])
# Recriando novamente o ID da matriz
        FaceList.append(faceNP)
# Anexar a matriz Numpy à lista
        IDs.append(ID)
# Anexe o ID à lista de IDs
        cv2.imshow('Training Set', faceNP)
# Mostrar as imagens da lista
```

```
        cv2.waitKey(1)
    return np.array(IDs), FaceList
# Os IDs são convertidos em uma matriz Numpy
IDs, FaceList = getImageWithID(path)
print('TRAINING......')
EigenFace.train(FaceList, IDs)
# O recongniser é treinado usando as imagens
print('EIGEN FACE RECOGNISER COMPLETE...')
EigenFace.save('Recogniser/trainingDataEigan.xml')
print('FILE SAVED..')
FisherFace.train(FaceList, IDs)
print('FISHER FACE RECOGNISER COMPLETE...')
FisherFace.save('Recogniser/trainingDataFisher.xml')
print('Fisher Face XML saved... ')
LBPHFace.train(FaceList, IDs)
print('LBPH FACE RECOGNISER COMPLETE...')
LBPHFace.save('Recogniser/trainingDataLBPH.xml')
print ('ALL XML FILES SAVED...')
cv2.destroyAllWindows()
```

Recogniser_Video_EigonFace

```
import cv2
# Importando o opencv
import NameFind
# Importando o NameFind
face_cascade = cv2.CascadeClassifier('Haar/haarcascade_frontalcatface.xml')
# Classificador "frente da face" Haar Cascade
eye_cascade = cv2.CascadeClassifier('Haar/haarcascade_eye.xml')
# Classificador "olho" Haar Cascade
recognise = cv2.face.EigenFaceRecognizer_create(15,4000)
 # Criando EIGEN FACE RECOGNISER
```

```
recognise.read("Recogniser/trainingDataEigan.xml")
# Carregando os dados do treinamento
cap = cv2.VideoCapture(0)
# Objeto da camera
cap = cv2.VideoCapture('TestVid.wmv')
# Objeto do video
ID = 0
while True:
   ret, img = cap.read()
# Leia o objeto da câmera
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Converta a câmera em cinza
   faces = face_cascade.detectMultiScale(gray, 1.3, 5)
# Detecte os rostos e armazene as posições
   for (x, y, w, h) in faces:
# Lugar das Frames X, Y  WIDTH, HEIGHT
      gray_face = cv2.resize((gray[y: y+h, x: x+w]), (110, 110))
# O rosto é isolado e cortado
      eyes = eye_cascade.detectMultiScale(gray_face, 1.3, 5)
      for (ex, ey, ew, eh) in eyes:
         ID, conf = recognise.predict(gray_face)
# Determine o ID da foto
         NAME = NameFind.ID2Name(ID, conf)
         NameFind.DispID(x, y, w, h, NAME, gray)
   cv2.imshow('EigenFace Face Recognition System', gray)
# Mostrar imagem
   if cv2.waitKey(1) & 0xFF == ord('q'):
# Para sair aperta a telca “Q”
      break
cap.release()
cv2.destroyAllWindows()
```