



**UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

GUILHERME FONTANIVA

**MONITORAMENTO DE AMBIENTES E CONTROLE DE
FREQUÊNCIA ACADÊMICA ATRAVÉS DE VISÃO
COMPUTACIONAL**

**CHAPECÓ
2014**

GUILHERME FONTANIVA

**MONITORAMENTO DE AMBIENTES E CONTROLE DE
FREQUÊNCIA ACADÊMICA ATRAVÉS DE VISÃO
COMPUTACIONAL**

Trabalho de conclusão de curso de graduação
apresentado como requisito para obtenção do
grau de Bacharel em Ciência da Computação da
Universidade Federal da Fronteira Sul.

Orientador: Prof. Me. Fernando Bevilacqua

CHAPECÓ
2014

Fontaniva, Guilherme

Monitoramento de ambientes e controle de frequência acadêmica
através de visão computacional / por Guilherme Fontaniva. – 2014.

54 f.: il.

Orientador: Fernando Bevilacqua

Trabalho de conclusão de curso (graduação) - Universidade Federal da Fronteira Sul, Curso de Ciência da Computação, Chapecó, SC, 2014.

1. Detecção. 2. Reconhecimento. 3. Visão computacional. 4. Câmera comum. 5. Computação gráfica. 6. Processamento de Imagens. 7. Não intrusivo. I. Bevilacqua, Fernando, orient. II. Universidade Federal da Fronteira SulIII. Título.

© 2014

Todos os direitos autorais reservados a Guilherme Fontaniva. A reprodução de partes ou do

todo deste trabalho só poderá ser feita mediante a citação da fonte.

E-mail: guilhermefontaniva@gmail.com

GUILHERME FONTANIVA

**MONITORAMENTO DE AMBIENTES E CONTROLE DE
FREQUÊNCIA ACADÊMICA ATRAVÉS DE VISÃO COMPUTACIONAL**

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Me. Fernando Bevilacqua

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 10, 12, 14

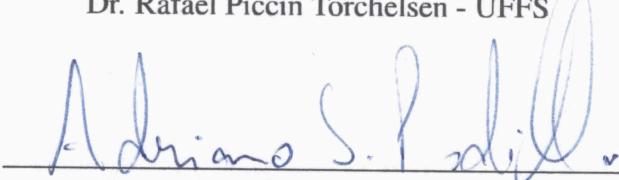
BANCA EXAMINADORA:



Me. Fernando Bevilacqua - UFFS



Dr. Rafael Piccin Torchelsen - UFFS



Me. Adriano Sanick Padilha - UFFS

AGRADECIMENTOS

Agradeço primeiramente a minha família que sempre apoiou-me e me deu forças em toda essa trajetória.

Agradeço a todos meus amigos que apoiaram-me e participaram de alguma forma em meu trabalho e em todo esse período que passou.

Agradeço ao meu orientador Prof. Me. Fernando Bevilacqua, que sempre esteve presente para mostrar-me qual caminho seguir para a realização deste trabalho.

Agradeço aos professores Dr. Rafael Piccin Torchelsen e Me. Adriano Sanick Padilha por participarem da banca deste trabalho, e também a todo o corpo docente que possibilitou a realização dessa etapa.

RESUMO

Com a evolução da velocidade dos computadores e da melhoria das técnicas para algoritmos, uma quebra do paradigma intrusivo para o não intrusivo torna-se viável. Um usuário não precisa necessariamente trocar seu meio para usar a computação, ela pode ser inserida nos locais, tornado-se algo mais natural e não intrusivo. O reconhecimento de padrões a partir de imagens digitais é uma tarefa que essa evolução permite realizar. Esse reconhecimento é estudado pela área da computação conhecida como *visão computacional*. O reconhecimento facial é um dos reconhecimentos de padrões que foi realizado. Esse possibilita um monitoramento de ambientes, realizando assim tarefas como o controle de frequência acadêmica. O objetivo desse trabalho é a construção de uma aplicação que realize o controle de frequência automatizado para ambientes monitorados com uma câmera comum. O trabalho apresenta a construção de uma aplicação que utiliza uma câmera conectada a um computador para capturar uma sequência de imagens digitais. A partir dessas faz-se a detecção de rostos, e para todos os rostos na imagem aplica-se um algoritmo de reconhecimento facial. O resultado desse reconhecimento é salvo em um *log*, que realiza o controle automatizado de frequência de um determinado ambiente, conforme os padrões estipulados para esse. O trabalho apresenta comparações e testes da aplicação, mostrando o tempo de reconhecimento e o impacto que esse sofre com diversos fatores, como iluminação heterogênea durante o treinamento do reconhecedor.

Palavras-chave: Detecção. Reconhecimento. Visão computacional. Câmera comum. Computação gráfica. Processamento de Imagens. Não intrusivo.

ABSTRACT

The evolution of computer vision and the increase of available processing power enabled a paradigm shift that can change the use of computers from an intrusive to a non-intrusive approach. The idea of a non-intrusive system is to allow users to interact with computers without the use of peripherals or direct contact with a machine. Computer vision can be used to achieve this paradigm shift. This work presents the implementation and validation of a software able to monitor an environment, acting as an attendance checker for use in the classroom. The application uses a camera attached to a computer capturing a sequence of digital images. By analyzing those images, the system is able to detect and recognize human faces captured by the camera, registering the date and the time for each person found. A set of tests were designed and performed to measure the recognition time and how it was affected by different factors as the environment illumination or the amount of training images used to add a new person to the application database.

Keywords: Detection, Recognition, Computer vision, Simple camera, Graphic computer, Image processing, Non-intrusive.

LISTA DE FIGURAS

Figura 2.1 – Exemplo de processamento de imagem digital com visão computacional para segmentação da imagem [19].	17
Figura 2.2 – Partes de um algoritmo de visão computacional, a extração de características está implícita junto com o pré-processamento [1].	18
Figura 2.3 – Faces detectadas em imagens através de métodos de visão computacional [14].	19
Figura 2.4 – Tipos possíveis de características Haar-like e um exemplo de aplicação numa imagem [26].	20
Figura 2.5 – Etapas do algoritmo HOG [4].	21
Figura 2.6 – Imagem binária e as bordas da esquerda, cima e direita para gerar uma silhueta final da face [22].	22
Figura 2.7 – Detecção de face feita por detecção de cor com filtro de falsos positivos [13].	24
Figura 2.8 – Passos para o reconhecimento facial usado pela biblioteca <i>OpenBR</i> [12].	27
Figura 3.1 – Diagrama da aplicação, com o fluxo de módulos e sub-módulos.	29
Figura 3.2 – A esquerda é mostrado um exemplo de detecção de face com a câmera. A direita o resultado de uma imagem processada gerada pelo treinamento.	30
Figura 3.3 – Telas da aplicação do cadastro de aula.	30
Figura 3.4 – Fluxo de cadastro de aula com exemplo de arquivo gerado de saída.	31
Figura 3.5 – Exemplo de arquivo de aula, de nome teste e com três alunos cadastrados. . .	31
Figura 3.6 – Fluxo de aquisição de imagens do treinamento com exemplo de saída.	32
Figura 3.7 – Exemplo de cadastro de uma nova pessoa	32
Figura 3.8 – Exemplo de imagem de cadastro de uma nova pessoa de ID 1 e com as posições dos olhos (72, 79) e (139, 85)	33
Figura 3.9 – Fluxo de alinhamento de imagens do treinamento com exemplo de saída. . .	33
Figura 3.10 – Exemplo de arquivo de treinamento para a fase de alinhamento das imagens de treinamento.	34
Figura 3.11 – Exemplo de arquivo gerado após o alinhamento, descrevendo todas as imagens alinhadas para o treinamento.	34
Figura 3.12 – Diagrama da aplicação com destaque no módulo de reconhecimento.	35
Figura 3.13 – Arquivo de exemplo de entrada do treinamento contendo duas colunas, o caminho da imagem de treino e seu ID associado.	36
Figura 3.14 – Fluxo do treinamento para a aplicação.	37
Figura 3.15 – Exemplo de imagens para o treinamento com várias iluminações e expressões. .	37
Figura 3.16 – Diagrama da aplicação com destaque no reconhecimento facial	38
Figura 3.17 – Fluxo do reconhecimento aplicação.	38
Figura 3.18 – Resultado de quando não é reconhecida a pessoa na imagem, retornando o ID : -1.	39
Figura 3.19 – Exemplo de retorno do reconhecimento de faces.	40
Figura 3.20 – Fluxo do reconhecimento aplicação com exemplo de <i>log</i> de saída.	40
Figura 3.21 – Exemplo de arquivo de log para o controle de frequência, onde apenas um aluno, <i>Guilherme</i> , foi reconhecido, no instante 30/10/2014 08 : 41 : 30. . .	41
Figura 4.1 – Exemplo de reconhecimento de pessoa não cadastrada, mostrando a variação da margem de erro para o reconhecedor.	43
Figura 4.2 – Pessoas utilizadas no treinamento para os testes da aplicação de reconhecimento facial.	44

Figura 4.3 – Três pessoas foram encontradas na imagem, mas nenhuma delas estava previamente cadastrada.	45
Figura 4.4 – A esquerda: exemplo de imagem com reconhecimento e não reconhecimento. A direita: exemplo de imagem com múltiplo reconhecimento.	45
Figura 4.5 – Exemplo de imagem com faces não detectadas, estando elas de perfil e inclinadas.	46
Figura 4.6 – Exemplo de imagem com quatro faces, onde uma foi reconhecida errada, utilizando o treinamento padrão dos teste.	47
Figura 4.7 – Pessoas utilizadas para o teste de acurácia.	48

LISTA DE TABELAS

Tabela 2.1 – Sub Divisão dos quatro grupos de detecção facial [28].	19
Tabela 4.1 – Resultados da aplicação de reconhecimento facial com variação do número de imagens do treinamento.	48

LISTA DE ABREVIATURAS E SIGLAS

HOG	<i>Histogram of Oriented Gradients</i>
AdaBoost	<i>Adaptative Booster</i>
SVM	<i>Support Vector Machines</i>
OpenBR	<i>Open Biometric Recognition</i>
OpenCV	<i>Open Source Computer Vision Library</i>
PCA	<i>Principal Components Analysis</i>
RGB	<i>Red Green Blue</i>
RNA	Redes Neurais Artificiais
UFFS	Universidade Federal da Fronteira Sul
LDA	<i>Linear Discriminant Analysis</i>

SUMÁRIO

1 INTRODUÇÃO	13
2 REVISÃO BIBLIOGRÁFICA	16
2.1 Visão computacional	16
2.2 Detecção de pessoas	17
2.2.1 Detecção facial	18
2.2.2 Características Haar-Like em cascata	20
2.2.3 HOG.....	21
2.2.4 Segmentação por bordas	22
2.2.5 Detecção por cor da pele	23
2.2.6 EigenFaces	23
2.2.7 Adaptative Booster	24
2.2.8 Support Vector Machines	25
2.2.9 FisherFaces	25
2.3 Bibliotecas de visão computacional	26
2.3.1 <i>OpenCV</i>	26
2.3.2 <i>OpenBR</i>	26
3 IMPLEMENTAÇÃO	28
3.1 Cadastro de treinamentos	29
3.1.1 Aulas	30
3.1.2 Pessoas	31
3.2 Reconhecimento	35
3.2.1 Treinamento.....	36
3.2.2 Reconhecimento facial.....	38
3.2.3 Geração de <i>logs</i>	40
4 RESULTADOS	42
4.1 Reconhecimento	42
4.1.1 Treinamentos.....	47
5 CONCLUSÃO	50
5.1 Trabalhos futuros	50
REFERÊNCIAS	52

1 INTRODUÇÃO

A evolução da tecnologia possibilitou um aumento da velocidade de processamento dos computadores abrindo um leque de caminhos que não eram possíveis até pouco tempo, gerando capacidade para realização de diversas novas tarefas. Uma dessas tarefas é permitir a máquina “enxergar” o mundo externo através de imagens, realizando uma integração com os meios onde está inserida. Uma das áreas da computação impactada diretamente por esse avanço no processamento foi a *visão computacional*.

A visão computacional é uma das áreas da computação cujo enfoque está em fazer máquinas interagirem com o mundo através de dados multi-dimensionais, como imagens, em um nível similar às pessoas. Pode-se realizar através dela a extração de características e modelos 3D a partir de imagens, por exemplo na técnica Haar-Like, citada na Seção 2.2.2, que retira características de imagens através de comparação de cores.

A Visão Computacional mescla-se ao cotidiano, possibilitando a realização de tarefas por comandos gestuais [10]. Essa abordagem de fazer as máquinas interagirem com pessoas de uma maneira natural, em seu meio, sem a necessidade de procurar por um computador é definido por *computação não intrusiva* [16]. A *computação intrusiva*, por sua vez, é definida como uma maneira de comunicação *homem-máquina*, na qual o homem acessa a máquina e realiza comandos e tarefas através de periféricos, dirigindo-se ao computador fisicamente. A *computação não intrusiva* quebra esse conceito, e insere a computação no meio, para que seja usada sem necessidade de acesso direto à máquina.

A computação não intrusiva vem ganhando espaço integrando-se com a sociedade. A visão computacional é uma forma de conexão entre o computador e o meio externo, traduzindo dados de entrada para uma linguagem computacional. Essa área da computação possui um papel significativo na troca do paradigma intrusivo ao não intrusivo. Ferramentas que ajudem a máquina a identificar e detectar pessoas são primordiais para que essa troca de paradigmas ocorra.

Detectar e identificar um usuário através de uma câmera comum seria uma vantagem, tratando-se de custos de *hardware*. Com o avanço das tecnologias observa-se a redução nos custos de equipamentos de *hardware*. A fácil disponibilidade de novos *hardware*s com maior poder de processamento, aliados a novos dispositivos de entrada de dados, oportuniza a realização de novas abordagens sobre métodos computacionais já consolidados. Essas oportunidades

tendem a gerar soluções que facilitem ou aprimorem a experiência do usuário. Um exemplo de facilidade é o reconhecimento de pessoas em imagens digitais.

O reconhecimento de pessoas através de imagens digitais possibilita a quebra do paradigma intrusivo entre *homem-máquina*, transformando o processo em uma tarefa não intrusiva. Com essa passagem, a computação torna-se algo mais natural no cotidiano das pessoas, sendo que elas não precisam estar sentadas na frente de uma máquina comunicando-se apenas por meios de periféricos como teclado [22]. Uma pessoa poderia realizar várias ações através de gestos passando comandos para uma máquina no meio em que está sem a necessidade de ir até um computador.

O primeiro passo para que possa haver uma quebra desse paradigma é a detecção de pessoas e seu reconhecimento. Isso não é realmente necessário em alguns contextos, mas sem essa funcionalidade perde-se o controle de quem está acessando o sistema, e se aquela pessoa possui permissão de realizar determinada tarefa.

Um dos passos para a construção desses sistemas é a identificação de um usuário, relacionando-o com suas permissões e papéis. Para isso utiliza-se autenticação do usuário, baseados em uso de *login e senha*, limitando seu acesso apenas a certas partes do sistema as quais foram permitidas a ele.

Os *sistemas de login* fazem a destinação dos papéis e permissões de cada um. Apesar de possuir uma boa segurança esse sistema é sujeito a falha, visto que um intruso pode acessar o *software* usando as credenciais de um usuário válido. Para resolver esse problema existem detecções biométricas como a leitura ocular e de digitais, um meio eficaz no qual apenas um usuário pode acessar as suas permissões.

Um sistema baseado em visão computacional possui autenticação diferente dos sistemas tradicionais baseados em *login e senha*. Para o reconhecimento de um usuário de forma não intrusiva faz-se necessário a identificação pela face ou por um *token* específico como um cartão. Isso já é realizado com o *kinect* [20], mas os custos de *hardware* são altos comparados a uma câmera comum. Com a identificação facial do usuário ocorre a transformação do *login* em algo não intrusivo, mais natural, que reconhece o usuário sem a necessidade que ele use um periférico [22].

A utilização de uma solução baseada em visão computacional abre um leque de possibilidades futuras referentes à autenticação e interação de usuários com sistemas. Um exemplo é a possibilidade de sistemas de monitoramento e segurança de locais inteligentes, no qual seja

disparado um alerta caso alguém que não esteja cadastrado com permissões de acesso visite o local. Outra aplicação seria uma *smart house*, fazendo com que portas abram-se, aparelhos liguem e desliguem e diversas ações que pessoas poderão desempenhar com simples gestos. Possibilita-se também a realização de um controle de frequência acadêmico em uma sala de aula, automatizando todo o processo.

Sistemas atuais de controle de frequência são manuais e dependem da colaboração de todos os envolvidos no processo. A metodologia consome tempo de aula e, além disso, os alunos não chegam necessariamente ao mesmo tempo, havendo necessidade de atualização do estado da presença ao decorrer da aula. Essa abordagem manual, que exige uma constante atualização durante ou depois do período de aula, consome tempo. Esse tempo poderia ser empregado para a aula em sí, não para o controle de frequência, sendo essa automatizada.

Em meio a esse contexto, o desenvolvimento de uma solução baseada em visão computacional, utilizando *hardware* de baixo custo, apresenta-se como uma contribuição. Fundamentando-se na utilização de *hardware* de baixo custo, a partir desta instrumentação não intrusiva tem-se uma possível automatização de frequência acadêmica, fazendo com que a computação mescle-se com o cotidiano, tornando-a mais natural.

O objetivo deste trabalho é a implementação de uma aplicação experimental que seja capaz de detectar que há uma, ou mais, pessoas em um ambiente e identifica-las através de uma câmera comum e, com essa identificação, realizar o controle de frequência acadêmico. Esse reconhecimento é feito através das faces encontradas na imagem digital. Essa aplicação demonstra uma possibilidade de prática de um uso não intrusivo para a computação, permitindo a execução de tarefas sem periféricos.

2 REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta as técnicas e algoritmos relacionados com o presente trabalho. Para a realização do trabalho utilizando *visão computacional*, utilizou-se de vários conceitos, que são descritos a seguir, sendo um deles a própria *visão computacional*. Outro tópico de destaque da seção é a *detecção de pessoas*. Nela são descritos métodos de detecção de pessoas em imagens digitais. Algumas técnicas computacionais são descritas para a realização da detecção facial e reconhecimento através de *visão computacional*.

2.1 Visão computacional

A visão pode ser considerada um importante sentido, visto que é usada para reconhecer o meio e tudo que está ao seu redor. As imagens, portanto, desempenham grande papel na percepção das pessoas. As tarefas executadas por máquinas com imagens são diversas, por exemplo o *zoom* em uma foto. Essa tarefa de *zoom*, por exemplo, é um tipo de processamento de imagem digital. Algoritmos de processamento de imagens digitais são caracterizados por terem como entrada ou saída uma imagem, com alteração ou coleta de dados.

O processamento de imagens digitais deriva de *processamento de sinais*, que carregam consigo informações. Essas informações estão associadas a medidas, como uma imagem tem *pixels* que possuem cor [5]. Pode-se separar em dois ramos da computação o processamento de imagens digitais, sendo eles a computação gráfica, a parte responsável por gerar imagens de modelos [5], e a visão computacional, a parte responsável pelo processamento de dados e extração de modelos a partir de imagens.

A visão computacional é a área responsável pela visão de uma máquina a partir de imagens, sejam elas bi-dimensional ou tri-dimensional. Sua grande motivação é fazer com que as máquinas enxerguem como humanos para auxiliar em tarefas cotidianas [23]. A Figura 2.1 apresenta uma ideia de visão computacional para reconhecer objetos em uma imagem digital. À direita da ilustração encontram-se as imagens originais e na esquerda a imagem com as bordas, da moto e da pessoa destacadas, exemplificando um processamento de imagem digital por visão computacional.

Um algoritmo de visão computacional pode ser dividido em cinco partes formando uma rotina de execução, como pode ser observado na Figura 2.2, sendo elas:

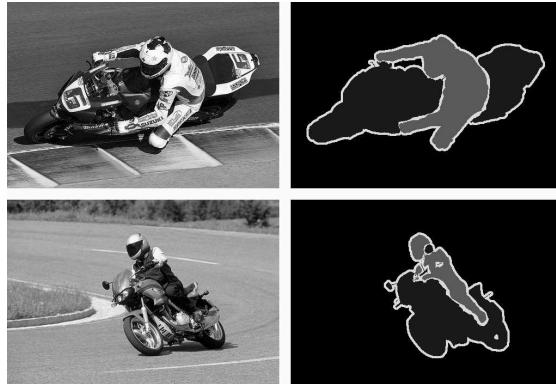


Figura 2.1: Exemplo de processamento de imagem digital com visão computacional para segmentação da imagem [19].

- **Aquisição da imagem** - Obtem-se uma matriz de *pixels* contendo informações para formar uma imagem digital capturada por uma câmera, compondo os dados de entrada do sistema.
- **Pré-processamento** - Nesta fase são aplicados métodos e técnicas de processamento de imagem, como a transformação para imagem binária e a normalização de dados, para transformá-la e possibilitar, ou melhorar a eficiência, da extração de características para reconhecimento de objetos.
- **Extração de características** - São encontradas as características dos objetos da imagem, que serão selecionados para a utilização no sistema. Um exemplo dessa técnica é a extração de características usada no método Haar-Like visto na Seção 2.2.2.
- **Segmentação** – Extraem-se objetos em sub-imagens menores, com apenas o objeto de interesse para futuro processamento.
- **Pós-processamento** – Em aplicações de reconhecimento facial, esse passo engloba ações para reagir conforme os objetos encontrados, como fazer o controle automático de frequência reconhecendo as pessoas.

2.2 Detecção de pessoas

Um sistema de visão computacional definido para o reconhecimento de pessoas em imagens digitais deve ter como objetivo a extração de características de pessoas ou de faces. A parte que o difere de detecção de outros objetos é o fato de uma mesma pessoa variar suas características. Exemplos dessa diferenciação são as roupas, óculos, o cabelo, a barba, a expressão

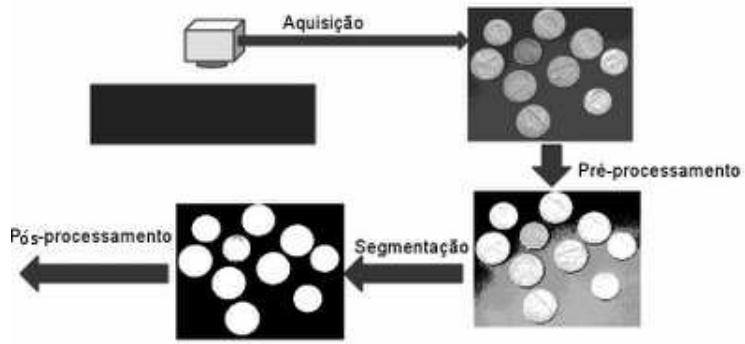


Figura 2.2: Partes de um algoritmo de visão computacional, a extração de características está implícita junto com o pré-processamento [1].

da pessoa (alegre, triste, zangada, etc.), e diversos outros fatores que tornam-se complicadores para o reconhecimento. As características que podem ser usadas na detecção de pessoas em uma imagem, por meio de visão computacional, são a face humana, a cabeça, todo o corpo, incluindo as pernas, e a pele [14].

2.2.1 Detecção facial

O passo inicial, e o mais importante, para a detecção e reconhecimento de pessoas através de imagens digitais geradas por câmeras comuns é a detecção da face [18]. Embora o reconhecimento facial seja uma tarefa simples e rotineira para uma pessoa, em um meio computacional, ela torna-se complicada devido à face não ter características fixas. A face muda continuamente devido a diversos fatores, tanto da cena, como iluminação, e também da pessoa, como expressão facial [18]. O resultado de uma detecção facial é ilustrado na Figura 2.3, mostrando cinco imagens distintas, onde nelas são reconhecidas faces, porém tem-se o exemplo da não detecção de uma face existente, sendo ela um falso negativo, devido a diversos fatores da imagem.

O reconhecimento facial era realizado com técnicas compostas de *redes neurais*, e *heurísticas* [2]. Essas técnicas eram pouco eficientes e muito sujeitas a erros [2]. O limitado poder de processamento não permitia esse reconhecimento em tempo real. A evolução da área, permitiu o desenvolvimento de técnicas mais eficientes e poderosas computacionalmente [18]. Essas técnicas melhoraram outras já existentes com novas técnicas [26], permitindo a computadores, por exemplo, detectar múltiplas pessoas em tempo real [27, 9, 14].

Algoritmos de detecção de faces em imagens podem ser classificados em quatro grupos [28]:

- **Baseado no conhecimento** : Fundamenta-se em detectar faces a partir de regras e características tendo como base o conhecimento humano.
- **Características invariantes** : Apresentam métodos para detecção facial independente de iluminação, pose, ponto de vista, etc.
- **Modelos correspondentes** : Baseiam-se em comparações entre a imagem de entrada e as características inicialmente inseridas no programa.
- **Baseado em aspecto** : Assimilam-se aos modelos correspondentes, mas esse tem um aprendizado com novas imagens.

A Tabela 2.1 mostra um conjunto de técnicas distribuídas nesses quatro grupos.

Baseados em conhecimento	Baseado em regras de multi-resolução
Características invariantes	Características faciais
	Textura
	Cor da pele
	Múltiplas características
Modelos correspondentes	Modelos pré-definidos
	Modelos mutáveis
Baseado em aspecto	Eigenface
	Baseado em distribuição
	Rede Neural
	SVM
	Classificadores nativos
	HMM
	Aproximação por teoria de informações

Tabela 2.1: Sub Divisão dos quatro grupos de detecção facial [28].



Figura 2.3: Faces detectadas em imagens através de métodos de visão computacional [14].

As seções seguintes apresentam técnicas consolidadas para a detecção de faces em imagens.

2.2.2 Características Haar-Like em cascata

Características Haar-Like derivam da técnica matemática *transformada de Haar*, uma transformada de sinais. As características Haar-Like são uma forma de extração de características de objetos através de imagens digitais [21].

Essas características são regiões retangulares divididas em duas partes, positivas e negativas. São aplicadas várias vezes na imagem conforme ilustra a Figura 2.4. Para a detecção calcula-se a soma das intensidades dos *pixels* da região positiva, subtraindo-se a soma da intensidade dos *pixels* da região negativa. Esse valor final é comparado com uma faixa aceitável para ser classificado como característica ou não.

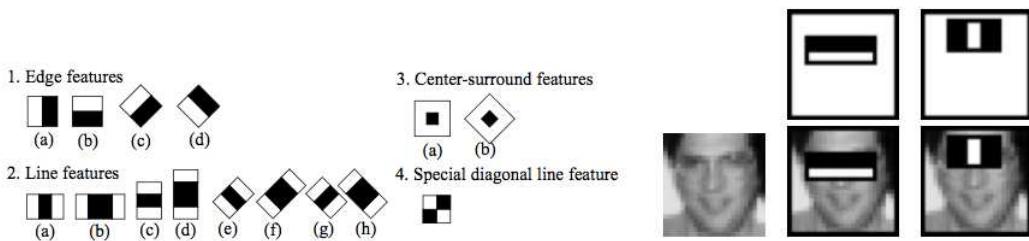


Figura 2.4: Tipos possíveis de características Haar-like e um exemplo de aplicação numa imagem [26].

O Haar-Like em cascata baseia-se em três características, que tornam ele significativamente melhor que o Haar-Like [26, 27]:

- De uma técnica denominada *integral image*, uma facilitadora na detecção de características Haar-like. Essa técnica não utiliza a intensidade do *pixel*, isso resulta um melhor desempenho. Para o uso da imagem somam-se todos os *pixels*, da área em branco e subtraem os *pixels* da área em preto, visto na Figura 2.4. Se essa soma estiver fora de uma certa faixa esperada, é porque há uma característica Haar-like.
- O refinamento com o uso do AdaBoost, abordado na seção 2.2.7, para resolver problemas em sua fase de treino.
- Uma técnica combinando características Haar-like para criar um filtro entre elas, que decide quais delas serão úteis, pois uma imagem pode ter inúmeras características Haar-like, e somente as mais significativas são necessárias.

Os principais problemas desta técnica são a dificuldade em detecção de característica Haar-Like com a cor de fundo próximo a cor do objeto e as mudanças muito intensas na iluminação [27].

2.2.3 HOG

O método *Histogram of Oriented Gradients*, ou HOG, é uma técnica baseada em gradientes para gerar histogramas orientados para uma imagem e classificada de acordo com os dados de treinamento do objeto definido. Para que sejam definidos esses histogramas e o objeto seja reconhecido, o algoritmo deve seguir uma série de passos [4], conforme exibido pela Figura 2.5.

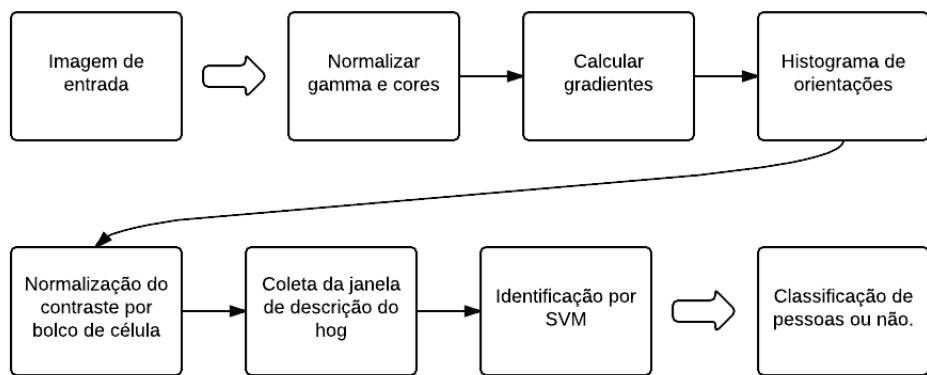


Figura 2.5: Etapas do algoritmo HOG [4]

As etapas do algoritmo são:

- **Normalizar gamma e cores** é um passo opcional, feito para um melhor desempenho a partir de esquemas de cores RGB e Lab.
- **Calcular gradientes** usando a técnica de *Sobel 1D*, que é basicamente o cálculo da norma de cada ponto da imagem, fazendo-se o cálculo dos gradientes da imagem. Para imagens coloridas o cálculo é feito para cada canal de cor.
- **Histograma de orientações** são construídos a partir dos gradientes calculados na fase anterior, sendo eles orientados (0° a 360°) ou não orientados (0° a 180°). São descritos como células, que podem ser circulares ou retangulares. Os intervalos dos histogramas variam para cada aplicação, para detecção de pessoas o ideal é 20° .

- **Normalização do contraste por bloco de células**, realizado para que dois blocos de células não sejam sobrepostos na imagem.
- **Coleta da janela de descrição do HOG** é o passo que coleta a janela da imagem na qual deve ser realizada a detecção.
- **Identificação por SVM** (descrito na Seção 2.2.8) é feita na fase final para identificar o bloco de células otimizado entre todos da janela.

O HOG é mais lento que o *Haar-Like em cascata*, porém a relação de erros quanto a falsos negativos e falsos positivos é menor [29]. HOG também pode ser utilizado em cascata e com tamanho de blocos de células não fixos para gerar um melhor desempenho na técnica [29].

2.2.4 Segmentação por bordas

A segmentação por bordas é uma técnica que consiste primeiramente em transformar uma imagem em tons de cinza. Para isso calcula-se o gradiente dos *pixels* a partir de seus vizinhos. Esse gradiente é a base da transformação para uma imagem binária (apenas com *pixels* brancos e pretos). Depois de obter a imagem binária, faz-se uma varredura por todos os lados definindo-se as bordas para gerar uma silhueta da face [6].

Como mostrado na Figura 2.6, inicialmente é realizada a transformação da imagem para imagem binária, após isso é feito, a partir de todos os lados, uma varredura de um extremo a outro até encontrar um valor significativo. Após esse passo todo o resto da linha ou coluna é preenchido com o mesmo *pixel*. Para finalizar agregam-se todas as imagens resultando em uma borda como demonstrado na imagem mais à direita.



Figura 2.6: Imagem binária e as bordas da esquerda, cima e direita para gerar uma silhueta final da face [22].

2.2.5 Detecção por cor da pele

A detecção de uma face por cor da pele em imagens digitais tem como seu principal obstáculo as alterações de cores de pele, que geram uma gama de cores a serem analisada [25]. Para essa técnica funcionar, são analisados todos os *pixels* da imagem. Há diversas formas de realizar essa tarefa, um exemplo do resultado final é exibido na Figura 2.7, onde a imagem 1 representa a imagem original, a próxima mostra a detecção de cor de pele realizada separando apenas os *pixels* com coloração de pele, no passo 3 são analisadas as possíveis faces da imagem, e no passo final são descartados os falsos negativos gerados no passo 3.

O método conhecido como *região da pele explicitamente definido*, constrói regras específicas para a classificação de pele, por exemplo, usando RGB: se $R > 95$ e $G > 40$ e $B > 20$ e $(\max(R, G, B) - \min(R, G, B) > 15)$ e $|R - G| > 15$ e $R > G$ e $R > B$, então é pele.

A *modelagem de distribuição não paramétrica* baseia-se em treinamento de dados através de modelos de peles até que ele seja capaz de reconhecer padrões. Inclui-se nesse grupo algoritmos como redes de *Kohonen* e métodos de tabelas de referência [25].

A *modelagem de distribuição paramétrica* baseia-se na curvatura gaussiana para seu funcionamento. Requer muito espaço de armazenamento e depende exclusivamente de imagens de treinamento [25].

Modelos de distribuição de pele dinâmicos é uma técnica mais flexível, mas com um limitador de que o fundo da imagem deve ser distinto da face. Conforme [25], pode-se utilizar diversas técnicas, alguma delas são *adaptação distribuição de Gauss*, *histogramas dinâmicos* e *maximização de expectativa lineares*. Estão fora do escopo desse trabalho.

2.2.6 EigenFaces

A técnica *EigenFaces* é baseada em um conceito de que faces não encontram-se aleatoriamente em um espaço relativamente grande, podendo ser representadas por espaços menores, definidos espaços faciais [24]. EigenFaces são características de imagens, encontradas pelas transformada de Karhunen-Loeve. Apenas as EigenFaces que tem maior peso relacionado com os dados de treinamento são selecionadas.

A técnica é composta por cinco passos:

- Carregar dados de treinamento e calcular as eigenfaces que definem o espaço facial.



Figura 2.7: Detecção de face feita por detecção de cor com filtro de falsos positivos [13].

- Calcular os pesos baseando-se nas EigenFaces de entrada, quando uma nova provável face é encontrada.
- Checar se a nova provável face é uma face e encontra-se próxima de algum espaço facial.
- Classificar, se for face, como padrão conhecido ou desconhecido.
- Calcular o peso das suas características padrões e adiciona-la as conhecidas, se for classificado como desconhecido e aparecer diversas vezes (Passo opcional de treinamento).

2.2.7 Adaptive Booster

Para a inteligência artificial, mais especificamente em aprendizado de máquina, é considerado um *boosting*, algoritmos que melhoram o desempenho de um aprendizado supervisionado [8]. Esses são algoritmos que partem da dúvida “Podemos transformar vários indivíduos de baixa aprendizagem em um indivíduo de alta aprendizagem?”. Introduzido em 1995 por Yoav Freund e Robert Schapire [7], o AdaBoost, ou Adaptive Booster, que é um algoritmo de *boosting* aprimora o algoritmo de aprendizado supervisionado.

Seu funcionamento inicia com um analisador fraco, indivíduo de baixo aprendizado, iterando-o com outros analisadores fracos, mesclando características até que haja um analisador forte [7].

2.2.8 Support Vector Machines

A técnica *Support Vector Machines*, ou SVM, é um método de aprendizado supervisionado associado com algoritmo de aprendizado, que analisa dados para o reconhecimento de padrões, usado para classificação e análise de regressão [3]. A SVM, a partir dos dados de entrada, usa seus dados de treinamento para decidir de qual classe o dado de entrada encontra-se. A técnica é geralmente classificada como operador binário, por fazer comparação entre duas classes para gerar a saída. Esse método é comparável, em desempenho, e muitas vezes superior a métodos de inteligencia artificial como RNAs [3].

2.2.9 FisherFaces

A técnica de FisherFaces é uma técnica de aprendizado de máquina baseada em lógica difusa, na qual é pouco afetada pela iluminação ou ambiente de seus treinamentos [15]. Essa técnica utiliza da variação de sub espaços vetoriais para deferir as características do conjunto de treino através dos componentes principais de analise (*Principal Components Analysis - PCA*) [11]. Apos a análise dos PCAs da imagem também é realizada uma analise do LDA (*Linear Discriminant Analysis*), que são os responsáveis pela redução da imagem em um sub-espáço facial ótimo.

PCA é um método matemático que usa transformação ortogonal para converter um grupo de variáveis correlacionados a um grupo de componentes linearmente descorrelacionadas, descritos como componentes principais. LDA é uma técnica que busca a otimização de um sub-espáço facial encontrado em uma imagem. Essa técnica é a responsável pela busca de um raio que descreve a variância das pessoas pré treinadas pelo método. Além dessa busca a técnica de LDA também busca maximizar as características entre todas as pessoas cadastradas, fazendo uma maior diferenciação das pessoas reconhecidas.

Esse método baseia-se no método de EigenFaces 2.2.6, sendo ele composto do mesmo núcleo mas com a adição dos LDAs, achando combinações lineares para aumentar a variância de seus dados melhorando o seu desempenho. Seu funcionamento divide-se em duas fases, a fase de extração dos PCAs e a otimização através do método de LDA.

2.3 Bibliotecas de visão computacional

2.3.1 *OpenCV*

A *Open Source Computer Vision Library* ou *OpenCV* é uma biblioteca aberta para o desenvolvimento de aplicações em visão computacional, que abrange em seu escopo também aprendizado de máquina, estrutura de dados e álgebra Linear.

A *OpenCV* disponibiliza diversas ferramentas de visão computacional, desde operações como filtro de ruídos até análise de movimentos, reconhecimento de padrões e reconstrução em 3D. A *OpenCV* foi desenvolvida com o intuito de tornar a visão computacional mais acessível [17].

Pode-se dividi-la em cinco grupos: processamento de imagens, análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões e calibração de câmera e reconstrução 3D [17].

2.3.2 *OpenBR*

A *Open Biometric Recognition*, ou *OpenBR*, é uma biblioteca aberta de visão computacional que oferece ferramentas para o desenvolvimento de novas interfaces para incorporação com algoritmos biométricos [12].

A *OpenBR* tem no seu escopo várias técnicas, já consolidadas, de visão computacional e processamento de imagens como o reconhecimento facial, mostrado na Figura 2.8, além de outros recursos para o reconhecimento biométrico. Essa biblioteca teve suas pesquisas mais voltadas a reconhecimento de rostos através de visão computacional do que outros reconhecimentos biométricos, e destaca-se por isso.

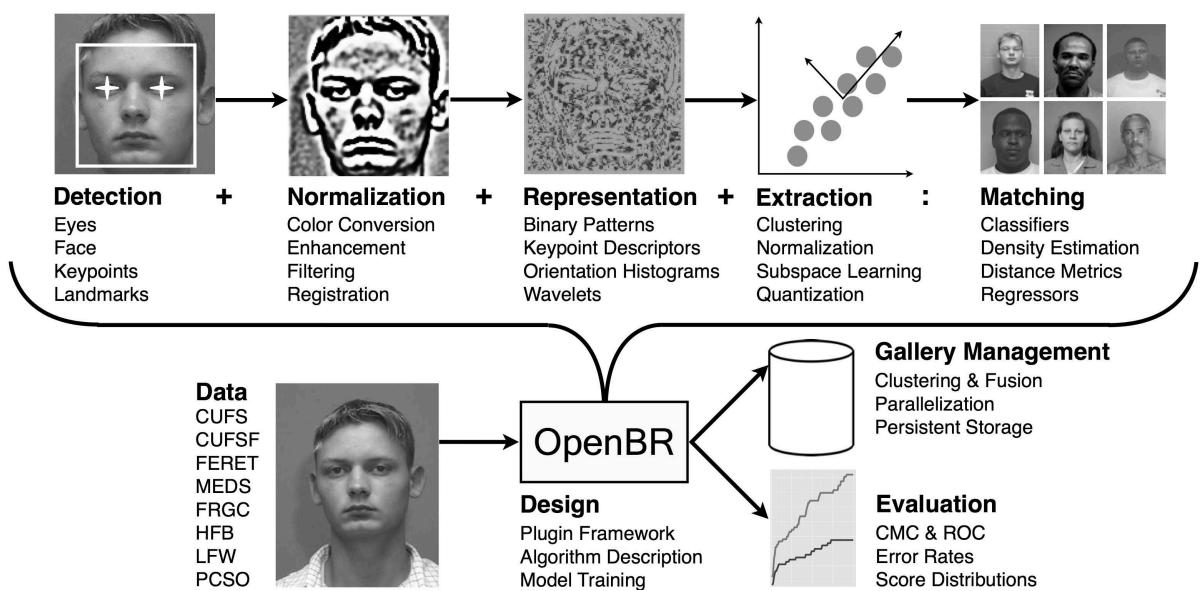


Figura 2.8: Passos para o reconhecimento facial usado pela biblioteca *OpenBR* [12].

3 IMPLEMENTAÇÃO

A implementação do projeto foi dividida em dois módulos principais, cadastro de treinamentos e reconhecimento, mostrados na Figura 3.1. O **cadastro de treinamentos** é responsável pela entrada de informações, que serão mantidas pelo programa, as aulas e as pessoas. As pessoas serão cadastradas para o reconhecimento e as aulas para a geração do controle de frequência. Esses cadastros são salvos para possibilitar o funcionamento do módulo de **reconhecimento**. O **reconhecimento** é o responsável pela realização do reconhecimento das pessoas previamente cadastradas pelo módulo de **cadastro de treinamentos** e também pelo controle de frequência.

O módulo de **cadastro de treinamentos** subdivide-se em dois, aulas e pessoas. Para aulas salva-se um arquivo de texto contendo a informação de cada aula, como o nome e os participantes. Essas informações das aulas são responsáveis pela geração do controle de frequência. Para pessoas são salvadas imagens de uma pessoa, essas associadas a um ID, que é o identificador. Essas imagens das pessoas são salvadas em tons de cinza e são pre-processadas para seguirem um padrão.

O módulo de **reconhecimento** segue três passos, treinamento, reconhecimento facial e geração de *logs*. A inicialização define a aula utilizada pelo controle de frequência realizado pela aplicação. O treino obtém todas as imagens de cada pessoa salva pelo módulo de **cadastro de treinamentos**, com o identificador associado, e treina a aplicação para realizar o reconhecimento.

O **reconhecimento facial** trabalha com os dados do treinamento anterior, coletando imagens da câmera e analisando as características semelhantes com as imagens dos treinamentos. E por fim a geração de *logs* cria um arquivo com as pessoas que foram reconhecidas pela fase anterior e que estavam cadastradas na aula inicializada. Esse *log* possui o nome da aula e sua respectiva data. Esse *log* também contém o nome do aluno cadastrado na aula e a data e hora de sua chegada.

As seções desse capítulo estão divididas conforme cada módulo e sub-módulo do diagrama mostrado pela Figura 3.1, explicando-se detalhadamente cada parte de sua implementação.

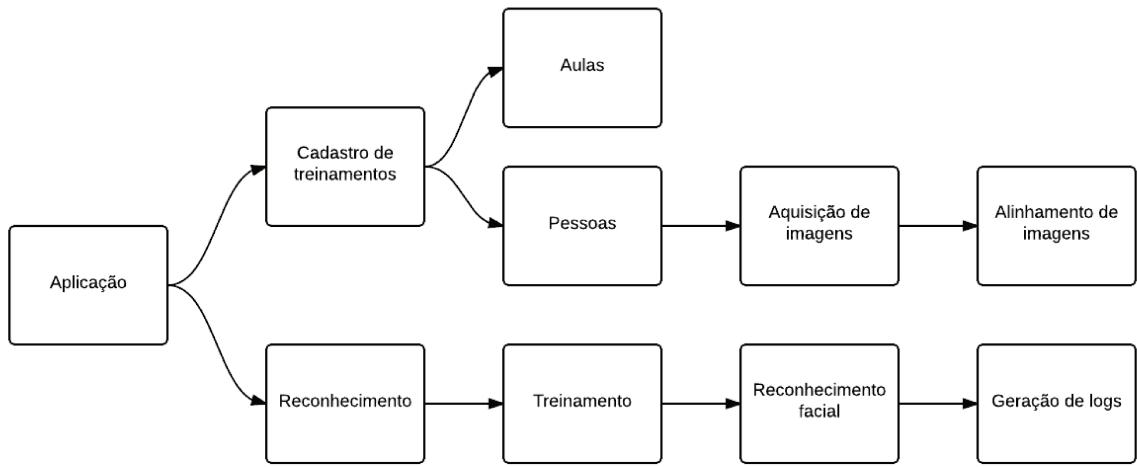


Figura 3.1: Diagrama da aplicação, com o fluxo de módulos e sub-módulos.

3.1 Cadastro de treinamentos

O módulo de cadastros de treino realiza a inserção de dados necessários para a execução do programa, sendo eles as pessoas e as aulas cadastradas, possibilitando o controle de frequência automatizado. O funcionamento do módulo de cadastros de treinos é composto de duas etapas: o treino de pessoas para o reconhecimento e o cadastro de aulas.

O cadastro de aulas foi desenvolvido com a inserção de um novo nome da aula e de seus alunos participantes com seu respectivos identificadores (ID). Um arquivo texto com múltiplas linhas contendo o ID e o nome de cada aluno cadastrado é gerado a partir dessas inserções. Esse arquivo é guardado dentro de uma pasta própria para as aulas no projeto, que será usado na geração de *logs*.

O cadastro de pessoas realiza a inserção de imagens de uma pessoa, sendo essa associada a um ID, para serem utilizadas no treinamento do reconhecimento facial, descrito na Seção 3.2.1. Para obter essas imagens faz-se a captura a partir da câmera. Na imagem encontra-se o rosto da pessoa que deve ser cadastrada, recortado-o e salvando-o em uma nova imagem em tons de cinza. Após o armazenamento das images essas são processadas para que fiquem todas com o mesmo tamanho e rotação, exemplificado na Figura 3.2.

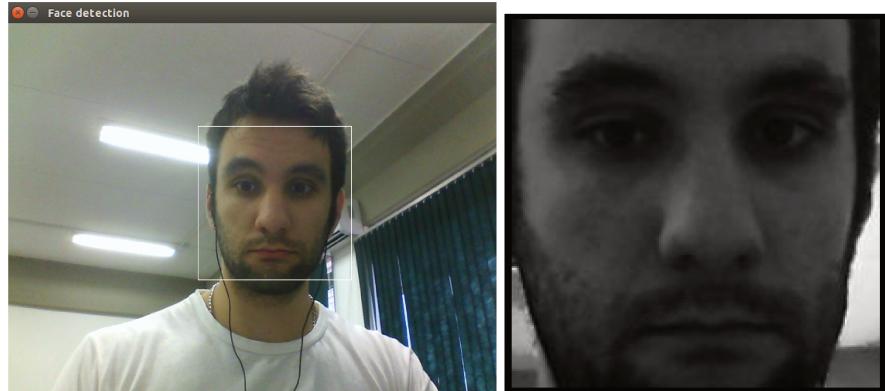


Figura 3.2: A esquerda é mostrado um exemplo de detecção de face com a câmera. A direita o resultado de uma imagem processada gerada pelo treinamento.

3.1.1 Aulas

O cadastro de aulas tem a função de cadastrar novas aulas para possibilitar o controle de frequência a partir dos dados das aulas cadastradas. Nesse cadastro tem-se o nome da aula e os alunos participantes dela. Sua utilização, em nível de usuário, funciona conforme as telas mostradas na Figura 3.3. Inicialmente espera-se um nome para a aula, que será equivalente ao nome do arquivo gerado para a aula. Em seguida o ID e o nome da pessoa cadastrada na aula, repetindo o processo até que todas as pessoas da aula sejam cadastradas, gerando por fim um arquivo da aula como mostrado pela Figura 3.5.

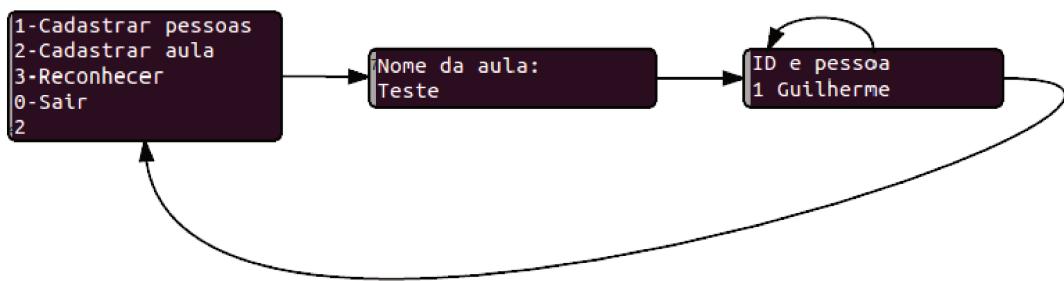


Figura 3.3: Telas da aplicação do cadastro de aula.

O sub-módulo de cadastro de aulas, demonstrado pela Figura 3.4, segue um fluxo de três partes. A primeira parte, nome da aula, é responsável pelo controle de qual aula está sendo cadastrada, que será utilizada para o controle de frequência automatizada. A partir do seu nome que será gerado o *log* com as frequências, explicado na Seção 3.2. A segunda parte, cadastro de aluno, insere nos dados fixos do sistema quais alunos estão incluídos na aula, sendo esses

os que serão esperados na aula e são registrados na frequência caso sejam reconhecidos. A terceira parte, geração de arquivo, é responsável por acessar os dados das fases anteriores e salvá-los, para que o programa os utilize posteriormente, sendo esse arquivo a lista de chamada. Esse arquivo gerado é a base de registros das pessoas reconhecidas, só as pessoas presentes no arquivo devem ser registradas pelo controle de frequência da aula que está sendo controlada pela aplicação.

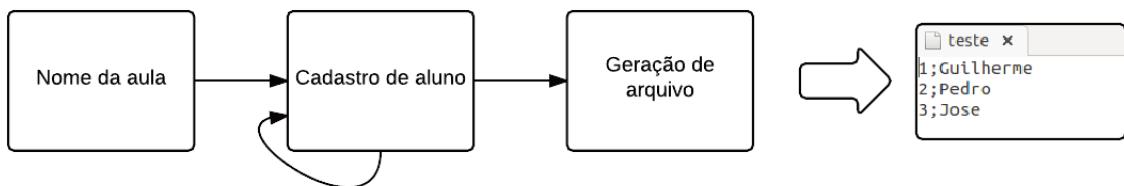


Figura 3.4: Fluxo de cadastro de aula com exemplo de arquivo gerado de saída.

A estrutura do arquivo resultante, de nome igual à aula, possui várias linhas, cada uma com o ID de uma pessoa e seu nome, exemplificado na Figura 3.5. A figura mostra o nome do arquivo *teste*, também nome da aula, e na sua primeira linha pode-se ver o ID 1 associado ao aluno *Guilherme*.

Esse arquivo de texto, com todas as aulas disponíveis, é salvo pelo programa em uma pasta na máquina onde está sendo executado. Essas aulas disponíveis são usadas posteriormente pelo programa, na fase de reconhecimento, Seção 3.2. Esse reconhecimento usa o arquivo de aulas para gerar um *log* com as pessoas presentes na aula. A aula atual que é utilizada no reconhecimento é definida pelo usuário.

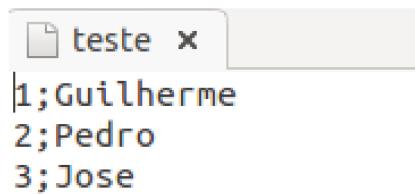


Figura 3.5: Exemplo de arquivo de aula, de nome teste e com três alunos cadastrados.

3.1.2 Pessoas

O cadastro de pessoa é o sub-módulo responsável por inserir imagens de novas pessoas, para que o programa possa realizar o reconhecimento. Essas fotos cadastradas devem ser associadas a um ID, sendo esse o identificador da pessoa. Uma pessoa deve possuir várias fotos

para o reconhecimento, visto que poucas fotos podem causar erros e uma redução na taxa de acerto do reconhecimento.

A implementação do cadastro de pessoas possuí um sub-módulo com três partes, sendo cada uma dessas dependente da anterior. Para a inicialização do cadastro de pessoas, o programa precisa de um ID, para associar a nova pessoa que será cadastrada. Esse deve ser o mesmo associado à pessoa no cadastro de aula. Após a inserção do ID é feito a comunicação com a câmera, abrindo a tela de captura para cadastro de pessoas.

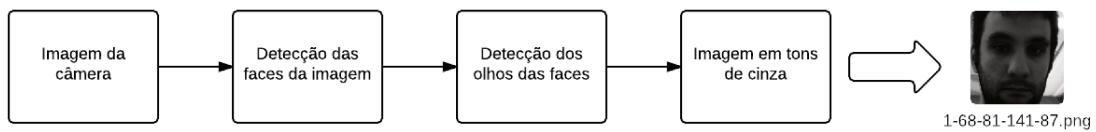


Figura 3.6: Fluxo de aquisição de imagens do treinamento com exemplo de saída.

A primeira parte consiste na aquisição de imagens da câmera, cujo fluxo é mostrado pela Figura 3.6. A inicialização obtém imagens da câmera, enviada para o programa. A imagem é processada utilizando-se a técnica Haar-Like em cascata, Seção 2.2.2, recebendo o treinamento padrão da *OpenCV*. Com essa técnica encontram-se os rostos da imagem recebida, mostrada na Figura 3.7.



Figura 3.7: Exemplo de cadastro de uma nova pessoa

A partir dos rostos encontrados são detectados os olhos da pessoa reconhecida, utilizando-se a mesma técnica. Em seguida o programa gera uma imagem em tons de cinza,

normalizando-a para utilização do algoritmo de reconhecimento. O resultado é mostrado na Figura 3.8. O nome da nova imagem gerada possui informações do ID da pessoa e as posições dos olhos, exemplificada pela Figura 3.8.

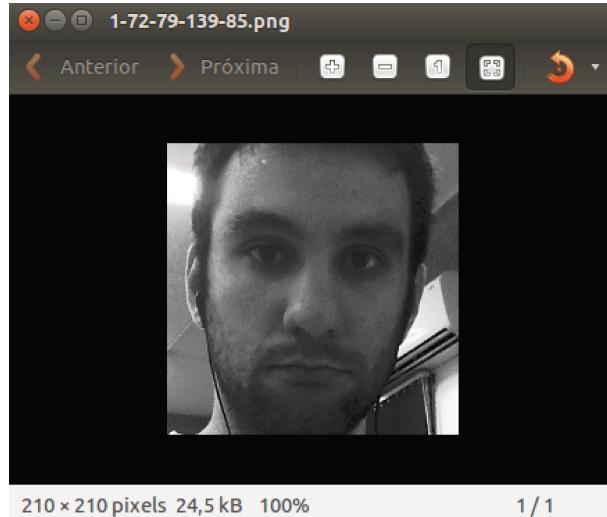


Figura 3.8: Exemplo de imagem de cadastro de uma nova pessoa de ID 1 e com as posições dos olhos (72, 79) e (139, 85)

A segunda fase do módulo de cadastro de treinamentos de pessoas, alinhamento de imagens, tem como entrada a imagem gerada pela fase anterior do processo, e tem como saída uma imagem alinhada, demonstrada através da Figura 3.9. Alinhamento é o processo que padroniza as imagens para que todas fiquem em formato semelhante.

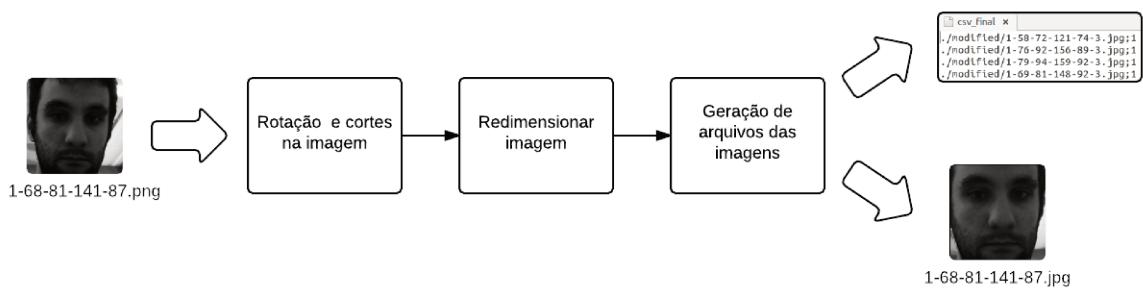
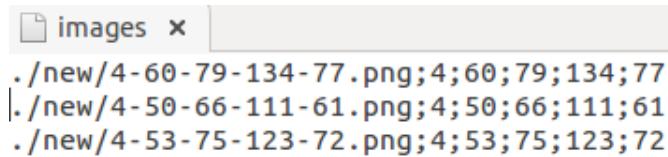


Figura 3.9: Fluxo de alinhamento de imagens do treinamento com exemplo de saída.

No alinhamento a imagem passa por três processos. Na rotação, o primeiro a ajustar a imagem, alinha-se os olhos com a posição horizontal semelhante. O segundo, corte da imagem, remove as informações mais irrelevantes, como o fundo da imagem, tentando otimizar a imagem para o reconhecimento facial. Por fim tem-se o redimensionamento da imagem, garantindo que as imagens do treinamento fiquem com uma matriz que possua o mesmo tamanho.

Para realizar essas tarefas utiliza-se a posição dos olhos, que está salva no nome da imagem. Através dessas é possível fazer a rotação e o corte da imagem, deixando-a mais limpa e clara para a utilização no treinamento do reconhecimento.

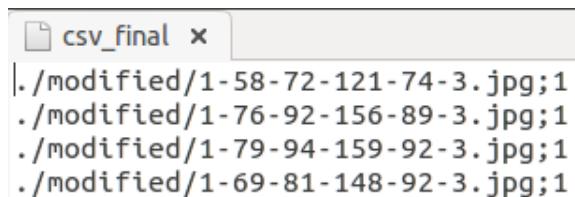
O sub-módulo apresentado utiliza um arquivo de texto como entrada contendo o caminho das imagens, o ID e as posições dos olhos, informações necessárias para o processo de alinhamento das imagens de treinamento. A Figura 3.10 mostra um exemplo, em que a primeira linha, por exemplo, tem no conteúdo da primeira coluna o caminho para a imagem `./new/4 - 60 - 79 - 134 - 77.png`, o ID 4 na segunda coluna, e os pontos dos olhos na imagem (60, 79) e (134, 77) nas quatro colunas seguintes.



```
images
./new/4-60-79-134-77.png;4;60;79;134;77
./new/4-50-66-111-61.png;4;50;66;111;61
./new/4-53-75-123-72.png;4;53;75;123;72
```

Figura 3.10: Exemplo de arquivo de treinamento para a fase de alinhamento das imagens de treinamento.

Além da imagem alinhada o sub-módulo também gera um arquivo que descreve todas as imagens alinhadas resultantes desse e dos treinamentos passados. Esse arquivo tem em sua estrutura o caminho da imagem e também o ID da pessoa associada a imagem, exemplificado na Figura 3.11. Usando a primeira linha da imagem de exemplo do arquivo, observa-se o caminho na primeira coluna, `./modified/1 - 58 - 72 - 121 - 74 - 3.jpg` e o ID 1, na segunda coluna. Esse arquivo é utilizado para o reconhecimento facial.



```
csv_final
./modified/1-58-72-121-74-3.jpg;1
./modified/1-76-92-156-89-3.jpg;1
./modified/1-79-94-159-92-3.jpg;1
./modified/1-69-81-148-92-3.jpg;1
```

Figura 3.11: Exemplo de arquivo gerado após o alinhamento, descrevendo todas as imagens alinhadas para o treinamento.

Para um treinamento eficiente são necessárias várias imagens com diferentes posições, angulações, expressões e iluminações, para que haja uma melhor consistência nos dados das fotos e do reconhecimento. O ideal é fazer a utilização de imagens capturadas com a mesma câmera, ou com uma de resolução similar para não haver dispersão de dados obtidas em imagens de diferentes resoluções.

3.2 Reconhecimento

O reconhecimento segue um fluxo de três partes principais, como mostrado na Figura 3.12. As partes são: treinamento, que utiliza elementos obtidos pelos cadastros de pessoas e treina o algoritmo de aprendizado de máquina; reconhecimento facial, utilizando a técnica de FisherFaces (Seção 2.2.9) para reconhecer as faces encontradas nas imagens obtidas pela câmera; por fim a geração de *logs* para os que estavam presentes e cadastrados na aula, responsável por realizar o controle de frequência automatizado.

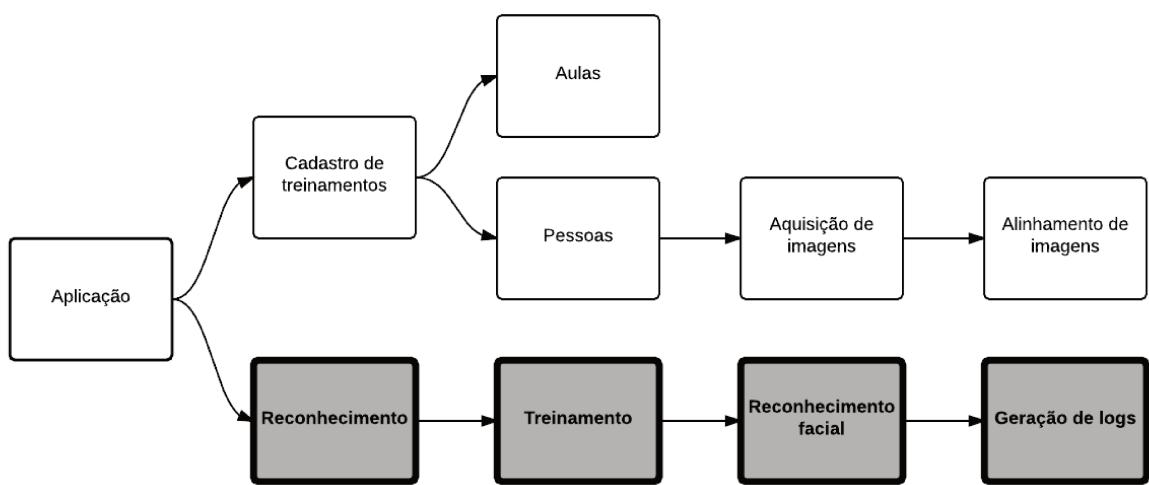


Figura 3.12: Diagrama da aplicação com destaque no módulo de reconhecimento.

O módulo de reconhecimento utiliza as informações de aulas e pessoas, cadastradas pelo módulo de **cadastro de treinamentos**, para reconhecer determinada pessoa que está no campo de visão da câmera no momento de execução do reconhecedor, e através desse realizar o controle de frequência. Utilizando as aulas cadastradas e o reconhecimento facial ele faz a geração de um *log* com o controle de frequência.

Para que o reconhecimento seja feito o programa precisa inicialmente possuir um treinamento de no mínimo duas classes, possibilitando a execução da técnica de FisherFaces (Seção 2.2.9). Entende-se por classe um modelo de reconhecimento composto por várias imagens de treinamento associadas a um identificador (ID) da classe, sendo o mesmo associado à pessoa. Também faz-se necessário uma aula cadastrada para que esse realize a geração de *log*.

3.2.1 Treinamento

Para todo algoritmo de aprendizado de máquina tem-se necessidade de um treinamento prévio. No caso da aplicação tem-se um algoritmo de aprendizado de máquina não supervisionado, necessitando de um treino prévio para que ele possa realizar sua execução. O programa tem como entrada para seu treinamento apenas um arquivo contendo o caminho para uma imagem e o ID de sua respectiva classe (ID do usuário correspondente à foto do caminho).

A Figura 3.13 mostra um exemplo desse arquivo de entrada com os IDs 1,2,3 e 4. Na figura pode-se ver duas colunas formando a estrutura *caminho; ID*. A primeira linha, por exemplo, tem o caminho da imagem *./modified/1-67-80-153-84-3.jpg* e o ID 1. No nome da imagem é possível observar o ID e as posições dos olhos localizados na mesma, no caso da primeira linha de exemplo, tem-se o ID 1 e as posições dos olhos (67, 80) e (153, 84).

```
create_csv
./modified/1-67-80-153-84-3.jpg;1
./modified/3-5-3.jpg;3
./modified/1-75-82-153-87-3.jpg;1
./modified/4-8-3.jpg;4
./modified/2-5-3.jpg;2
./modified/1-85-89-189-83-3.jpg;1
./modified/1-85-101-183-108-3.jpg;1
./modified/1-75-90-160-94-3.jpg;1
./modified/1-70-88-152-98-3.jpg;1
./modified/1-89-109-176-120-3.jpg;1
./modified/1-60-76-130-81-3.jpg;1
./modified/3-2-3.jpg;3
./modified/1-73-86-150-92-3.jpg;1
./modified/4-1-3.jpg;4
./modified/1-60-77-134-82-3.jpg;1
./modified/1-68-79-141-87-3.jpg;1
./modified/2-10-3.jpg;2
./modified/4-9-3.jpg;4
./modified/3-1-3.jpg;3
```

Figura 3.13: Arquivo de exemplo de entrada do treinamento contendo duas colunas, o caminho da imagem de treino e seu ID associado.

O fluxo de treino pode ser visto na Figura 3.14. Cada linha do arquivo de entrada é um caso de treinamento para o programa. O treinamento sempre que iniciado passa por todas as linhas do arquivo de entrada, treinando a aplicação com todas as imagens descritas no arquivo. Cada par caminho/ID do arquivo de entrada cria ou associa a uma classe através do ID. Para o algoritmo ser executado são necessárias no mínimo duas pessoas cadastradas para que não haja apenas um padrão de comparação no algoritmo.

Para que possa ser feito um treinamento que gera resultados satisfatórios, é necessário existir as imagens descritas pelo arquivo de entrada do treino, possuir um único ID associado a uma pessoa e também várias imagens de treino em diferentes iluminações e posições para que o

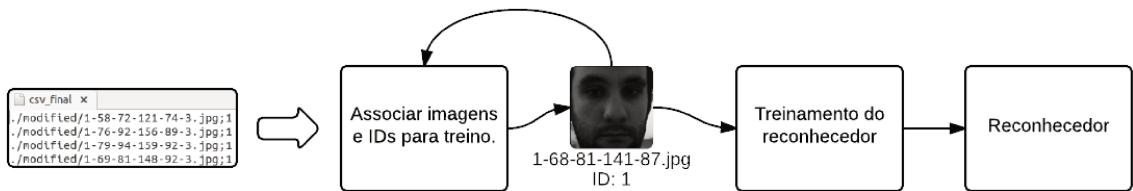


Figura 3.14: Fluxo do treinamento para a aplicação.

máximo de casos de treino sejam atendidos. Sem essa variação de iluminação e posições, devido a utilização de uma câmera de baixa resolução e a dificuldade da tarefa de reconhecimento facial, o reconhecedor confunde as classes, gerando erros no reconhecimento com variações nesses fatores.

Utilizando o algoritmo de FisherFaces (Seção 2.2.9), é necessário haver a construção de uma classe com variações nas imagens de treinamento, para aumentar a eficácia do reconhecimento. Mesmo ele sendo um algoritmo que sofre pouca influência de variações de luminosidade e expressões, elas implicam no seu treinamento. A Figura 3.15 mostra um exemplo de variação de iluminação e expressão nos dados de treinamento.

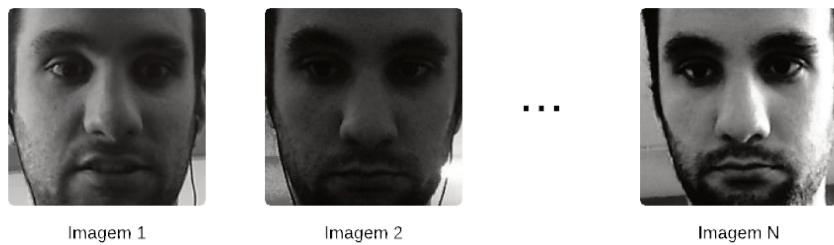


Figura 3.15: Exemplo de imagens para o treinamento com várias iluminações e expressões.

O treinamento é finalizado quando todas as imagens no arquivo de entrada são utilizadas pelo modelo. Quando isso ocorrer várias classes serão criadas para o modelo, uma para cada ID associado nas imagens. Essas classes são usadas para o reconhecimento, e são elas que contêm as características extraídas a partir de todas as imagens de treinamento de cada pessoa, possibilitando assim a construção de um reconhecedor.

3.2.2 Reconhecimento facial

A parte principal do programa é o **reconhecimento facial**, e também a parte final do fluxo apresentado pela Figura 3.16. Este é responsável pela identificação de pessoas em imagens digitais. Nessa fase o programa recebe uma imagem em tons de cinza, sendo que imagens de treinamento são obtidas em tons de cinza, das faces encontradas na imagem digital obtida pela câmera, então ela devolve o ID da classe que foi reconhecida para cada face.

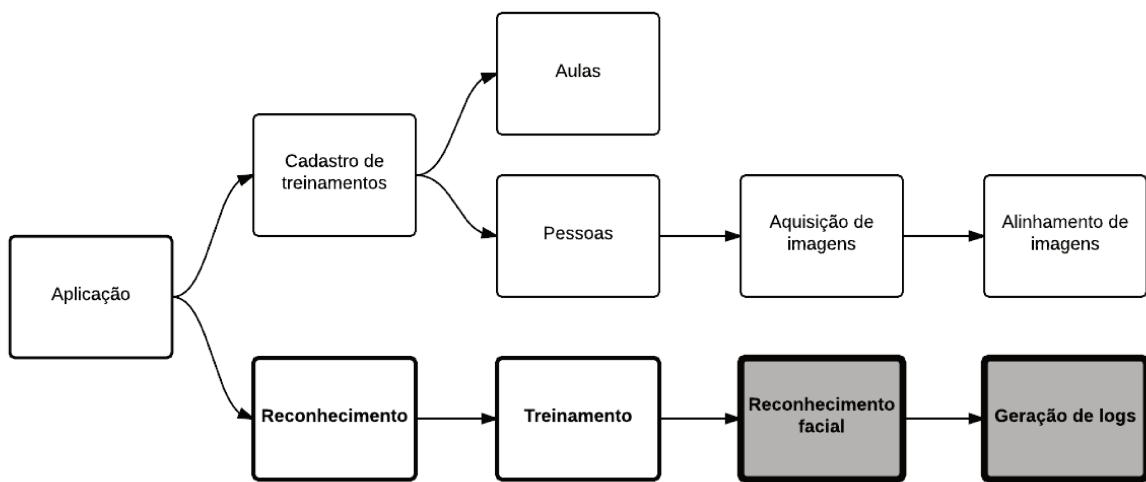


Figura 3.16: Diagrama da aplicação com destaque no reconhecimento facial

O sub-módulo de reconhecimento facial segue um fluxo mostrado pela Figura 3.17. Esse fluxo é responsável pela criação e utilização de um modelo reconhecedor. Um modelo reconhecedor é definido por um conjunto de várias classes de reconhecimento, sendo essas características extraídas de imagens de pessoas com IDs associados. Essas classes são analisadas pelo modelo reconhecedor, que então pode fazer uma comparação entre elas e uma nova imagem de entrada.



Figura 3.17: Fluxo do reconhecimento aplicação.

A parte inicial para o reconhecimento é a criação do reconhecedor através da OpenCV. A criação desse modelo é feita com a definição de máxima distância entre as características

dos modelos de treinamento e a imagem atual do reconhecimento. Caso ele não encontre uma classe compatível, retornará -1 para a face encontrada na imagem que não foi reconhecida. O caso é exemplificado na Figura 3.18. O ID retornado pela função de reconhecimento é o mesmo ID da classe cadastrada, assim se houver no treinamento o ID 1 associado a uma pessoa, e esse for o ID de retorno do reconhecimento, então a classe 1 seria a que mais aproxima-se à imagem obtida pela câmera.



Figura 3.18: Resultado de quando não é reconhecida a pessoa na imagem, retornando o $ID : -1$.

Após a criação do modelo de reconhecimento através da *OpenCV*, o sub-módulo passa para a próxima fase, aquisição da imagem da câmera. Com essas imagens obtidas utiliza-se o algoritmo de Característica Haar-Like em cascata (Seção 2.2.2) na imagem para serem encontrados os rostos que existem nela. Também utiliza-se o treinamento padrão da *OpenCV* para o algoritmo.

Sabendo quais são os rostos que estão na imagem capturada pela câmera aplica-se o algoritmo de FisherFaces em cada um deles. Através do modelo de treinamento, resultante da fase anterior ao reconhecimento, esse algoritmo distingue as características da pessoa e a compara com cada uma das classes geradas no treino.

O reconhecedor usa a classe que mais se aproxima da imagem atual obtida, caso ela esteja dentro dos limites de diferenças definidos na criação do reconhecedor. Por fim ele obtém o ID da classe reconhecida e o retorna. O ID associado à classe que foi reconhecida é exibido na tela logo acima da cabeça do usuário, conforme a Figura 3.19.

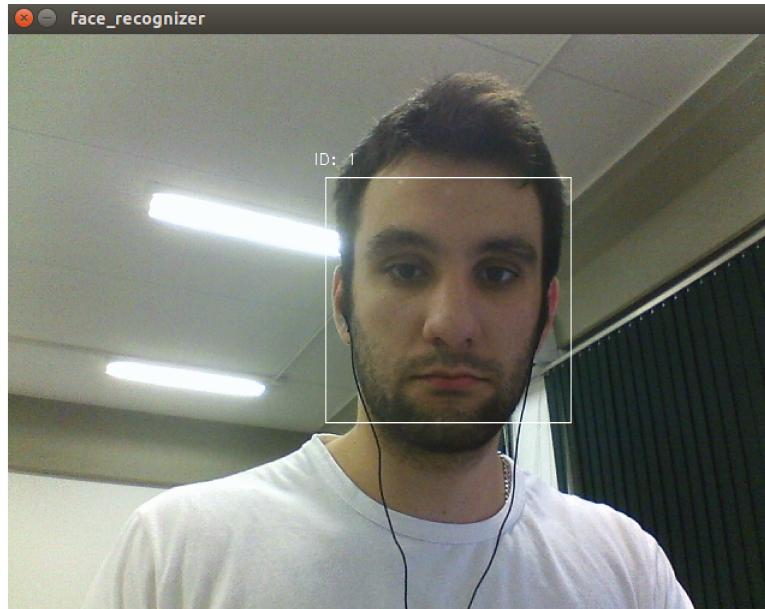


Figura 3.19: Exemplo de retorno do reconhecimento de faces.

3.2.3 Geração de *logs*

A geração de *log* inicia quando um ID válido é encontrado. Essa é responsável pela criação de um *log* qual é salvo o controle de frequência da aula iniciada no reconhecimento. Esse sub módulo tem como entrada o ID da pessoa reconhecida e como saída um arquivo de *log* com várias linhas, sendo cada uma dessas um controle de frequência. Nele tem-se o nome do aluno cadastrado e o momento que sua presença foi registrada. O fluxo do sub-módulo é exibindo na Figura 3.20.

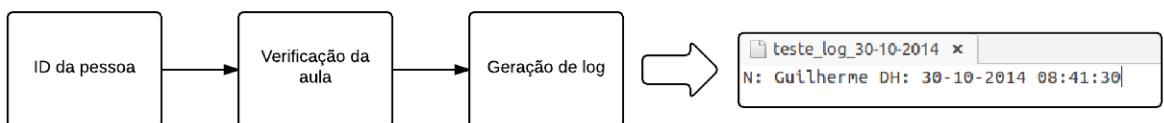


Figura 3.20: Fluxo do reconhecimento aplicação com exemplo de *log* de saída.

Quando uma classe de reconhecimento válida é encontrada o fluxo do sub-módulo inicia-se. Primeiramente há uma verificação se o ID da pessoa reconhecida está entre um dos cadastrados no arquivo da aula que foi inserido no começo do reconhecimento. Caso ele exista é gerado um arquivo de *log*, ou atualizado o já existente, inserindo uma linha adicional com o nome do aluno cadastrado e a data e hora que ele foi reconhecido.

Um exemplo de geração de *log* pode ser observado na Figura 3.21. Essa imagem mostra um *log* de apenas uma linha, onde pode-se observar duas informações: o nome do aluno e o

momento do registro de seu reconhecimento. Nesse caso o aluno *Guilherme* foi registrado às 08 : 41 : 30 do dia 30/10/2014.

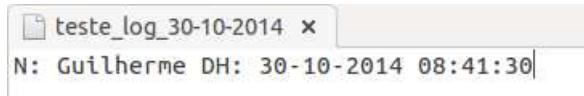


Figura 3.21: Exemplo de arquivo de log para o controle de frequência, onde apenas um aluno, *Guilherme*, foi reconhecido, no instante 30/10/2014 08 : 41 : 30.

Esse processo é repetido para cada nova entrada de imagem da câmera. Todas as pessoas que aparecerem e forem reconhecidas na imagem obtida pela câmera e registradas na aula aparecerão no *log*. Será gerado um arquivo de *log* com as pessoas presentes na aula para cada dia em que a aplicação for rodada, para facilitar o controle de frequência do aluno, por exemplo. Também são gerados diferentes arquivos de *logs* de frequência para cada aula inicializada no reconhecimento. Na Figura 3.21, por exemplo, o nome da aula, *teste*, aparece no arquivo de *log* de frequência, sendo essa a aula de entrada dada para o programa pelo usuário.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos com a aplicação de reconhecimento facial desenvolvida no presente trabalho. Para essa espera-se um reconhecimento de várias pessoas em uma imagem digital obtida pela câmera local de uma máquina. Os testes da aplicação foram organizados com múltiplas pessoas para realizar um reconhecimento simulando um caso real, como uma sala com vários alunos. Para essas múltiplas pessoas são usados treinamentos diferentes, que variam de acordo com o teste. Para os resultados não foram analisados os cadastros de treinamentos, tanto de aula quanto de pessoas.

Realizou-se um teste de performance para medir o tempo de reconhecimento da aplicação. Cada caso de teste possui variação no número de pessoas da imagem para simular vários cenários para o reconhecimento. Verificou-se o crescimento no tempo de reconhecimento de acordo com o número de pessoas.

Todos os testes realizados tem o objetivo de simular casos reais da aplicação de reconhecimento facial. Para isso realizou-se uma troca nos dados de treinamentos para demonstrar as diferenças de reconhecimento. Em todos esses testes espera-se o reconhecimento de todas as pessoas de acordo com o treinamento prévio.

Os testes foram separados em duas etapas: não reconhecimento de uma pessoa presente na imagem, mas não cadastrada na lista de alunos, e o teste de reconhecimento da pessoa cadastrada na aplicação.

4.1 Reconhecimento

Os testes de reconhecimento da aplicação foram realizados simulando casos reais de um controle de frequência. Para as pessoas da imagem espera-se o reconhecimento de todas as faces detectadas, e também a detecção de todas as faces da imagem. Os testes foram feitos para uma e para várias pessoas na mesma imagem capturada pela câmera, demonstrando exemplos de vários casos da aplicação. A aplicação deve reconhecer todos os rostos na imagem que foram cadastrados previamente. Os resultados esperados para esses testes são dois: o reconhecimento de pessoas cadastradas e o não reconhecimento de pessoas não cadastradas na aplicação.

A alteração de treinamentos para as pessoas também foi realizada, observando a mudança de resultados conforme o número de casos de treino para cada pessoa. Essas imagens de treinamento de cada pessoa pré-cadastradas no sistema englobam vários casos de posições,

iluminações e expressões diferentes. Esses testes mostram o impacto do número de treinamento na aplicação.

Para o teste de não reconhecimento foram inicializadas as classes de treino de pessoa que não aparecem na imagem digital no qual o reconhecedor é aplicado. Quando a imagem capturada pela câmera não é reconhecida então a saída do ID fica como -1 , exemplificado na parte superior da Figura 4.1. O reconhecedor da figura foi construído com duas margens de erro entre as características, uma delas pequena e a outra grande. Essa figura mostra dois casos, o de não reconhecimento e o de erro. A margem de erro é encontrada testando a aplicação com casos de treinos com uma taxa de acerto considerável, podendo definir qual a distância máxima das características.

O treinamento do reconhecedor foi construído por duas classes, ambas inexistentes na imagem analisada. O caso de não reconhecimento o ID: -1 aparece para a imagem, sendo esse um caso correto da aplicação, que a margem de erro das características da face do reconhecedor é pequena. Aumentando-se essa margem de erro, o reconhecedor erra e acaba tendo falhas no não reconhecimento, observado na parte inferior da figura.

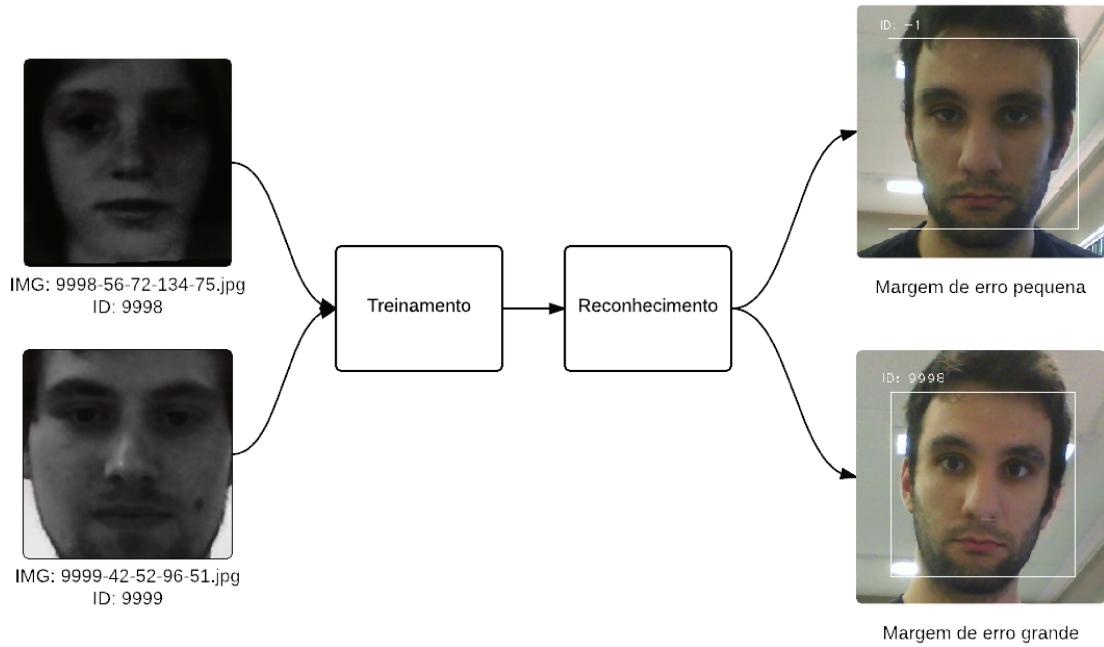


Figura 4.1: Exemplo de reconhecimento de pessoa não cadastrada, mostrando a variação da margem de erro para o reconhecedor.

O teste para verificar as pessoas cadastradas simula o reconhecimento de várias pessoas em uma única cena. Para essa simulação faz-se necessário um treinamento prévio da aplicação.

O treinamento realizado para os testes foi composto de seis pessoas voluntárias para o reconhecimento facial. Cada pessoa recebeu um treinamento diferenciado para o número de imagens. A Figura 4.2 mostra o treinamento utilizado para a realização dos testes para as pessoas com cada ID associado e o número de imagens de treino para cada pessoa cadastrada.

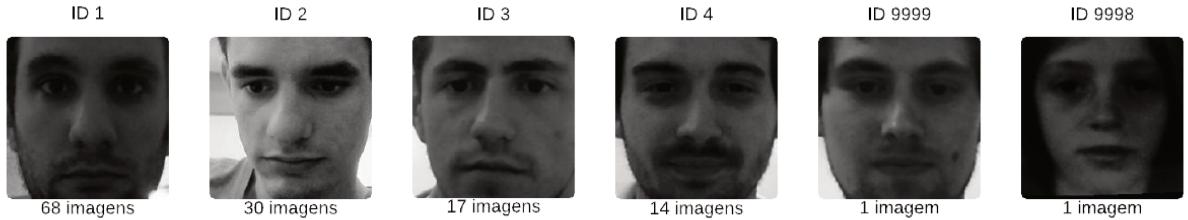


Figura 4.2: Pessoas utilizadas no treinamento para os testes da aplicação de reconhecimento facial.

Este treinamento foi definido como o padrão dos testes, sendo que cada caso de teste tem um objetivo. Apenas o número de imagens de treino de cada pessoa e o número de pessoas treinadas na aplicação varia em cada teste.

Para o teste de não reconhecimento de pessoas, inseriu-se um treinamento com as pessoas de ID 9999 e 9998 na aplicação. O esperado para o teste de não reconhecimento é nenhum reconhecimento realizado para as pessoas detectadas na imagem digital, sendo essas quaisquer diferentes das pessoas de ID 9999 e 9998. Três pessoas foram utilizadas no teste de não reconhecimento. A Figura 4.3 demonstra o resultado obtido pelo programa com essas definições. Observa-se três pessoas na imagem capturada pela câmera, porém nenhuma delas previamente cadastrada no sistema.

As três pessoas encontradas pela imagem, na direita, centro e esquerda, estão com os IDs respectivos como –1. Nenhuma delas aproximou-se de algum treinamento prévio realizado para o reconhecedor, o que é o resultado esperado, visto que apenas as classes de ID 9999 e 9998 poderiam ser reconhecidas. Com a margem de erro das características, nenhuma das pessoas da imagem foi reconhecida, porém três pessoas são encontradas na imagem, conforme o esperado, nenhuma pessoa é reconhecida a partir do grupo de treinamento inserido na aplicação, que continha apenas as pessoas de ID 9999 e 9998.

Uma das tarefas esperadas para a aplicação é o reconhecimento de múltiplas pessoas em uma imagem digital. Em um caso real várias pessoas apareceriam em uma imagem, sendo que todas devem ser reconhecidas, se estiverem cadastradas na imagem, como mostra a Figura 4.4.

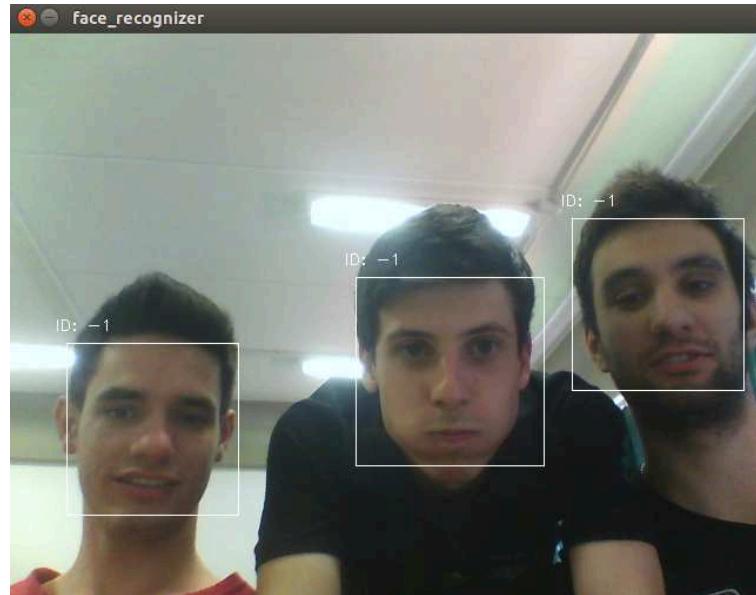


Figura 4.3: Três pessoas foram encontradas na imagem, mas nenhuma delas estava previamente cadastrada.

O teste de reconhecimento múltiplo demonstra a capacidade da aplicação de detectar e reconhecer várias faces em uma imagem, sendo essas cadastradas, ou casos de não reconhecimento para pessoas não cadastradas. Para esse teste utilizou-se o treinamento padrão, com todas as pessoas envolvidas cadastradas, conforme a Figura 4.2.



Figura 4.4: A esquerda: exemplo de imagem com reconhecimento e não reconhecimento. A direita: exemplo de imagem com múltiplo reconhecimento.

A Figura 4.4 mostra dois exemplos de reconhecimento facial múltiplo. Na imagem à esquerda, há um reconhecimento e um não reconhecimento na mesma imagem. Apenas a pessoa da esquerda (ID 1) estava cadastrada, enquanto a pessoa da direita (ID -1), que não está cadastrada, não foi reconhecida pela aplicação. Essa imagem demonstra um caso real onde algum desconhecido estaria no campo de visão da câmera, sendo que esse deve ser detectado,

mas não registrado pela aplicação.

Na imagem da direita, observa-se o reconhecimento de todas as pessoas encontradas na imagem conforme o treinamento de entrada. Observa-se no canto inferior esquerdo o reconhecimento do rosto mesmo com uma leve inclinação. Os outros estão em posição normal e ambos foram reconhecidos corretamente. O caso de inclinação do rosto não é atendido pelo programa, porém, neste caso funcionou, pois a inclinação da pessoa reconhecida não é muito grande. Caso a inclinação fosse grande, a detecção da pessoa não seria feita.



Figura 4.5: Exemplo de imagem com faces não detectadas, estando elas de perfil e inclinadas.

Na Figura 4.5 observa-se um reconhecimento facial realizado para a pessoa do meio, que foi previamente cadastrada. Outras duas pessoas são vistas na imagem, porém, ambas não são detectadas pela aplicação. Na pessoa da esquerda, a detecção não foi feita pela inclinação da cabeça. O algoritmo não foi treinado para casos como esse, logo a detecção e reconhecimento não são feitos para a pessoa. A pessoa da direita não foi detectada também. Para essa a detecção e reconhecimento não foram feitos pelo fato de encontrar-se de perfil, um caso que o algoritmo não foi treinado também. Para o reconhecimento ser realizado com sucesso a pessoa deve estar de frente para a câmera e sem inclinação, ou com uma leve inclinação, só assim então a face será reconhecida.

Outro caso de falha do programa são os reconhecimentos errados. A incidência de reconhecimentos errados ocorre quando o treinamento de uma pessoa não possui informações suficientes para o reconhecimento. A Figura 4.6 demonstra um exemplo de reconhecimento com falha. Nela há quatro pessoas, todas detectadas, porém no reconhecimento das faces,

utilizando o treinamento padrão dos testes, o resultado deveria retornar os respectivos IDs 1, 2, 3 e 4. Três delas são reconhecidas corretamente, no canto inferior direito com o ID 3, no canto inferior esquerdo com o ID 4 e no canto superior direito com o ID 1. O ID 1, porém, aparece na pessoa central da imagem, sendo que essa pessoa está associada ao ID 2.



Figura 4.6: Exemplo de imagem com quatro faces, onde uma foi reconhecida errada, utilizando o treinamento padrão dos teste.

Casos de reconhecimento errado como esse são comuns quando o treinamento possui imagens insuficientes para criar uma classe com boas chances de reconhecimento. Esse erro pode ocorrer pela posição da face no treinamento, com uma inclinação, por exemplo, ou no reconhecimento a face estar em uma posição que não é atendida pelo treinamento. Também erros como esse são comuns no caso de variações nas expressões faciais. Uma foto com uma expressão diferente, que não está cadastrada em algum dos treinamentos, pode gerar um erro de reconhecimento pelo fato da mudança das características da pessoa na expressão facial.

4.1.1 Treinamentos

O reconhecimento facial através de imagens digitais é uma técnica de aprendizado de máquina, necessitando de um treinamento. Para o reconhecimento ser eficaz, é necessário fazer um treinamento significativo, que englobe vários casos para o reconhecimento. O teste de eficácia dos treinamentos foi realizado com o intuito de analisar a quantidade de imagens por pessoa para uma melhor eficácia do reconhecimento. Para esse teste foi utilizado um treinamento da aplicação personalizado, variando de 1 a 50 imagens para cada pessoa, exibidas na Figura 4.7.

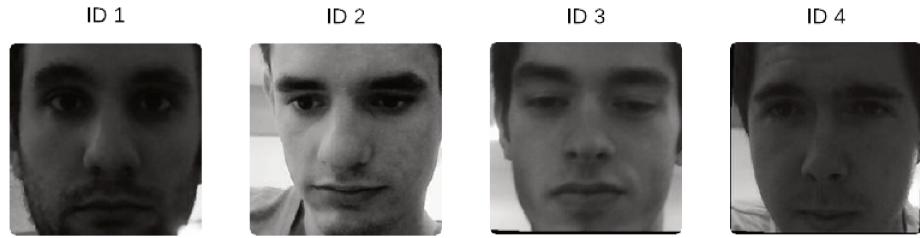


Figura 4.7: Pessoas utilizadas para o teste de acurácia.

Para o teste de acurácia foram utilizadas 20 imagens tiradas previamente de cada uma das pessoas exibidas na Figura 4.7. Para essas imagens foi aplicado a detecção e o reconhecimento. A Tabela 4.1 mostra a progressão do treinamento, relacionando o número de imagens de treinamento como a taxa de acerto no reconhecimento. O teste de acurácia foi realizado com essas 20 imagens, elas são a base da coluna de **Acertos** da Tabela 4.1. Observa-se números de treinamentos para cada pessoa, e os acertos de cada reconhecimento para as 20 imagens inseridas no reconhecedor.

Número de treinamentos	ID da pessoa	Acertos	Taxa de acerto	Média
1	1	1/20	5%	3,75%
	2	2/20	10%	
	3	0/20	0%	
	4	0/20	0%	
5	1	9/20	45%	41,25%
	2	13/20	65%	
	3	6/20	30%	
	4	5/20	25%	
10	1	14/20	70%	55%
	2	8/20	40%	
	3	11/20	55%	
	4	9/20	45%	
20	1	19/20	95%	75%
	2	16/20	80%	
	3	13/20	65%	
	4	12/20	60%	
50	1	20/20	100%	86,25%
	2	17/20	85%	
	3	17/20	85%	
	4	15/20	75%	

Tabela 4.1: Resultados da aplicação de reconhecimento facial com variação do número de imagens do treinamento.

Para cada pessoa foi alterado o número de treinamento de sua classe reconhecedora,

visto na primeira coluna da Tabela 4.1 , alterando assim os resultados conforme seu treinamento. A Tabela 4.1 exibe a variação da taxa de acerto relacionado pelo número de treinamentos para cada uma das pessoas utilizadas nesse teste. Há uma progressão da taxa de acerto conforme o aumento de número de treinamentos para cada uma das pessoas definidas nas imagens do teste de acurácia. Com esse teste pode observar-se a influência do treinamento no reconhecimento facial da aplicação, onde um maior treinamento implica no aumento da taxa de reconhecimento.

5 CONCLUSÃO

Observa-se uma evolução na área de visão computacional devido ao avanço tecnológico em questões tanto de *hardware* como de *software*. Essa evolução fez com que novas tarefas fossem possíveis, como o maior desempenho de *hardware* e a otimização de novas técnicas de *software*. Uma dessas tarefas, que a visão computacional pode realizar, é a detecção e reconhecimento de rostos em imagens digitais. A possibilidade desse reconhecimento de pessoas em imagens digitais possibilita a *softwares* o controle de ambientes através de dados obtidos por uma câmera. Um desses controles, que pode ser realizado com visão computacional através de uma câmera, é o de controle de frequência automatizado. Um controle de frequência automatizado faz-se útil em qualquer situação de controle de tráfego de pessoas em determinado ambiente.

Os resultados da aplicação demonstram a possibilidade de um reconhecimento facial múltiplo em ambientes monitorados. Para o reconhecimento facial apresentou-se uma solução baseada em visão computacional que segue um fluxo de reconhecimento. Esse fluxo pode ser descrito em: criação do modelo reconhecedor; obtenção de imagens; detecção de faces; reconhecimento de faces detectadas.

A criação do modelo reconhecedor obtém imagens de treinamento para associar a pessoas que devem ser reconhecidas. A obtenção de imagens abre uma comunicação com a câmera que envia uma imagem para a aplicação. A detecção de faces utiliza um treinamento para o reconhecimento de característica comuns nas faces, detectando-as e extraiendo-as. O reconhecedor é aplicado em cada uma dessas imagens, esperando um resultado de reconhecimento ou não reconhecimento da pessoa.

O reconhecimento é aplicado em todas as faces encontradas na imagem. Com esse reconhecimento é possível criar um controle de frequência, com as informações presentes em algum ambiente em determinada hora. A aplicação cria um *log* desse controle realizado. O sistema foi desenvolvido em módulos independentes, podendo assim haver uma futura integração com outro sistema.

5.1 Trabalhos futuros

O reconhecimento de perfil e face inclinada, não foi realizado para detecção e também para reconhecimento na aplicação. Para a detecção utilizou-se exclusivamente a face frontal,

detectando-a apenas em casos específicos e sem, ou com pouca, inclinação. O funcionamento da detecção pode ser melhorado com um maior treinamento usando os casos inclinados das faces. Para isso seria necessário o treinamento com pessoas em perfil e inclinadas, com um reconhecedor da pessoa de vários ângulos diferentes.

A aplicação de controle de frequência possui um gerenciamento de frequência rudimentar. O controle de frequência pode ser integrado com outros sistemas. Como a aplicação foi separada em módulos, o controle de frequência poderia ser integrado a outra aplicação, como o sistema acadêmico da própria UFFS. Isso eliminaria a adição manual das informações de aulas e de alunos, por exemplo.

Para melhorar o desempenho da aplicação poderiam ser realizados testes nos treinamentos de pessoas para um reconhecimento mais eficaz. O treinamento poderia ser testado com diferentes iluminações, criando um modelo sobre quanto o reconhecimento é afetado devido a variação de iluminação encontrada nos treinamentos. Outros testes podem ser feitos para construir um treinamento ideal, onde seria possível definir quantas fotos, com quais posições, e com quantas iluminações e expressões seriam necessárias para o reconhecedor ter a melhor taxa de reconhecimento.

Comparações entre algoritmos de reconhecimento facial também podem ser realizados para esclarecer qual seria o método com maior acurácia e também melhor desempenho.

REFERÊNCIAS

- [1] R. Amaro and A. Dieguez. imagem do site iceinteractive. 2010.
- [2] M. C. Burl, T. Leung, and P. Perona. Face localization via shape statistics. In *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, pages 154–159, 1995.
- [3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [5] M. P. de Albuquerque and M. P. de Albuquerque. Processamento de imagens: métodos e análises. *Centro Brasileiro de Pesquisas Físicas MCT*, 2000.
- [6] E. R. Dougherty and R. A. Lotufo. Hands-on morphological image processing. Spie Bellingham, 2003.
- [7] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In P. Vitányi, editor, *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin Heidelberg, 1995.
- [8] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5):771–780, 1999.
- [9] D. González-Ortega, F. Diaz-Pernas, J. Diez-Higuera, M. Martinez-Zarzuela, and D. Boto-Giralda. Computer vision architecture for real-time face and hand detection and tracking. In *Visual Information and Information Systems*, pages 35–49. Springer, 2006.
- [10] H. Gunes, M. Piccardi, and T. Jan. Face and body gesture recognition for a vision-based multimodal analyzer. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*, VIP '05, pages 19–28, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

- [11] X.-Y. Jing, H.-S. Wong, and D. Zhang. Face recognition based on 2d fisherface approach. *Pattern Recognition*, 39(4):707–710, 2006.
- [12] J. C. Klontz, B. F. Klare, S. Klum, A. K. Jain, and M. J. Burge. Open source biometric recognition. *To appear in IEEE BTAS*, 2013.
- [13] J. Kovac, P. Peer, and F. Solina. *Human skin color clustering for face detection*, volume 2. IEEE, 2003.
- [14] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 157–164, 2003.
- [15] K.-C. Kwak and W. Pedrycz. Face recognition using a fuzzy fisherface classifier. *Pattern Recognition*, 38(10):1717–1732, 2005.
- [16] S. Z. Li and A. K. Jain. *Handbook of face recognition*. Springer, 2011.
- [17] M. Marengoni and S. Stringhini. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, 16(1):125–160, 2010.
- [18] J. Meynet. Fast face detection using adaboost. *University of Trier*, 2003.
- [19] B. Miranda. imagem do site opservices. 2012.
- [20] B. Munsell, A. Temlyakov, C. Qu, and S. Wang. Person identification using full-body motion and anthropometric biometrics from kinect videos. In A. Fusiello, V. Murino, and R. Cucchiara, editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, volume 7585 of *Lecture Notes in Computer Science*, pages 91–100. Springer Berlin Heidelberg, 2012.
- [21] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Computer Vision, 1998. Sixth International Conference on*, pages 555–562, 1998.
- [22] F. M. Reaes. Reconhecimento de faces em imagens: Projeto beholder. *Trabalho de formatura Supervisionado–Universidade De São Paulo (USP)*-2006, 2006.
- [23] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.

- [24] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.
- [25] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003.
- [26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.
- [27] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.
- [28] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, 2002.
- [29] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.