

**UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO**

LAHIS GOMES DE ALMEIDA

**SISTEMA DE MONITORAMENTO DE AMBIENTES PRIVADOS BASEADO
EM VISÃO COMPUTACIONAL E *INTERNET DAS COISAS***

Manaus

2017

LAHIS GOMES DE ALMEIDA

**SISTEMA DE MONITORAMENTO DE AMBIENTES PRIVADOS BASEADO
EM VISÃO COMPUTACIONAL E *INTERNET DAS COISAS***

Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Engenheiro de Computação.

Orientador(a): Prof. Dr. Carlos Maurício Seródio Figueiredo

Manaus

2017

Universidade do Estado do Amazonas - UEA
Escola Superior de Tecnologia - EST

Reitor:

Cleinaldo de Almeida Costa

Vice-Reitor:

Mario Augusto Bessa de Figueiredo

Diretor da Escola Superior de Tecnologia:

Roberto Higino Pereira da Silva

Coordenador do Curso de Engenharia de Computação:

Raimundo Corrêa de Oliveira

Coordenador da Disciplina Projeto Final:

Raimundo Corrêa de Oliveira

Banca Avaliadora composta por:

Data da Defesa: / /2017.

Prof. Dr. Carlos Maurício Serório Figueiredo(Orientador)

Prof. M.Sc Ingrid Sammyne Gadelha Figueiredo

Prof. Bel. Ismael Junior Vidal Paz

CIP – Catalogação na Publicação

L864a ALMEIDA, Lahis Gomes de

Sistema de Monitoramento de Ambientes Privados baseado em Visão Computacional e *Internet das Coisas*/ Lahis Almeida; [orientado por] Prof. Dr. Carlos Maurício Serório Figueiredo – Manaus: UEA, 2017.

240 p.: il.; 30cm

Inclui Bibliografia

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação).

Universidade do Estado do Amazonas, 2017.

CDU: _____

LAHIS GOMES DE ALMEIDA

**SISTEMA DE MONITORAMENTO DE AMBIENTES PRIVADOS BASEADO
EM VISÃO COMPUTACIONAL E *INTERNET DAS COISAS***

Trabalho de Conclusão de Curso apresentado
à banca avaliadora do Curso de Engenharia
de Computação, da Escola Superior de
Tecnologia, da Universidade do Estado do
Amazonas, como pré-requisito para obtenção
do título de Engenheiro de Computação.

Aprovado em: / /2017

BANCA EXAMINADORA

Prof. Dr. Carlos Maurício Seródio Figueiredo, Doutor.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. M.Sc. Ingrid Sammyne Gadelha Figueiredo, Mestre.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Prof. Ismael Junior Vidal Paz, Engenheiro de Computação.

UNIVERSIDADE DO ESTADO DO AMAZONAS

Resumo

A utilização de mecanismos de acesso a ambientes privados, como senhas, impressão digital e, até mesmo, a checagem de nomes em lista de autorização, é muito comum. Esses mecanismos acabam sendo um incômodo na vida de seus usuários pois são burocráticos e intrusivos. Com o avanço de tecnologias de Visão Computacional e Internet das Coisas, esse tipo de cenário tende a ficar para trás. Este trabalho propõe um sistema de monitoramento de baixo custo através de detecção e reconhecimento facial aliado com conceitos de *Internet das Coisas*, como conectividade constante e capacidade de controle remoto.

Palavras Chave: Sistema de Monitoramento, *internet* das coisas, reconhecimento facial, *raspberry pi*, *mqtt*.

Abstract

The use of mechanisms for access private environments, such as passwords, fingerprints, and even the checking of names in authorization lists, is very common. These mechanisms end up being a nuisance in the life of its users because they are bureaucratic and intrusive. With the advancement of Computer Vision and the Internet of Things technologies, this kind of scenario tends to lag behind. This work proposes a low cost monitoring system through detection and facial recognition allied to Internet of Things concepts, such as constant connectivity and remote control capability.

Key-words: Monitoring Systems, internet das coisas, facial recognition, *raspberry pi*, *mqtt*.

Sumário

Lista de Tabelas	vii
Lista de Figuras	ix
1 Introdução	1
1.1 Descrição do Problema	1
1.2 Objetivos Gerais	2
1.2.1 Objetivos Específicos	2
1.3 Justificativa	3
1.4 Metodologia	3
2 Fundamentação Teórica	5
2.1 Visão Computacional	5
2.1.1 Imagem Digital	5
2.2 Detecção Facial	7
2.2.1 Métodos de Detecção Facial	7
2.2.2 Algoritmo <i>Viola-Jones</i>	9
2.3 Reconhecimento Facial	13
2.3.1 <i>Eigenfaces</i>	14
2.3.2 <i>Fisherfaces</i>	17
2.3.3 <i>Local Binary Patterns</i>	20
2.3.4 <i>OpenCV</i>	23

2.4	Internet das Coisas (<i>Internet of Things</i>)	24
2.4.1	Sistemas Embarcados	24
2.4.2	Protocolo de comunicação MQTT	26
2.5	Trabalhos Relacionados	27
3	Solução Proposta	29
3.1	Sistema de Monitoramento de Ambientes	29
3.1.1	Captura de Fotografia e Processamento do Sistema	31
3.1.2	Reconhecimento Facial	31
3.1.3	Envio de Notificação ao Aplicativo	32
4	Experimentos e Resultados	34
4.1	Etapa de Detecção Facial	34
4.1.1	Experimentos da Detecção Facial	35
4.1.2	Resultados da Detecção Facial	35
4.2	Etapas de Extração de Características e Reconhecimento	37
4.2.1	Experimentos do Reconhecimento Facial com AT&T	38
4.2.2	Resultados do Reconhecimento Facial	40
4.3	Etapa de Teste de Sistema de Monitoramento	41
4.3.1	Experimentos e Resultados do Sistema de Monitoramento	41
5	Conclusão	45

Lista de Tabelas

4.1	Resultados do Reconhecimento Facial	40
4.2	Resultados das Métricas de Comparação dos Algoritmos	41
4.3	Resultados do Monitoramento em Tempo Real	44

Lista de Figuras

2.1	Níveis de Cinza. (JUNIOR, 2014)	6
2.2	Representação de Imagem Digital 2D. (QUEIROZ; GOMES, 2006)	7
2.3	Método <i>Knowlegde Based.</i> (BRAGA et al., 2013)	8
2.4	Método <i>Template Based.</i> (BRAGA et al., 2013)	9
2.5	Etapas do Algoritmo <i>Viola-Jones</i>	10
2.6	Exemplos de Sub-regiões. (BRAGA et al., 2013)	11
2.7	Etapa de <i>Training Classifier</i>	11
2.8	Cascata de Classificadores. (BRAGA et al., 2013)	12
2.9	Etapas do Reconhecimento Facial.	14
2.10	Conjunto de Imagens. (HANZRA, 2015)	15
2.11	Fluxograma do Algoritmo <i>Eigenfaces</i>	16
2.12	Subespaço do <i>Eigenfaces</i> . (MACHADO et al., 2009)	17
2.13	Gráficos de PCA e LDA. (BRAGA et al., 2013)	19
2.14	Fluxograma do Algoritmo <i>Fisherfaces</i>	20
2.15	Análise LBP. (DA; SANG, 2009)	20
2.16	Variações de <i>Neighbors</i> . (OPENCV, 2012)	21
2.17	Circular Neighboors. (OPENCV, 2012)	22
2.18	Variações nos níveis de cinza. (OPENCV, 2012)	22
2.19	Fluxograma do Algoritmo LBP.	23
2.20	<i>Raspberry Pi</i> . (SILVA; SILVA, 2014)	25
2.21	Visão Geral do Protocolo MQTT (BARROS, 2015).	26

3.1	Sistema de Monitoramento Proposto.	30
3.2	Fluxograma do Sistema de Monitoramento Proposto.	31
3.3	Abordagem de comunicação adotada.	32
4.1	Testes de Detecção Facial em fotografias. (ROSEBROCK, 2016)	36
4.2	Testes de Detecção Facial em tempo real.	37
4.3	Testes de Reconhecimento Facial.	39
4.4	Conjunto de Imagens de teste selecionado.	39
4.5	<i>Scripts</i> Desenvolvidos.	41
4.6	Exemplo de Fotos Cadastradas por Usuário.	42
4.7	Monitoramento em Tempo Real.	42
4.8	Exemplo de Fotos Cadastradas por Usuário.	43

Capítulo 1

Introdução

Neste capítulo, serão apresentados os principais problemas que cercam o monitoramento de ambientes privados e como eles podem ser sanados com a abordagem proposta por este trabalho. Os objetivos, justificativa e metodologia também serão detalhados nesse capítulo a fim de mostrar a importância deste estudo e as etapas em que ele será desenvolvido.

1.1 Descrição do Problema

O monitoramento de ambientes privados é essencial para garantir a segurança de seus integrantes, bem como de seus produtos e equipamentos, evitando situações de cunho criminoso, como furtos e exposição de informações confidenciais. Tal monitoramento é realizado por mecanismos como cartões de identificação, senhas, biometria (e.g., impressão digital, reconhecimento de voz) e, até mesmo, listas com nomes de pessoas autorizadas. Entretanto, esses mecanismos tornam o acesso a esses ambientes burocrático, devido ao tempo que demandam para realizar a validação, muitas vezes tendo que ser repetidos pela demora na identificação do usuário (PATIL; SHUKLA, 2014); e incômodo, devido os usuários terem que interagir diretamente com o ambiente, seja encostando a palma da mão ou a ponta dos dedos, seja por exposição de sua voz (reconhecimento por voz).

À medida que avanços tecnológicos nas áreas de Visão Computacional e Sistemas Embardados crescem, métodos de monitoramento como citados anteriormente vão se tornando obsoletos.

Nesse contexto, um método que acompanha essas tendências tecnológicas é o monitoramento por reconhecimento facial. As vantagens desse método biométrico em relação aos demais são: não há interação do usuário (e.g., intervenção humana, contato físico) e a validação é rápida e eficiente, evitando a espera de acesso liberado.

Muitos sistemas de captura de imagens são baseados em computadores tradicionais (*desktops*). Contudo, a portabilidade deles é limitada pelo seu peso, tamanho, alto consumo de energia e, principalmente, pelo seu custo (SENTHILKUMAR; GOPALAKRISHNAN; KUMAR, 2014).

No paradigma de *Internet das Coisas* (*Internet of Things - IoT*), muitos dos objetos que rodeiam a população estão em rede. Sistemas de informação e comunicação estão invisíveis aos seus usuários, resultando na geração de enormes quantidades de dados que devem ser armazenados e apresentados de forma transparente e eficiente. Para prover este cenário, a *Internet das Coisas* se alia a Computação na Nuvem (*Cloud Computing*) em busca de uma infraestrutura virtual, que integre dispositivos de monitoramento, dispositivos de armazenamento, ferramentas de análises, entre outros, possibilitando a utilização de seus serviços em tempo real de qualquer lugar (GUBBI et al., 2013).

1.2 Objetivos Gerais

Projetar e implementar um sistema de monitoramento de ambientes baseado em reconhecimento de faces em plataforma embarcada.

1.2.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Avaliação experimental de algoritmos clássico de visão computacional para reconhecimento de faces;
- Experimentação dos algoritmos em plataforma computacional de baixo custo;

- Disponibilização de um protótipo de sistema para monitoramento de segurança.

1.3 Justificativa

As principais motivações do trabalho são: (i) evitar, através do reconhecimento facial, a burocracia e desconforto gerados pelos métodos biométricos de controle de acesso como, por exemplo, reconhecimento de voz, impressão digital e reconhecimento da íris; (ii) minimizar custos e possibilitar a portabilidade do sistema de monitoramento por meio do uso de sistemas embarcados e (iii) permitir o monitoramento de ambientes, a qualquer momento e em qualquer lugar, por meio de protocolo de *Internet das Coisas*.

1.4 Metodologia

A metodologia adotada no desenvolvimento deste trabalho consiste na execução das seguintes atividades:

1. Realizar levantamento bibliográfico, identificando e estudando as principais referências bibliográficas sobre detecção facial, reconhecimento facial e *internet das coisas*, em particular, sobre os algoritmos *Viola-Jones*, *Eigenfaces*, *Fisherfaces* e LBPH.
2. Implementar o algoritmo *Viola-Jones* para detecção facial.
3. Implementar três algoritmos de reconhecimento facial: *Eigenfaces*, *Fisherfaces* e LBPH.
4. Portar os algoritmos implementados na placa *Raspberry Pi*.
5. Realizar experimentos para comparar o desempenho dos algoritmos, levando em conta a base de faces *AT&T Database*.
6. Implementar cadastro de faces em base de dados própria.
7. Utilizar o protocolo de comunicação MQTT.

8. Monitorar, em tempo real, ambiente privado por meio de reconhecimento facial, levando em conta a base de faces cadastrada e o uso do protocolo MQTT, a fim de testar a viabilidade da abordagem proposta.

Todos os algoritmos serão codificados na linguagem de programação *Python* (versão *Python 3.5.2*). A biblioteca de visão computacional *OpenCV* será utilizada para os algoritmos de detecção e reconhecimento de faces; e para a comunicação entre os elementos de rede via protocolo MQTT será utilizada a biblioteca *Eclipse Paho*. O paradigma de programação utilizado foi O Estruturado.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados os principais conceitos e algoritmos sobre Visão Computacional voltados para o reconhecimento de faces; e *Internet das Coisas*. Estas definições são necessárias para uma boa compreensão do trabalho desenvolvido.

2.1 Visão Computacional

Visão Computacional consiste no processo de modelagem e replicação do olho humano (visão) por meio de software e hardware. Possui inúmeras aplicações como a identificação de doenças médicas em raios-x, identificação de produtos e onde comprá-los, reconhecimento facial, reconhecimento de código de barras, entre outros. Pode ser utilizada também nas mídias sociais, encontrando imagens relevantes que não podem ser descobertas por meio de pesquisas tradicionais (ACADEMY, 2017). As subseções seguintes, apresentarão os principais conceitos que cercam o Reconhecimento de Faces, ramo integrante da Visão Computacional, definindo, primeiramente, o que é uma imagem e descrevendo as etapas do reconhecimento facial e os principais métodos que são utilizados pelas mesmas.

2.1.1 Imagem Digital

Pixels são considerados blocos de construção de uma imagem. Considerando uma imagem de 300x300 como uma grade bidimensional, onde cada quadrado constituinte contém um *pixel*, a

mesma será formada por 90.000 *pixels* (300 linhas x 300 colunas). Existem duas formas de representação de *pixels*: através da escala de cinza e cor. Na escala de cinza, cada *pixel* possui um valor entre 0 e 255, no qual quanto mais próximo de zero, mais escuro e, quanto mais próximo de 255, mais claro (Figura 2.1). *Pixels* coloridos são comumente representados no espaço de cor *Red*, *Green* e *Blue* (RGB) – um valor entre 0 e 255 para cada componente de cor (ROSEBROCK, 2016).

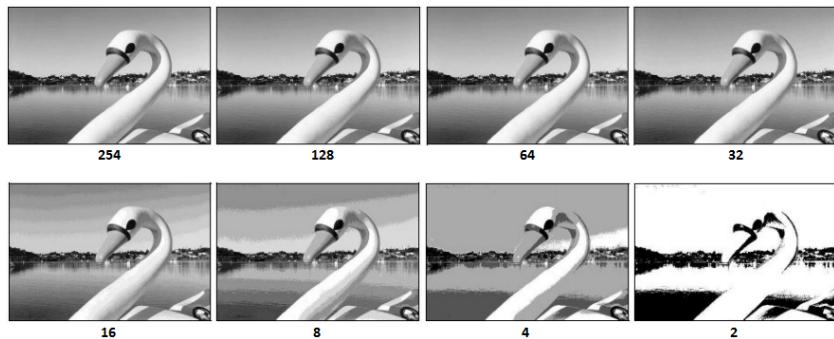


Figura 2.1: Níveis de Cinza. (JUNIOR, 2014)

Uma imagem é definida como uma função bidimensional contínua $f(x, y)$, na qual x e y são coordenadas espaciais e o valor de f em qualquer ponto (x, y) é proporcional à intensidade luminosa (brilho ou nível de cinza, valor de um *pixel* em uma escala monocromática) no ponto considerado (FILHO; NETO, 1999). Como os computadores não são capazes de processar imagens contínuas, a representação é dada através de *arrays* de números digitais, ou seja, as imagens são retratadas como arranjos bidimensionais de pontos (*pixels*).

Na Figura 2.2, a notação matricial usual que localiza um *pixel* em um arranjo de *pixels* de uma imagem *2D* é apresentada, bem como o sentido de leitura e a convenção comumente utilizada na representação espacial de uma imagem. O índice m , representa a posição da linha e o n , a posição da coluna em que o *pixel* em questão se encontra. Se a imagem digital contiver M linhas e N colunas, o índice m variará de 0 a $M - 1$, enquanto o n , variará de 0 a $N - 1$ (QUEIROZ; GOMES, 2006).

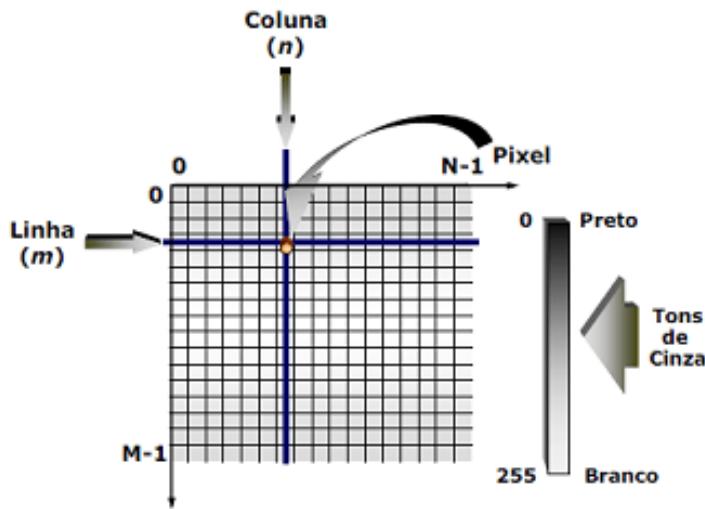


Figura 2.2: Representação de Imagem Digital 2D. (QUEIROZ; GOMES, 2006)

2.2 Detecção Facial

A detecção de face consiste no emprego de métodos computacionais que verificam a existência de uma face em uma determinada imagem digital, de vídeo ou fotografia. Apesar da detecção facial ser extremamente simples para os seres humanos, quando aplicada a sistemas computacionais, transforma-se em uma tarefa desafiadora (SORTE, 2011). Essa complexidade se dá em razão de não se saber, previamente, em que porção da imagem pode haver faces e em quais escalas elas estão; e em razão de alguns objetos ou a união deles se assemelharem a faces quando analisados em baixa resolução.

Dois fatores são levados em consideração para a avaliação do algoritmo de detecção facial: a quantidade de objetos que foram incorretamente identificados como face, chamados de falsos positivos; e a quantidade de faces que não foram identificadas, chamadas de falsos negativos. Idealmente, o algoritmo teria ambos os valores nulos (BRAGA et al., 2013).

2.2.1 Métodos de Detecção Facial

Os métodos de detecção facial são comumente classificados de três formas: Métodos baseados em conhecimento (*Knowledge-Based*), em modelos (*Template-Based*) e em aparência (*Appearance-Based*) (BRAGA et al., 2013).

Métodos que utilizam a abordagem *Knowledge-Based* buscam retratar os padrões da face

usando regras baseadas no conhecimento humano (Figura 2.3). A face humana típica tem dois olhos, um nariz e uma boca. A partir dessas características, regras são especificadas, relacionando-as em fatores como posição, distâncias relativas e contrastes. O problema dessa abordagem é, justamente, na definição de suas regras (BRAGA et al., 2013). Se o método opta por regras mais rígidas (detalhes mais específicos), ficará suscetível a falhas na detecção. Entretanto, se as regras são muito gerais, ficará suscetível há ocorrência de muitos falsos positivos.

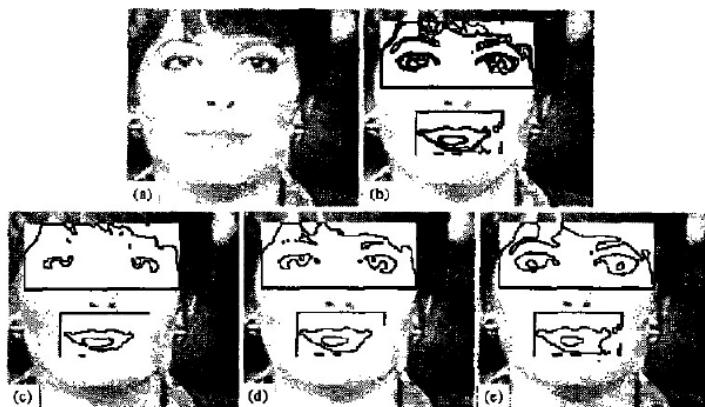


Figura 2.3: Método *Knowlegde Based*. (BRAGA et al., 2013)

No método *Template-Based*, por *default*, um padrão de faces é manualmente estabelecido ou parametrizado por meio de pontos de controle, que se modificam com o objetivo de descobrir um padrão na imagem (BRAGA et al., 2013), como é apresentado na Figura 2.4. Dada um imagem de entrada, os valores de correlação entre ela e os valores padrões previamente definidos são computados para nariz, olhos, contorno de face e boca de forma independente. A existência da face é definida pelos valores de correlação. Essa abordagem possui a vantagem de ser de simples implementação, mas é inadequada para detecção facial por não conseguir lidar efetivamente com variações na escala, pose e forma (RIZVI; AGARWAL; BEG, 2011).

A abordagem *Appearance-Based*, diferencia-se da *Template-Based* pois seus “*templates*” não são previamente definidos, sendo obtidos através de aprendizado de máquina e treinados por exemplos em imagens positivas e negativas. Para descobrir características relevantes nas imagens, geralmente, métodos baseados nessa abordagem dependem de técnicas nos campos de Análise Estatística e Aprendizado de Máquina (BRAGA et al., 2013).

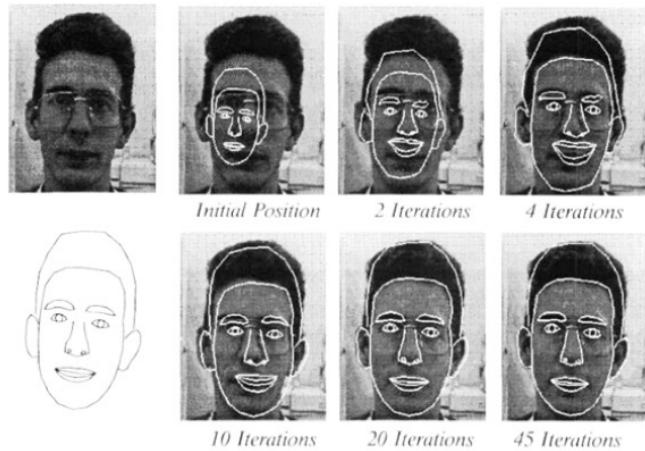


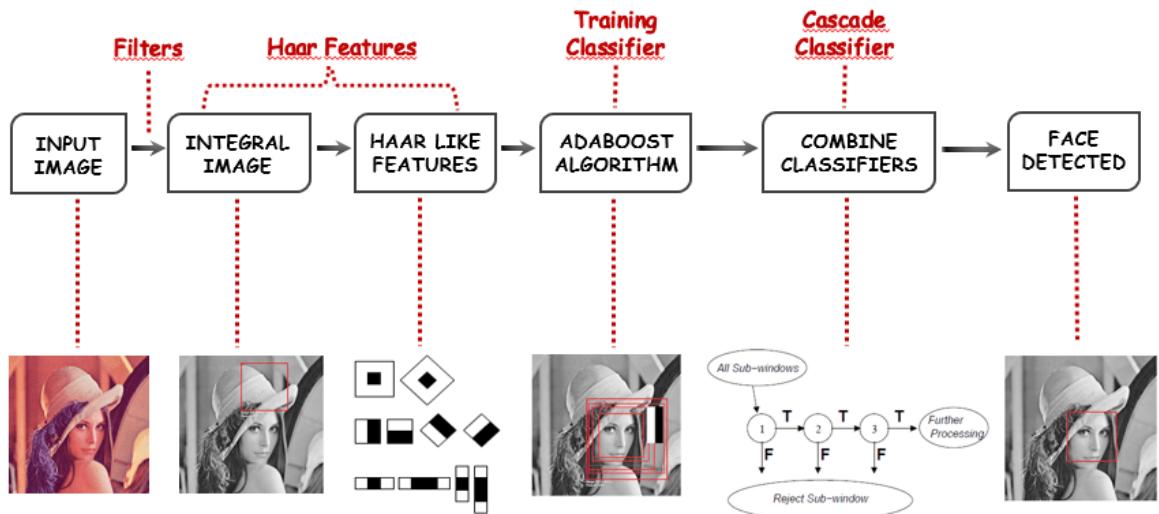
Figura 2.4: Método *Template Based*. (BRAGA et al., 2013)

2.2.2 Algoritmo *Viola-Jones*

A abordagem criada por Paul Viola e Michael Jones, conhecida por *Viola-Jones Algorithm*, para detecção visual de objetos é amplamente utilizada para detecção facial (VIOLA; JONES, 2001). Possui as seguintes vantagens: alta taxa de precisão, rapidez na execução (baixo custo computacional) e baixa taxa de falsos positivos (BRAGA et al., 2013). Este algoritmo pode ser dividido em três partes principais: *Haar like-features*, que representa a imagem de entrada em um espaço de características; *Training Classifier*, etapa onde o treinamento do sistema com imagens positivas (faces) e negativas (imagem sem faces) é realizado, tornando-o capaz de selecionar características mais relevantes na detecção; e *Cascade Classifier*, que processa de forma eficiente *sub-windows* (sub-regiões) da imagem em busca de um padrão (ARAUJO, 2010). A Figura 2.5 ilustra o funcionamento do algoritmo.

2.2.2.1 *Haar like-features*

No algoritmo *Viola-Jones*, a representação dos dados de treinamento no espaço de características é alcançado através da integral da imagem, que permite determinar eficientemente a soma dos valores dos *pixels* (níveis de cinza) de uma área retangular em uma sub-região da imagem de entrada (BRAGA et al., 2013). Um conjunto de características (*Haar-Like Features*), definido pela diferença entre a soma dos *pixels* dessas sub-regiões, é obtido por meio dessa integral. O

Figura 2.5: Etapas do Algoritmo *Viola-Jones*.

cálculo da integral da imagem em uma determinada coordenada é dado por:

$$I(x, y) = \sum_{x' \geq x, y' \geq y} f(x', y') \quad (2.1)$$

Onde $I(x, y)$ é a integral da imagem nas coordenadas do *pixel* (x, y) e $f(x, y)$ é a imagem original. Assim, é possível determinar facilmente a soma de qualquer região retangular na imagem (sub-região). Dada uma sub-região $ABCD$ de uma imagem (BRAGA et al., 2013), a soma das intensidades dos *pixels* dessa área (figura) pode ser calculada como:

$$\sum_{(x,y) \in ABCD} f(x, y) = I(A) + I(B) + I(C) + I(D) \quad (2.2)$$

O conjunto de características é calculado subtraindo-se a soma dos valores dos *pixels* da região branca, da soma dos *pixels* da região preta, representando, assim, uma diferença significativa de intensidade luminosa. Para calcular a característica A , da Figura 2.6 por exemplo, são necessárias oito integrais de imagem, respectivamente para os oito pontos indicados na figura. A característica B apresenta uma diferença de intensidade significativa entre a parte superior e inferior da sub-região, podendo ser aplicada para análise na detecção de faces, já que é comum a face humana possuir na região dos olhos uma coloração mais escura que a região da bochecha (BRAGA et al., 2013).

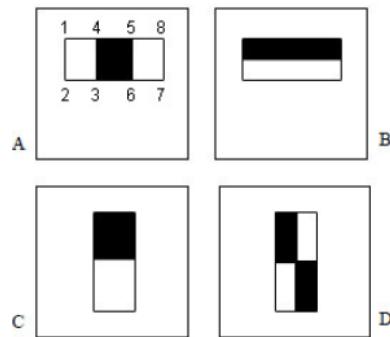


Figura 2.6: Exemplos de Sub-regiões. (BRAGA et al., 2013)

2.2.2.2 *Training Classifier*

Como a dimensão das sub-regiões torna o numero total de características maior que 180.000, faz-se necessário, visando um baixo custo computacional, a seleção de características mais significativas para detecção facial. Dessa forma, na próxima etapa do algoritmo *Viola-Jones*, o sistema é treinado com imagens positivas e negativas, utilizando o algoritmo *Adaboost*, responsável por aprender as funções de classificação (BRAGA et al., 2013). Ele é utilizado tanto na escolha das características (*Haar features*) mais adequadas, quanto para treinar classificadores com as características escolhidas (Figura 2.7).



Figura 2.7: Etapa de *Training Classifier*.

Esse algoritmo combina linearmente classificadores considerados “fracos” com o objetivo de construir um classificador de alta precisão (“forte”) como pode ser observado na equação a seguir:

$$f(x) = \sum_{t=1}^T a_t h_t(x) \quad (2.3)$$

Onde $h(x)$ representa classificadores fracos, podendo assumir valores negativos (0) e positivos (1) e x representa uma amostra da imagem (sub-região), geralmente de 24x24. O peso de cada classificador fraco é representado por a_t (BRAGA et al., 2013). O classificador forte é expresso pela função $H(x)$:

$$H(x) = \begin{cases} 1, & \text{se } pf(x) < p\theta \\ 0, & \text{caso contrário} \end{cases} \quad (2.4)$$

2.2.2.3 Training Classifier

A etapa final do algoritmo combina os classificadores na forma de uma árvore degenerada (ou cascata de classificadores), processando de forma eficiente as amostras de imagens a procura de um padrão. Em cada fase da cascata, é aplicado um classificador mais específico e complexo que o anterior, fazendo com que o algoritmo rejeite sub-regiões muito diferentes da característica procurada, encerrando o processo de busca nesse caso e prevenindo o sistema de processar desnecessariamente fases futuras. Assim, muitos estágios são descartados nas primeiras fases e apenas faces e outros objetos semelhantes são analisados de forma mais exaustiva (BRAGA et al., 2013). A Figura 2.8 apresenta a cascata de classificadores. Na detecção, não há como saber

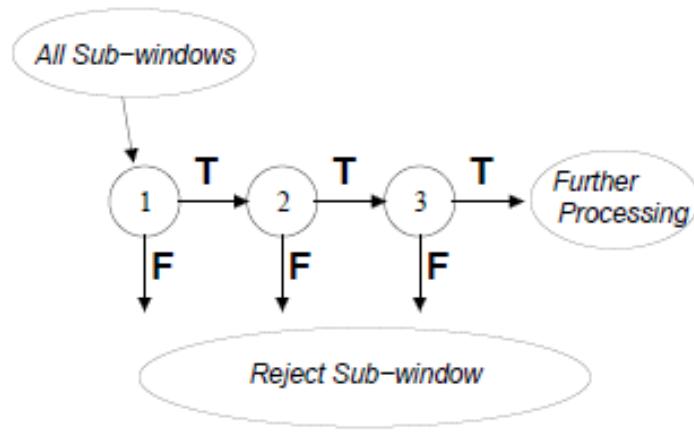


Figura 2.8: Cascata de Classificadores. (BRAGA et al., 2013)

a posição nem o tamanho da face na imagem de entrada, as características no treinamento são escalonadas do tamanho mínimo ao máximo da imagem nos respectivos estágios da árvore. Essas versões escalonadas são aplicadas em todas as amostras da imagem.

2.3 Reconhecimento Facial

O reconhecimento facial consiste na técnica biométrica de identificar características significativas em faces, como formato da boca, do rosto, a distância entre os olhos, entre outros. Humanos possuem células nervosas que tem como função responder a características locais específicas em um ambiente (cena), como linhas, arestas, ângulos ou movimento; tornando a tarefa de reconhecer faces simples. O córtex visual humano combina informações distintas de uma imagem em padrões. Os computadores utilizam a mesma ideia na automatização do reconhecimento facial, ou seja, extraem características significativas de um contexto (imagem ou vídeo) e as representam de forma útil, possibilitando sua classificação (SILVA; CINTRA, 1999). Para isso, entretanto, é necessária a realização de inúmeros processos que reconheçam padrões e determinem se uma face é conhecida ou não.

A Figura 2.9 ilustra as etapas principais do processo de reconhecimento facial, que, dada uma imagem de entrada, consiste, em: (1) a detecção facial na imagem é realizada, gerando como saída somente a sub-região onde a face foi localizada, (2) através de métodos matemáticos avançados, as características mais significativas e dominantes são extraídas da imagem, tornando a representação facial possível, (3) a base de dados faciais é consultada e é feita a verificação se a face pertence àquele conjunto, por fim, (4) é dada como desconhecida ou conhecida pelo algoritmo.

Há diversos algoritmos de reconhecimento facial na literatura. Eles utilizam conhecimentos avançados de matemática, utilizando ferramentas de Estatística e Álgebra Linear. As subseções seguintes apresentarão três dos algoritmos mais comumente utilizados, o *Eigenfaces*, o *Fisherfaces* e o *Local Binary Patterns*. Todos eles realizam o reconhecimento facial comparando determinada face com um conjunto de treinamento (*Training Set*) integrante de uma base de dados de face (ARUBAS, 2013). Os algoritmos *Eigenfaces* e *Fisherfaces* descobrem uma descrição matemática das características mais dominantes do conjunto de dados como um todo, enquanto que o *Local Binary Patterns* examina cada face contida no conjunto de treinamento de forma singular (independente e separadamente).

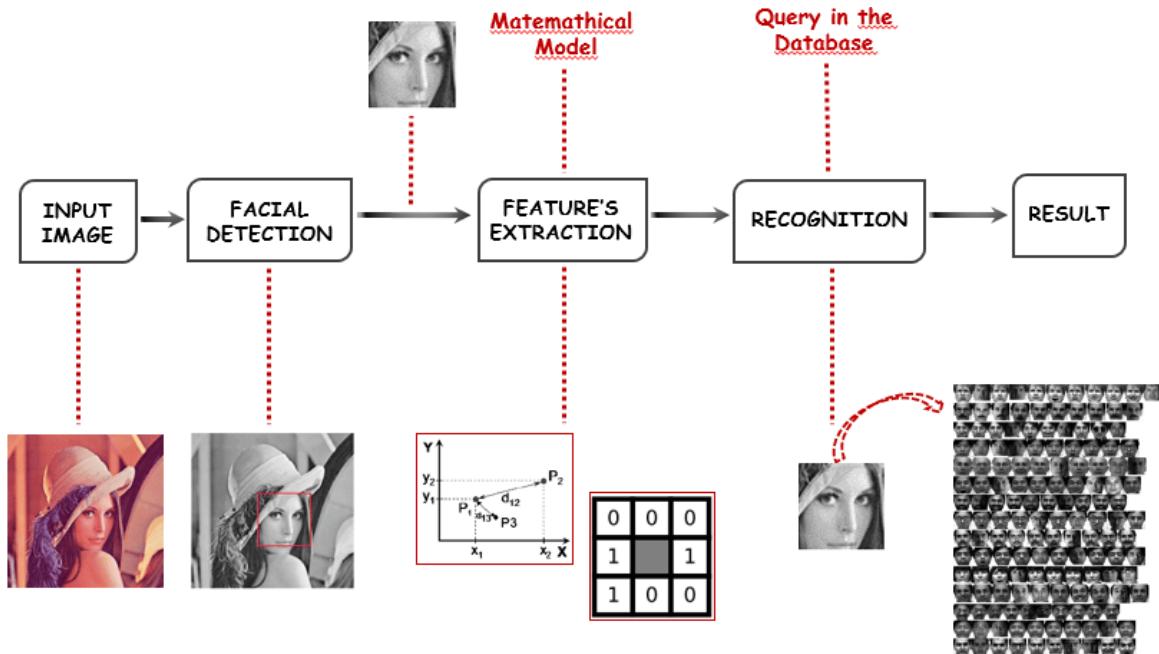


Figura 2.9: Etapas do Reconhecimento Facial.

2.3.1 *Eigenfaces*

O método *Eigenfaces* é conhecido por ter sido o primeiro algoritmo de sucesso no processo de reconhecimento de faces (OPENCV, 2012). Ele gera como saída um grupo de vetores utilizados na detecção de padrões em imagens ou vídeos. Tem como base o emprego de vetores de distribuições probabilísticas (*eigenvectors*) na geração de informações matemáticas da face humana, visando uma futura identificação (MACHADO et al., 2009).

A Análise de Componentes Principais (PCA) é um modelo matemático responsável pela etapa de extração de características. Quando aplicado pelo *Eigenfaces*, tem como função reduzir dimensões, baseada na extração de componentes marcantes que constituem um espaço multidimensional. Muito utilizada em reconhecimento de padrões, essa técnica elimina redundâncias e, ainda, consegue manter as principais características de padrão (BRAGA et al., 2013). O PCA funciona da seguinte forma: seus vetores básicos são processados a partir de um conjunto de imagens treinadas I , como o da Figura 2.10.



Figura 2.10: Conjunto de Imagens. (HANZRA, 2015)

2.3.1.1 Análise de Componentes Principais (*Principal Component Analysis*)

Primeiramente, a média da imagem I é calculada e, posteriormente, subtraída das imagens de treinamento, produzindo um grupo de amostras, de acordo com a equação:

$$i_1, i_2, \dots, \in I - \bar{I} \quad (2.5)$$

As amostras obtidas são vetorizadas em uma matriz X , possuindo uma coluna por imagem amostrada (equação 6).

$$X = \begin{bmatrix} \vdots & & \vdots \\ i_1 & \dots & i_n \\ \vdots & & \vdots \end{bmatrix} \quad (2.6)$$

XX^T é, por sua vez, a matriz de co-variância das amostras das imagens treinadas. O componente principal (PCA) é obtido através da equação:

$$R^T(XX^T)R = A \quad (2.7)$$

Onde A é a matriz diagonal dos valores dos *eigenvectors* e R é a matriz com os eles. Os *eigenvectors* são os vetores que ficam armazenados no subespaço vetorial. Geralmente, somente os N *eigenvectors* relacionados aos maiores valores definem o subespaço multidimensional (MACHADO et al., 2009).

2.3.1.2 Algoritmo de *Eigenfaces*

Para que o algoritmo de *Eigenfaces* seja computado, é necessária a utilização de uma base de dados facial capaz de armazenar os *eigenvectors*, gerados como saída do PCA, e a imagem de entrada deve conter apenas a face do indivíduo. Este cenário deve ser garantido pela detecção facial. O funcionamento do algoritmo pode ser visualizado no fluxograma da Figura 2.11.

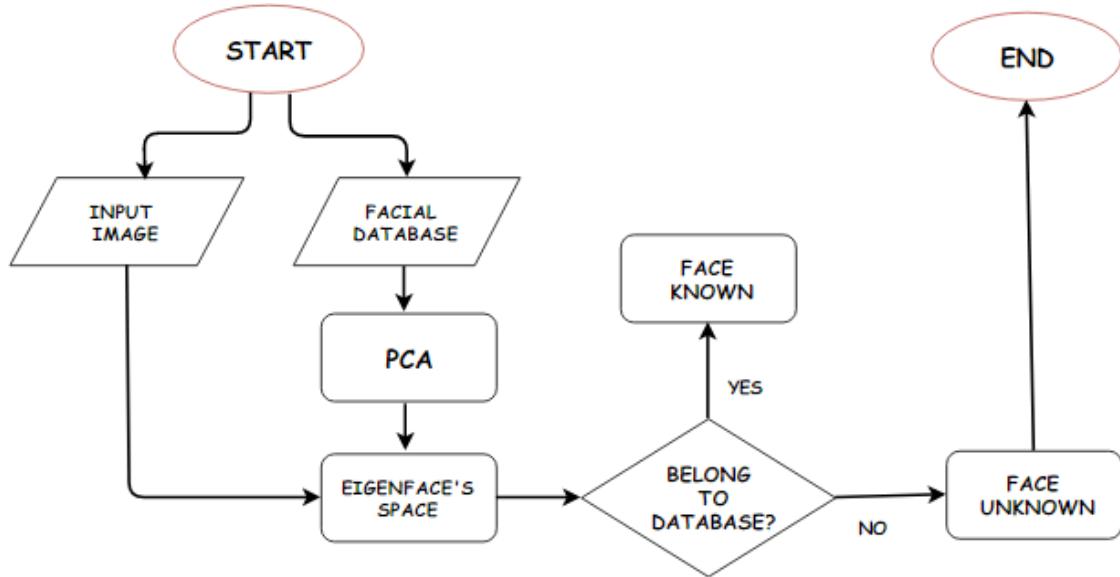


Figura 2.11: Fluxograma do Algoritmo *Eigenfaces*.

Como entrada, o algoritmo exige uma imagem para reconhecimento e uma base de dados contendo imagens para futuro treinamento. O primeiro passo do algoritmo é, justamente, a extração de características. Nessa etapa, o método PCA será aplicado nas imagens da base de imagens, fornecendo, como resultado do treinamento, vetores com as principais características dos indivíduos, armazenando-os na base de imagens e associando cada um deles a um identificador próprio. Como saída, o PCA também gera um espaço multidimensional reduzido (*Eigenface's Subspace*) em relação ao espaço original da imagem.

A próxima etapa, é a inserção da imagem de entrada no subespacial multidimensional gerado. Quando duas imagens diferentes são processadas, o subespaço assume uma representação de acordo com a figura 12 (item a). Nela, P_1 é a primeira imagem representada no subespaço, P_2 é a segunda e D_{12} é a distância entre elas no subespaço. O gráfico da imagem pode assumir várias dimensões de acordo com a quantidade de imagens cadastradas (MACHADO et al.,

2009).

A última etapa do algoritmo é a verificação se a imagem pertence ou não àquela base de imagens. Nesse processo, a imagem de entrada que foi projetada no *Eigenface's Subspace* tem sua distância para pontos já cadastrados calculada (Figura 2.12). Na figura, o ponto P_3 representa a imagem de entrada e D_{13} e D_{12} , são as distâncias entre P_3 e P_1 e P_3 e P_2 , respectivamente. Observando o item b , pode-se constatar que é mais provável que o resultado da verificação seja que a imagem de entrada é do indivíduo P_1 (MACHADO et al., 2009).

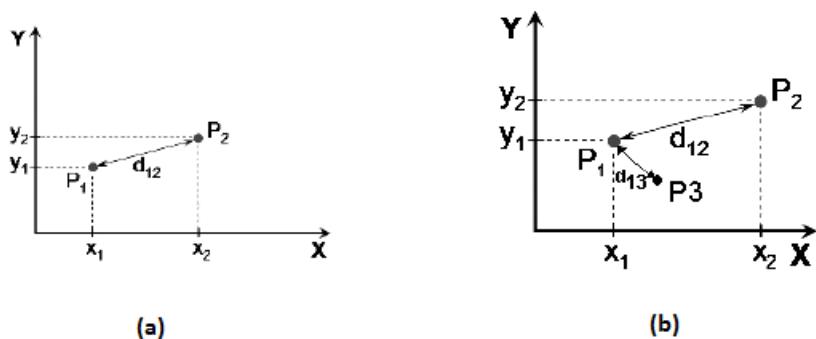


Figura 2.12: Subespaço do *Eigenfaces*. (MACHADO et al., 2009)

2.3.2 *Fisherfaces*

O método *Fisherfaces*, além de reduzir a espaço dimensional da imagem em questão, extrai as características mais discriminativas da face humana, agrupando-as em classes por meio da técnica de Análise Discriminante Linear. Esse método é bastante conhecido por possuir um bom desempenho em situações nas quais as imagens possuam variações luminosas ou variações de expressões faciais.

2.3.2.1 Análise Discriminante Linear (*Linear Discriminant Analysis*)

Cada componente da face humana possui certo poder discriminatório, permitindo a determinação de características como idade, etnia e sexo; e a diferenciação entre indivíduos. A Análise Discriminante Linear (LDA) é uma técnica estatística que tem como objetivo reduzir a dimensionalidade do espaço ao mesmo tempo que mantém as informações discriminatórias. Diferente

do método PCA, que extrai as características mais marcantes da face, o LDA escolhe o subespaço que distingue da melhor forma classes de faces, permitindo que o mesmo seja pouco sensível a variações luminosas nas imagens e bastante sensível à variação das identidades de indivíduos. Para que as classes sejam melhor determinadas e para que se consiga um melhor resultado que a técnica PCA, aconselha-se que a base de dados facial contenha diversas imagens de cada indivíduo (e.g., com diferentes expressões faciais e configurações de ambiente) (BRAGA et al., 2013).

No método LDA há dois conceitos fundamentais: a dispersão dentro da própria classe (características intrapessoais), que consiste em alterações na face da pessoa (e.g., variação na iluminação, nas expressões faciais e, até mesmo, obstrução da face em questão); e a dispersão entre faces, ou seja, a diferença na aparência de indivíduos associada a identidade (BRAGA et al., 2013). As matrizes de dispersão S_w e S_b apresentadas nas fórmulas seguintes, representam esses dois conceitos, respectivamente:

$$S_w = \sum_{j=1}^C \sum_{i=1}^{N_j} (x_i^j - \mu_j)(x_i^j - \mu_j)^T \quad (2.8)$$

$$S_b = \sum_{j=1}^C (\mu_j - \mu)(\mu_j - \mu)^T \quad (2.9)$$

Onde x_i^j representa a i-ésima imagem da classe j , μ_j representa a média da classe j , C representa o número de classes (quantidade de indivíduos diferentes na base de dados), N_j representa o número de imagens de j e μ representa a média de todas as classes. O subespaço gerado pelo LDA é formado por um conjunto de vetores discriminantes $W = [W_1 \ W_2 \ \dots \ W_n]$, que satisfaça:

$$W = argmax = \left| \frac{W^T S_b W}{W^T S_w W} \right| \quad (2.10)$$

Quando as faces são projetadas nos vetores W , as imagens que pertencem ao mesmo indivíduo (classe) ficam distribuídas de forma próxima uma da outra, esse agrupamento é conhecido como *cluster*. Esse *cluster* tem de ficar o mais longe possível do *cluster* gerado na projeção de imagens de indivíduos diferentes, obtendo-se assim uma discriminação de características

eficiente (BRAGA et al., 2013). Essa discriminação é alcançada ao se minimizar $W^T S_w W$ e maximizar $W^T S_b W$. Logo, W pode ser produzido pelos autovetores de $S_w - S_b$, conhecidos como *fisherfaces*.

A Figura 2.13 faz um comparativo das técnicas PCA e LDA. No gráfico (a), o método LDA é aplicado e as faces do mesmo indivíduo são projetadas bem próximas umas das outras (*clusters*), facilitando suas classificações de acordo com a identidade. Já na figura (b), o PCA é aplicado ao mesmo grupo de dados, resultando em faces de indivíduos diferentes próximos, tornando a classificação ineficiente.

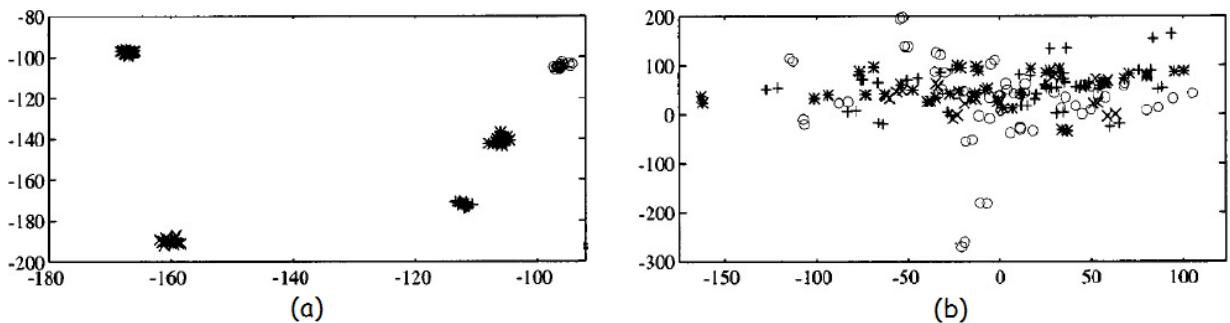
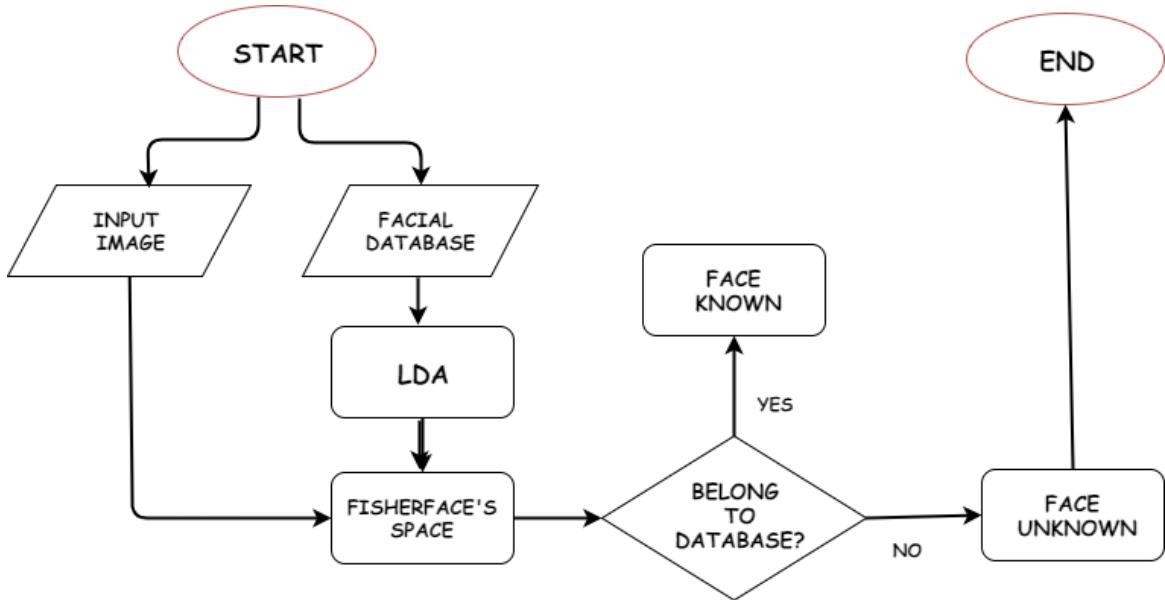


Figura 2.13: Gráficos de PCA e LDA. (BRAGA et al., 2013)

2.3.2.2 Algoritmo de *Fisherfaces*

O reconhecimento facial, dada uma imagem de entrada e uma base de dados de faces, realizado pelo o algoritmo de *Fisherfaces* é semelhante aos estágios do *Eigenfaces*, diferenciando-se apenas pelo modelo matemático empregado na extração das características dos indivíduos da base de imagens. A Figura 2.14 apresenta o algoritmo.

Nesse algoritmo, além da imagem, também é necessário uma base de dados como entrada. Através do LDA, a extração de características discriminativas das imagens é realizada e o treinamento do algoritmo com essas informações é feito. Em seguida, a imagem de entrada é projetada no subespaço dimensional reduzido (*Fisherface's Subspace*) e o reconhecimento da imagem é realizado, resultando em uma face conhecida ou não.

Figura 2.14: Fluxograma do Algoritmo *Fisherfaces*.

2.3.3 Local Binary Patterns

Os algoritmos *Eigenfaces* e *Fisherfaces* representam seus dados como vetores contidos em um espaço multidimensional, reduzindo o mesmo a um subespaço, onde as informações mais relevantes são projetadas. O *Fisherfaces* resolve o problema que é enfrentado pelo *Eigenfaces*, como variações na luminosidade e em expressões faciais, tendo um melhor desempenho. Entretanto, nem sempre é possível obter um conjunto de imagens diferentes e/ou com pouca variação luminosa de uma pessoa para alimentar a base de dados facial. Caso apenas uma imagem seja disponibilizada para cada indivíduo, os resultados seriam ineficazes.

A ideia do Algoritmo de *Local Binary Patterns* (LBP) é analisar a imagem, descrevendo-a apenas com características locais, ao invés de representá-la como um vetor multidimensional (Figura 2.15).

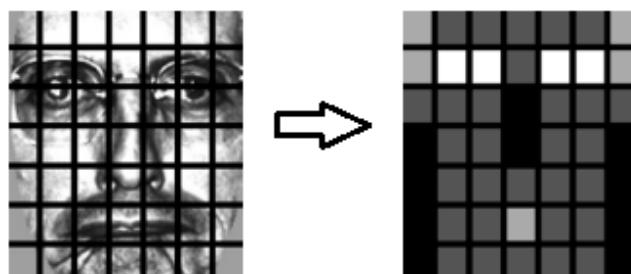


Figura 2.15: Análise LBP. (DA; SANG, 2009)

2.3.3.1 LBP Descriptor

O LBP resume a estrutura local de uma imagem comparando cada *pixel* constituinte com seus vizinhos. Tomando um *pixel* como centro, seus limites (*threshold*) serão os *pixels* próximos a ele. Caso a intensidade do *pixel* central escolhida for maior ou igual a de seu vizinho, denota-se 1 e, caso contrário, 0. No fim da análise, que pode seguir o sentido horário ou anti-horário, os vizinhos serão representados por 0s ou 1s, e o valor do *pixel* central, chamado *LBP Value* ou *LBP Descriptor*, será dado pela conversão da representação binária final da análise para um número decimal (OPENCV, 2012), como pode ser visto na Figura 2.16:

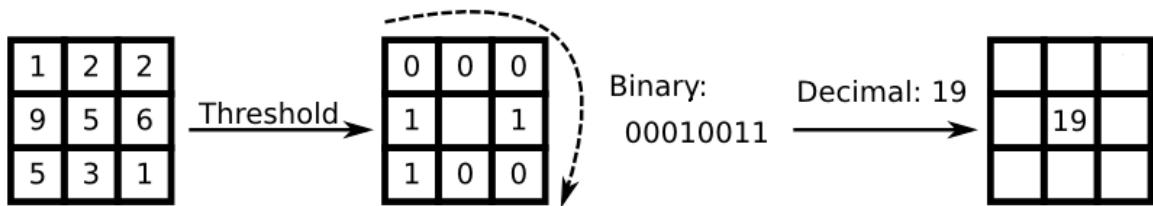


Figura 2.16: Variações de *Neighbors*. (OPENCV, 2012)

Matematicamente, o operador LBP pode ser descrito como:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^P s(i_p - i_c) \quad (2.11)$$

Onde (x_c, y_c) são as coordenadas do *pixel* central com intensidade i_c e i_p é a intensidade do *pixel* vizinho. A função que define se o valor dos *pixels* vizinhos é dada por:

$$s(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{caso contrário} \end{cases} \quad (2.12)$$

Essas fórmulas permitem que detalhes específicos na imagem sejam coletados. Para que o algoritmo não falhe na detecção de detalhes encontrados em escalas diferentes na imagem, foram criadas variações do algoritmo do LBP. Elas utilizam vizinhos dinâmicos ao invés de fixos (Figura 2.17). O objetivo é alinhar uma quantidade arbitrária de vizinhos em um círculo com diâmetro variável (OPENCV, 2012), permitindo a captura dos seguintes tipos de vizinhos:

Dados um vizinho (x_p, y_p) , com p pertencente a P , R representado o raio do círculo e P o

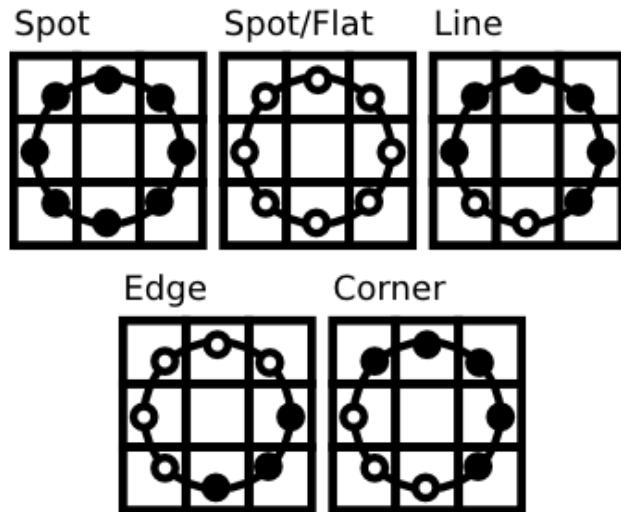


Figura 2.17: Circular Neighbors. (OPENCV, 2012)

número de vizinhos, um ponto central (x_c, y_c) e pode ser calculado por:

$$x_p = x_c + R \cos\left(\frac{2p\pi}{P}\right) \quad (2.13)$$

$$y_p = y_c - R \sin\left(\frac{2p\pi}{P}\right) \quad (2.14)$$

Por definição o valor do LBP não é sensível a variação de escala de nível de cinza, como pode ser visto na figura:

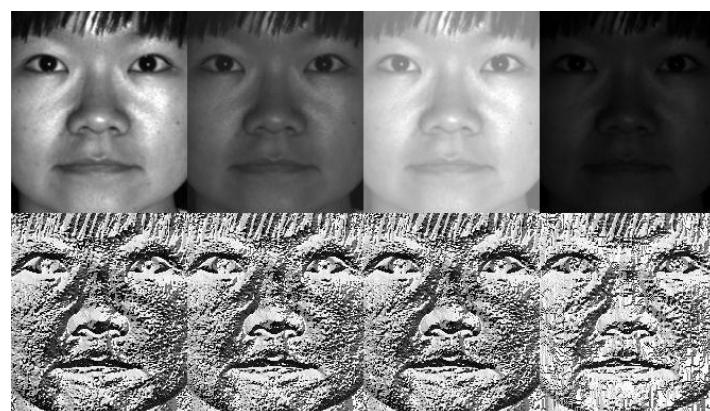


Figura 2.18: Variações nos níveis de cinza. (OPENCV, 2012)

2.3.3.2 Algoritmo do *LBP*

O LBP, assim como os outros algoritmos descritos, recebe como entrada uma imagem para reconhecimento e uma base de dados facial ou, até mesmo, somente uma imagem para comparação, diferenciando-se dos demais algoritmos por este último fator. As etapas desse método podem ser vistas na Figura 2.19.

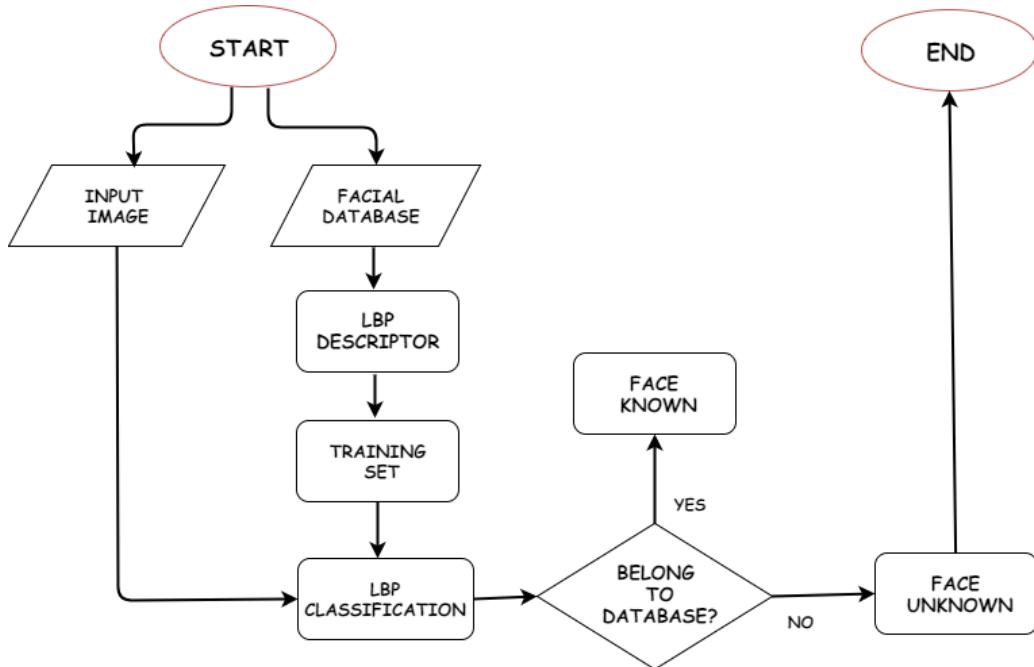


Figura 2.19: Fluxograma do Algoritmo LBP.

Primeiramente, deve-se disponibilizar a imagem de entrada e uma base de dados. A base de dados é treinada pelo descriptor LBP, resultando em uma classificação LBP, que permitirá a comparação da imagem de entrada com as cadastradas. Assim, a imagem de entrada passa pelo classificador LBP, onde é extraída as características locais da mesma, e o reconhecimento facial é realizado, resultando em uma face conhecida ou não.

2.3.4 *OpenCV*

A biblioteca *OpenCV* desenvolvida pela *Intel* (2000), é voltada para a implementação de aplicações no campo de Visão Computacional. É gratuita e aberta, possuindo módulos de Processamento de Imagens e Vídeos I/O, Estrutura de Dados, Álgebra Linear, Interface Gráfica de

Usuário (GUI), controle de mouse e teclado, além de disponibilizar mais de 350 algoritmos de visão computacional. Sua codificação pode ser feita em linguagens de programação famosas, como *C*, *C++*, *Java* e *Python*. Algumas de suas principais funções são: *tracking* de objetos, detecção de bordas, detecção de faces e padrões, algoritmos de treinamento de padrões e filtros variados (e.g., cores, ruídos, desenhos em imagens) (MACHADO et al., 2009). Neste trabalho, essa biblioteca será utilizada na implementação dos algoritmos de detecção e reconhecimento facial.

2.4 Internet das Coisas (*Internet of Things*)

A *Internet* das Coisas (*IoT*) é um paradigma inovador que vem crescendo rapidamente no cenário das redes sem fio modernas de Telecomunicações. Baseia-se no conceito da presença pervasiva (sem serem notados) de uma diversidade de coisas “*things*” ou objetos, que cercam a população e estão conectados entre si, como sensores, atuadores, *smartphones*, entre outros. O maior impacto que esta tecnologia vem causando são as modificações no dia-a-dia das pessoas, como por exemplo carros e, até mesmo, cafeteiras, que se comunicam com seus donos de forma remota. Na perspectiva de usuários empresariais, os avanços obtidos poderão ser aplicados em campos como logística, automação, fabricação industrial, transporte inteligente de pessoas, monitoramento de ambientes, entre outros.

2.4.1 Sistemas Embarcados

Sistemas embarcados são sistemas de computadores (*hardware* e *software*) utilizados no controle de diversos tipos de sistemas (e.g., hidráulicos, elétricos, mecânicos, pneumáticos). Por possuírem tamanhos compactos, design simplificado e baixo custo, são populares e extremamente empregados nas áreas de robótica, medicina, comunicação e entretenimento. Eles são baseados em microcontroladores, alguns utilizam microprocessadores (mais complexos) e *chips* para processamento dedicado. São classificados em quatro tipos: Computação Geral (*vídeo-games*), Sistemas de Controle (e.g., controles veiculares, de voo, de linhas de produção), Processamento

de Sinais (e.g., radares, sonares, analisadores de espectro) e Comunicação/Rede (e.g., telefones celulares, roteadores, *modems* de *internet*) (REIS, 2015).

Alguns sistemas embarcados não possuem interface, enquanto outros apresentam interfaces gráficas complexas, embarcando até mesmo sistemas operacionais (*System on a Chip*) para a interação com o usuário (REIS, 2015). Em alguns casos, a interface do embarcado consiste de apenas alguns botões, *LEDs*, sinais sonoros e *LCDs*. Os sistemas embarcados também podem, na maioria dos casos, ser acessados, por meio de protocolos de comunicação (e.g., *Ethernet*, *USB*, *RS-232*, *I₂C*).

2.4.1.1 *Raspberry Pi*

O *Raspberry Pi* (Figura 2.20) é um computador em uma única placa (*Single Board Computer*). Baseado no processador *ARM*, embarca sistemas operacionais (*System on a Chip - SoC*) e possui baixíssimo custo quando comparado a computadores tradicionais. É capaz de realizar grande parte do que é esperado de *desktops* (e.g., navegação na internet, execução de vídeos de alta resolução e *games*), além de ter a habilidade de interagir com o mundo externo, podendo realizar os mais variados tipos de controle, como o acionamento de atuadores e a leitura de sensores. Sua programação é feita, mais comumente, nas linguagens de programação *C*, *C++*, *Java* e *Python* (SILVA; SILVA, 2014). Seus principais componentes são: processador *ARM* *BROADCOM*, *GPIO* (Entrada/Saída), saídas de áudio e vídeo (RCA e HDMI), *slot* de *SD card* (cartão de memória), alimentação via *micro USB*, conector *CSI*, conector *DST display*, portas *USBs*, *leds* de *status* e saída *Ethernet*.



Figura 2.20: *Raspberry Pi*. (SILVA; SILVA, 2014)

2.4.2 Protocolo de comunicação MQTT

O MQTT (*Message Queuing Telemetry Transport*) é um protocolo aberto de comunicação entre máquinas (*machine-to-machine*). Ele apresenta um arquitetura simples e leve, voltada principalmente para sistemas de supervisão e coleta de dados. É baseado no protocolo de comunicação *TCP/IP*, sendo largamente utilizado em sistemas embarcados. Seu padrão de troca de mensagens é o *Publish/Subscribe (Pub/Sub)* e fazem parte de sua topologia de rede os elementos *MQTT-Client* e *MQTT-Broker* (BARROS, 2015).

A Figura 2.21 apresenta a abordagem de comunicação remota escolhida. Neste padrão, quando um *Client* visa conseguir uma informação, ele a subscreve, realizando uma requisição para um outro elemento da rede que possui a habilidade de gerenciar as publicações e subscrições. Este último, intermediário no processo de comunicação, é conhecido como *Broker*. Elementos que desejam publicar alguma informação também utilizam o *Broker*, enviando-lhe as informações que possuem. Esta topologia de comunicação permite que linguagens de programação diferentes sejam utilizadas para programação dos elementos de rede (BARROS, 2015).

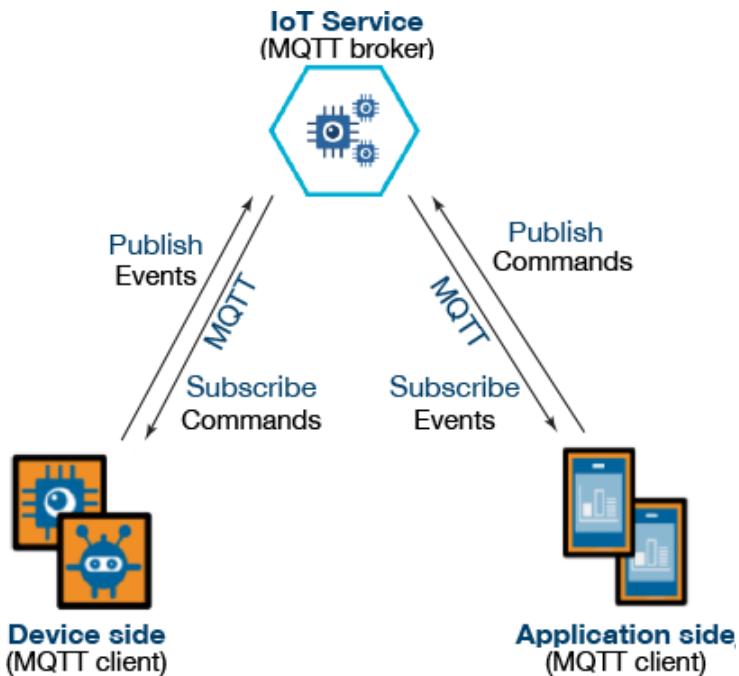


Figura 2.21: Visão Geral do Protocolo MQTT (BARROS, 2015).

No MQTT, a identificação das mensagens é feita por meio de tópicos (*topics*). Os elementos da rede podem publicar tópicos para o *Broker* ou selecionar os tópicos que almejam subscrever.

A seguir, alguns exemplos de tópicos que poderiam ser usados em uma residência:

- /casa/sensor/temperatura
- /casa/sensor/pressao
- /casa/sensor/umidade

Temperatura, pressão e umidade, no exemplo, são valores que integram os dados da mensagem. Como o MQTT não restringe o formato das mensagens, seria possível formatos *json*, inteiro ou, até mesmo, valores binários de 16 *bits*.

2.5 Trabalhos Relacionados

Os avanços nas técnicas de reconhecimento de padrões faciais possibilitaram o desenvolvimento de várias aplicações que são largamente utilizadas atualmente, como por exemplo sistemas que validam a presença de pessoas autorizadas por meio do reconhecimento de face, preservando a segurança de bancos, teatros, órgãos privados e públicos, entre outros.

Neste cenário, o trabalho de Patil e Shukla [2014] apresenta um sistema de reconhecimento facial em tempo real que substitui a folha de presença, responsável por conferir se o aluno compareceu a aula ou não. Os autores implementaram os algoritmos de reconhecimento facial na placa *Raspberry Pi*, reduzindo custos, aumentando a portabilidade do sistema e solucionando o problema da perda de tempo ocasionada pela folha de presença.

O trabalho desenvolvido por Senthilkumar et al. [2014] consiste em um sistema de captura de imagens por meio do kit de desenvolvimento *Raspberry Pi*. Seus experimentos mostraram que a arquitetura deste embarcado é rápida o bastante para suportar capturas de imagem e o processamento de algoritmos de reconhecimento; além do *stream* de dados poder circular suavemente entre o módulo da câmera e o kit.

Braga et al. [2013] apresenta um sistema de reconhecimento facial automatizado, que tem uma imagem como *input* e, como *output*, uma resposta de face conhecida ou não quando comparada as imagens cadastradas em sua base de dados. A partir de seus experimentos, pode

concluir que o reconhecimento de face exige algoritmos mais poderosos, propensos a lidar com variações, como as de luz e as de pose.

O trabalho de Kinuta et al. [2010] apresenta uma estudo comparativo dos algoritmos de reconhecimento facial *Eigenfaces*, *Fisherfaces* e o *Kernel Direct Discriminant Analysis* (KDDA). Na análise dos autores, os algoritmos são comparados em performance, na tentativa de identificar suas melhores aplicabilidades e cenários em que se sobressaem, com o objetivo de serem melhorados em variações futuras de implementações dos mesmos.

O presente trabalho se diferencia dos anteriormente citados porque propõe uma análise comparativa dos algoritmos de reconhecimento facial disponibilizados pela biblioteca de Visão Computacional *OpenCV*: *Eigenfaces*, *Fisherfaces* e LBP; e por propor uma abordagem de controle de acesso a ambientes privados na plataforma *Raspberry Pi*.

Capítulo 3

Solução Proposta

Neste capítulo, a solução proposta de monitoramento de ambientes privados será detalhada. Serão apresentados os módulos que a constituem, como serão implementadas as etapas do reconhecimento facial e como os elementos de rede do sistema se comunicarão.

3.1 Sistema de Monitoramento de Ambientes

O sistema de monitoramento foi dividido em dois módulos. O primeiro módulo, ficará responsável pela captura e pelo processamento dos algoritmos de Visão Computacional e, o segundo módulo, pelo acesso e envio de informações à nuvem e pela conectividade de dispositivos em rede (*smartphone* e a placa *Raspberry Pi*). A Figura 3.1 ilustra como o sistema funcionará.

Essa configuração passa pelos seguintes estágios:

1. O primeiro estágio consiste na captura de imagens em tempo real feita por uma câmera, que as envia a placa *Raspberry Pi* via *USB*;
2. O processamento dos algoritmos de detecção e reconhecimento facial é feito na própria *Raspberry Pi*.
3. Nessa etapa, caso a face não esteja cadastrada na base de dados, uma notificação de indivíduo desconhecido é enviada à nuvem. Do contrário, o monitoramento continua normalmente, repetindo-se os passos anteriores;

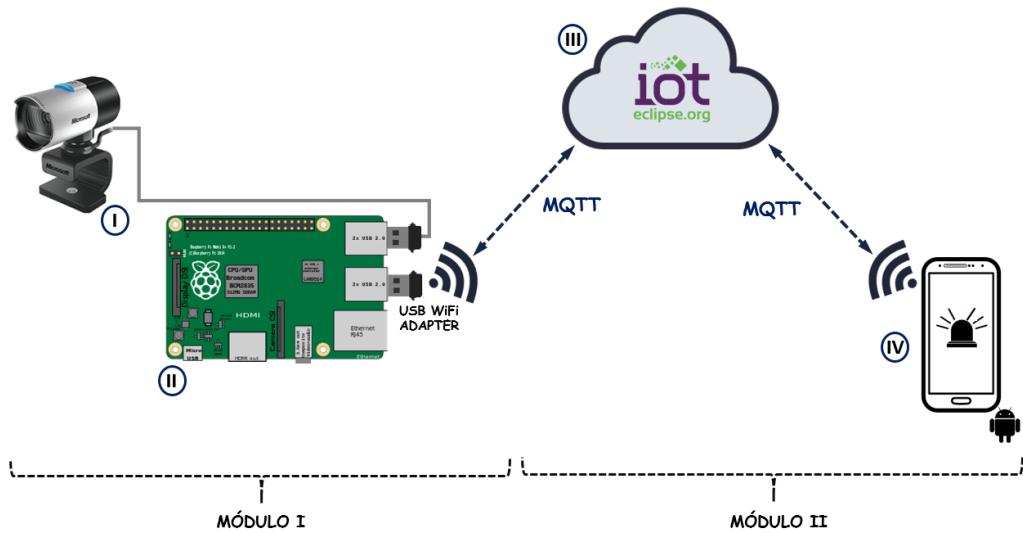


Figura 3.1: Sistema de Monitoramento Proposto.

4. Por fim, a nuvem fica encarregada de comunicar, por meio de protocolos de *IoT*, ao usuário do aplicativo de reconhecimento facial que intrusos entraram no ambiente.

O fluxograma do Sistema de Monitoramento proposto no trabalho pode ser visto na Figura 3.2. Seus estágios são: (i) primeiramente, é feita a captura de imagem, do ambiente em questão, por uma câmera USB ligada à placa *Raspberry Pi*. A fotografia dá início ao monitoramento; (ii) em seguida, são executados, pelo processador do *Raspberry*, os processos de detecção, onde é possível extraír a face do indivíduo da imagem, e o de reconhecimento, onde é possível comparar a face encontrada com as previamente cadastradas na base de imagens local. Caso a detecção falhe, o sistema pode escolher entre finalizar ou continuar o monitoramento, repetindo as etapas anteriores. Do contrário, segue para as próximas fases do reconhecimento facial. Caso estas últimas fases (extração de características e reconhecimento), tenham um resultado positivo, ou seja, a pessoa pertença de fato a base de dados, tendo permissão para entrar no ambiente, o sistema continuará com o monitoramento, recomeçando novamente ciclo. Do contrário, o sistema enviará um mensagem à nuvem (iii), informando que indivíduos não autorizados entraram no local, e esta por sua vez, a transmitirá a um aplicativo responsável por receber as informações de invasões (iv).

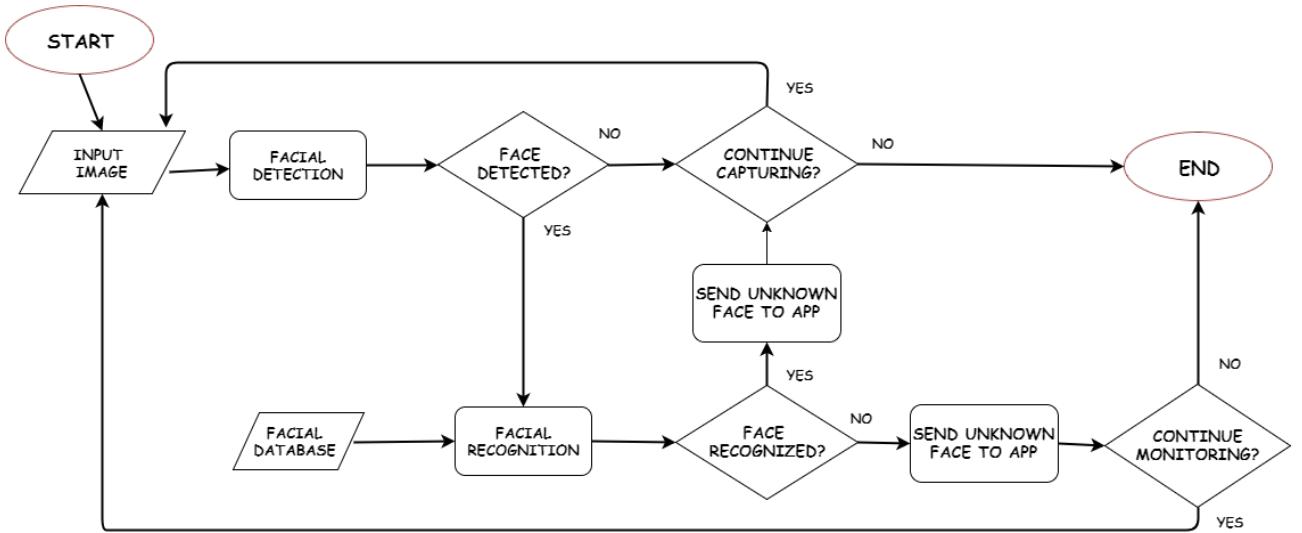


Figura 3.2: Fluxograma do Sistema de Monitoramento Proposto.

3.1.1 Captura de Fotografia e Processamento do Sistema

A placa *Raspberry Pi* é o módulo central do sistema de monitoramento. É através dela que os algoritmos de detecção e reconhecimento facial serão processados e executados. Para a captura de fotos, uma câmera USB é utilizada. Para o envio de informações à nuvem, referentes ao monitoramento do ambiente, é necessário um adaptador *wifi* USB.

3.1.2 Reconhecimento Facial

Para a implementação das etapas do reconhecimento facial, foram escolhidos os algoritmos *Viola-Jones* para detecção facial e o *Eigenfaces*, *Fisherfaces* e o LBP para as etapas de extração de características e reconhecimento; pois além de serem largamente utilizados no meio acadêmico e industrial, são disponibilizados pela biblioteca *OpenCV*. A base de dados escolhida, para o treinamento em tempo de execução dos algoritmos, foi a *AT&T Database*.

A base de dados facial escolhida para o trabalho foi a *AT&T database*, conhecida também como *ORL face database*. Ela possui 10 imagens distintas de 40 indivíduos (*subjects*). Essas imagens foram tiradas em diferentes épocas, possuindo variações de luminosidade, expressões faciais e detalhes faciais (fotos com e sem óculos por exemplo). Todas as imagens integrantes foram capturadas contra um fundo homogêneo e os indivíduos se encontravam em posição

frontal, com certa tolerância de movimentação (SAMARIA; HARTER, 1994).

Na etapa de detecção facial, foi escolhido o algoritmo de *Viola-Jones*, pertencente a classificação *Appearance-Based*. Logo, a face capturada pela câmera (imagem de entrada), é modificada por filtros que a convertem para a escala de cinza (cores monocromáticas), tornando a sua manipulação bem mais simples comparada a fotografia colorida. Em seguida, seus dados faciais são representados em um espaço característico (*Haar-Like-Features*) e suas características mais significativas são selecionadas (*Training Classifier*) e combinadas em cascata, a procura de um padrão facial. Por fim, o algoritmo resulta na localização ou não da face, dependendo do conteúdo da fotografia.

3.1.3 Envio de Notificação ao Aplicativo

A comunicação remota entre as partes constituintes do Sistema de Monitoramento (módulo de captura e processamento de algoritmos e o módulo de envio de informações ao aplicativo *Android*) é feita através do protocolo de troca de mensagens *Message Queue Telemetry Transport* (MQTT) como ilustrado na Figura 3.3. A infraestrutura de software e serviços necessários para habilitar essa comunicação é disponibilizada pelo servidor aberto *Eclipse IoT*.

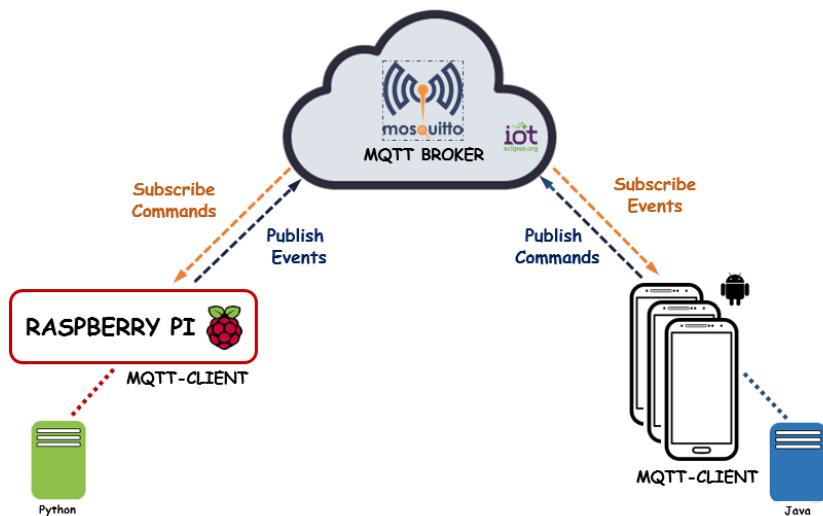


Figura 3.3: Abordagem de comunicação adotada.

Nessa configuração, os elementos de rede são as placas *Raspberry Pi*, o *Broker-Mosquitto* e os *smartphones*, que possuem o aplicativo de monitoramento. Essa abordagem pode ser utilizada

de duas formas: após a detecção de um intruso no ambiente em questão, o *Raspberry (Mqtt-Client)* publica as informações referentes a esse indivíduo no *Mqtt-Broker (Publish Events)* e este passa ao *smartphone* (outro *Mqtt-Client*), que previamente já tinha subscrito uma requisição (*Subscribe Events*), solicitando atualização do status do monitoramento, para o *broker*. Este cenário será utilizado neste trabalho. O outro cenário consiste no caso do aplicativo necessitar enviar algum comando de controle ao *Raspberry*, publicando esse comando ao *broker (Publish Commands)*, e este o passará ao *smartphone*, que terá, previamente, subscrito uma requisição (*Subscribe Commands*), solicitando atualização do status do monitoramento, para o *broker*.

O *Mqtt-broker* utilizado foi o *Eclipse Mosquitto*, pelo fato de possibilitar a transmissão de informações de forma leve e simples, além de ser gratuito e de possuir código aberto. Para que este cenário seja possível, os clientes tem que ter acesso a internet e o *broker* tem de estar hospedado em alguma nuvem (*Eclipse Iot*).

Capítulo 4

Experimentos e Resultados

Os experimentos feitos nesse trabalho tiveram como finalidade validar os algoritmos de detecção facial *Viola-Jones*, comparar os três métodos de Reconhecimento Facial disponibilizados pela biblioteca *OpenCV*: *Eigenfaces*, *Fisherfaces* e *Local Binary Patterns*, permitindo a avaliação de qual deles obtém melhores resultados quando processados na placa *Raspberry Pi*; e testar, por fim, o monitoramento por reconhecimento facial em ambiente privado. Foi utilizada a versão 3 do *OpenCV*, pois, além de possuir novas funcionalidades, dá suporte a sua codificação através de *Python 3*, linguagem de programação escolhida para o trabalho.

No monitoramento do ambiente, foram utilizadas para captura de imagens a *Webcam Logitech* com sensor HD de 1080p e, para o processamento dos algoritmos de visão computacional, a placa *Raspberry Pi 2* modelo B com frequência de 900MHz, processador *quad-core* ARM *Cortex-A7-CPU* e 1GB de RAM.

4.1 Etapa de Detecção Facial

Na etapa de detecção facial, os três métodos de reconhecimento de faces citados anteriormente utilizam o algoritmo *Viola-Jones*. Assim, para que o sucesso da etapa de extração de características e reconhecimento tivessem êxito, foram realizados testes nesse algoritmo, descobrindo-se suas vulnerabilidades e particularidades (parâmetros ajustáveis de precisão).

4.1.1 Experimentos da Detecção Facial

Na implementação da detecção facial, não foi necessário treinar um algoritmo para a etapa de *Cascade Classifier*, pois a biblioteca *OpenCV* disponibiliza esses classificadores através de um arquivo *xml* pré-treinado (ROSEBROCK, 2016). Na detecção, são considerados os seguintes parâmetros:

- ***scaleFactor***: indica quanto o tamanho da imagem é reduzido para cada escala de imagem. Esse valor é utilizado para rostos em múltiplas escalas (e.g., faces menores, em segundo plano, mais próximas da câmera (faces maiores)).
- ***minNeighbors***: indica quantos vizinhos cada janela (*sub-window*) deve ter para que a área delas seja considerada uma face. Esse valor controla quantos retângulos vizinhos precisam ser localizados para que a janela seja considerada uma face.
- ***minSize***: consiste de uma tupla de largura e altura. Determina o tamanho mínimo da janela, logo, retângulos menores que esse limite serão ignorados.

A detecção facial foi implementada, primeiramente, sem o uso da *Webcam Logitech*. Com a validação do algoritmos de detecção feita, foi implementado o monitoramento em tempo real por meio dessa câmera. O valores iniciais da função de detecção utilizados para teste foram: 1.1, 5 e a tupla (30,30) para *scaleFactor*, *minNeighbors* e *minSize*, respectivamente.

4.1.2 Resultados da Detecção Facial

A detecção facial em imagens obtidas sem o uso da câmera tiveram êxito, validando de fato o algoritmo de *Viola-Jones* e permitindo, assim, a implementação de detecção em tempo real (*webcam*). Já nesses testes iniciais, foi possível perceber que para uma detecção realmente precisa é necessário ajustar os parâmetros *scaleFactor*, *minNeighbors* e *minSize*, pois eles ajudam a discernir melhor o que é um padrão facial. A Figura 4.1 apresenta dois testes realizados com o algoritmo. Na imagem (a) a detecção é realizada com sucesso. Já na imagem (b), além da rosto, um falso positivo também foi detectado como face. Para a correta detecção desse último teste,

foi necessário ajustes em um dos parâmetros do método de detecção. O parâmetro modificado foi o *scaleFactor*, que teve seu valor de 1.2 aumentado de 1.1. para 1.2.



Figura 4.1: Testes de Detecção Facial em fotografias. (ROSEBROCK, 2016)

Os testes de monitoramento em tempo real (Figura 4.2) se mostraram bem mais sensíveis devido à exposição de luz. Nesses testes, também foram necessários ajustes nos parâmetros de detecção e foi notada a necessidade da implementação de filtros tanto de pele quanto de luminosidade.

Durante os testes foi realmente confirmado que o algoritmo *Viola-Jones* procura por padrões faciais em todas as partes da fotografia. Ele detecta bem múltiplas faces, porém se mostrou sensível a luz e a detecção de falsos positivos, como pode ser visto na figura. As fotos (a) e (c) mostram a detecção de um desenho de face, gerando a necessidade da implementação de um filtro de pele (*skin classification*); e a detecção de falsos positivos: o nariz da criança foi detectado como os olhos. Para a foto (c), ajustes nos parâmetros de detecção foram suficientes para resolver esse problema de falso positivo. Na imagem (b), é apresentado um exemplo de falha na detecção devido a luminosidade a qual a câmera foi exposta (lanterna de celular).

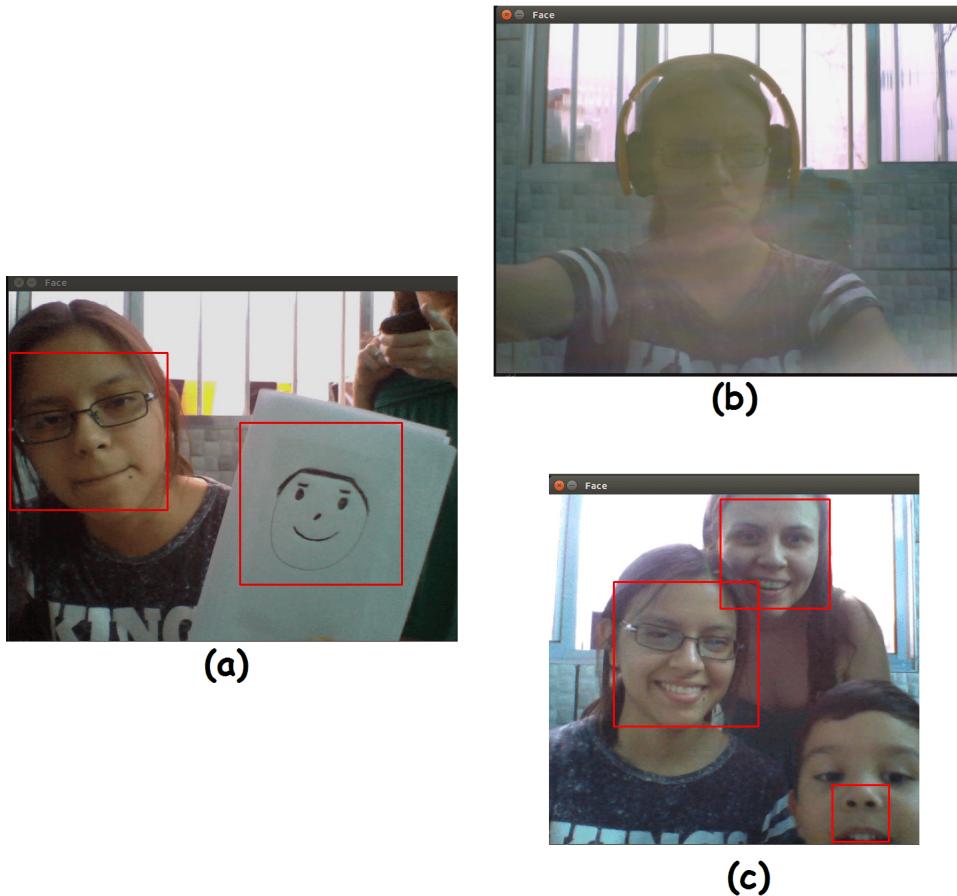


Figura 4.2: Testes de Detecção Facial em tempo real.

4.2 Etapas de Extração de Características e Reconhecimento

Realizados os testes de detecção facial, foi possível seguir com os experimentos das próximas etapas do reconhecimento de face. Para extração de padrões faciais mais dominantes foram utilizados: o PCA (*Eigenfaces*), o LDA (*Fisherfaces*) e o LBP Descriptor (*Local Binary Patterns*). Cada um dos algoritmos de reconhecimento foi executado por vez e, através das métricas de avaliação, foram analisados e comparados.

As métricas de avaliação que foram utilizadas foram as medidas de desempenho Precisão e Revogação. A precisão (P) consiste do número de elementos relevantes recuperados (E_r) dividido pelo número total de elementos recuperados (E_{tr}), como pode ser visto na equação 15. Já a revogação (R), consiste no número de elementos relevantes recuperados dividido pelo número

total de elementos relevantes existentes (E_{te}), ou seja, que deveriam ser recuperados (MATOS et al., 2009), como é apresentado na equação 16.

$$P = \frac{E_r}{E_{tr}} \quad (4.1)$$

$$R = \frac{E_r}{E_{te}} \quad (4.2)$$

No contexto deste trabalho, a precisão seria o valor resultante da quantidade de faces reconhecidas corretamente pela quantidade total de faces reconhecidas (correta ou incorretamente) de teste. A revocação, no presente contexto, seria o valor resultante da divisão da quantidade total de faces reconhecidas pelo número de faces que realmente deveriam ser reconhecidas.

Neste cenário, a pontuação ideal de precisão seria *1.0*, significando que cada resultado obtido por teste foi relevante e, a pontuação perfeita de revocação, seria também *1.0*, indicando que todos os elementos relevantes foram recuperados por teste de reconhecimento facial (MATOS et al., 2009).

4.2.1 Experimentos do Reconhecimento Facial com AT&T

Para a implementação do reconhecimento facial, foram executados os seguintes passos: (i) o primeiro passo foi alimentar o código com uma base de imagens (AT&T) e uma imagem de entrada (capturada pela *webcam*); (ii) em seguida, as faces da base de imagens foram identificadas, sendo usadas para treinar o algoritmo de reconhecimento; (iii) e, por fim, o algoritmo foi validado por meio de testes onde eram posicionadas, em frente a câmera, fotografias de integrantes e não integrantes (monitoramento em tempo real) da base de dados. Cada um dos algoritmos de reconhecimento segue essa ordem lógica de execução.

O *OpenCV* disponibiliza a criação de três tipos de objetos “reconhecedores” (*Face Recognizers*) que, dentre outras funções, permitem o treinamento da base de dados e o reconhecimento da face (HANZRA, 2015). São eles:

- *Eigenface Recognizer* - `createEigenFaceRecognizer()`;
- *Fisherface Recognizer* - `createFisherFaceRecognizer()`;

- Local Binary Patterns Recognizer - `createLBPHFaceRecognizer()`.

A Figura 4.3 apresenta alguns exemplos de reconhecimento, onde a imagem (b) apresenta o sucesso no reconhecimento do indivíduo (*subject*) número 2 da base de dados, e a imagens (a) mostra um caso em que o indivíduo não foi reconhecido como pertencente a base de dados).

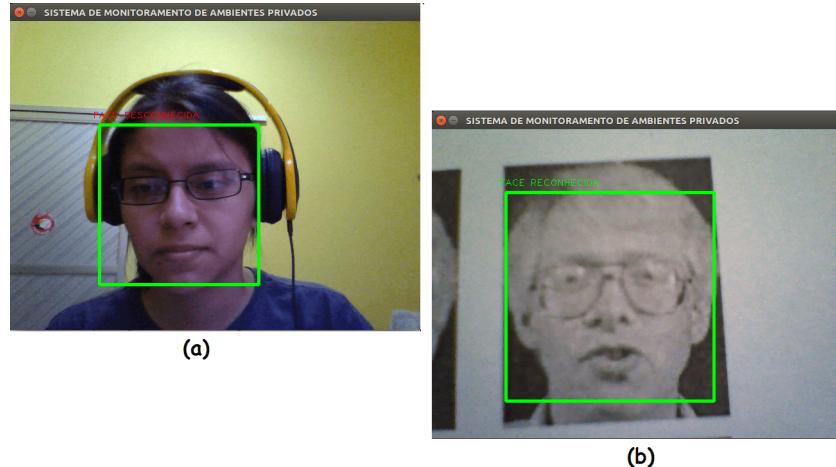


Figura 4.3: Testes de Reconhecimento Facial.

Após a implementação dos algoritmos de reconhecimento facial, a análise e comparação dos mesmos foi iniciada. Para tanto, foram impressas 4 fotografias da base de dados para serem colocadas em frente a câmera e fotos de um indivíduo que adentrou no ambiente monitorado (Figura 4.4).

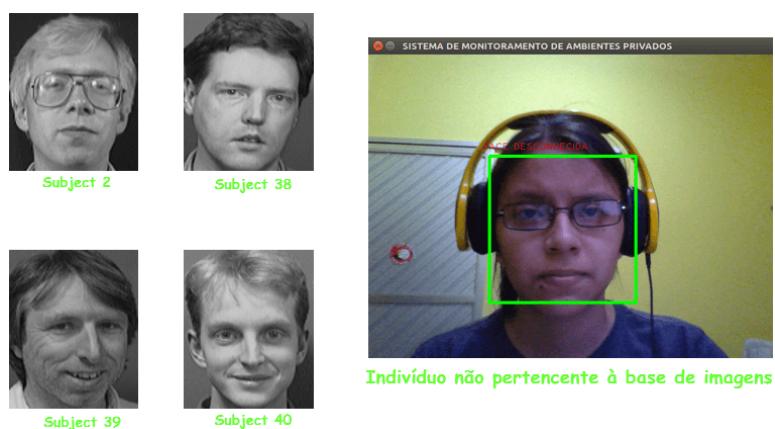


Figura 4.4: Conjunto de Imagens de teste selecionado.

4.2.2 Resultados do Reconhecimento Facial

Foram aplicados em cada um dos três algoritmos de reconhecimento, os testes com cada imagem do conjunto de imagens previamente selecionado, tornando possível a construção de Tabela 4.1 com os resultados dessa análise.

Tabela 4.1: Resultados do Reconhecimento Facial

Indivíduos	Eigenfaces	Fisherfaces	LBP
S2	Reconhecimento correto	Reconhecimento correto	Reconhecimento correto
S38	Reconhecimento incorreto	Reconhecimento correto	Reconhecimento incorreto
S40	Reconhecimento correto	Face desconhecida	Reconhecimento correto
S39	Face desconhecida	Face desconhecida	Reconhecimento correto
Externo	Face reconhecida	Face desconhecida	Face desconhecida

A partir da coleta de dados feita por meio dos testes, foi possível aplicar as métricas de Precisão e Revocação em cada um dos três algoritmos de reconhecimento facial. Nesses testes, foi possível notar quatro comportamentos diferentes nos resultados obtidos. São eles:

- **Falso positivo** - ocorre se uma face não pertencente ao banco é reconhecida;
- **Falso negativo** - ocorre se uma face pertencente ao banco não é reconhecida;
- **Verdadeiro positivo** - ocorre se uma face pertencente ao banco é reconhecida;
- **Verdadeiro negativo** - ocorre se uma face não pertencente ao banco não é reconhecida.

Neste contexto, foram calculados os valores de precisão e revocação. A tabela 4.2 apresenta o resultado dos testes realizados, apontando a quantidade detectada de falsos positivos (FP), falsos negativos (FN), verdadeiros positivos (VP) e verdadeiros negativos (VN); além de apresentar os resultados finais de precisão e revocação para cada um dos algoritmos de reconhecimento facial.

Dado um conjunto de 5 imagens teste e variações na luz do ambiente, por meio dessa tabela foi possível notar que o algoritmo LBP teve os melhores resultados de revocação, quando comparado ao *Eigenfaces* e o *Fisherfaces*, apresentando o maior valor nessa métrica (0.75). Em relação a precisão, os métodos LBPH e *Fisherfaces* apresentaram o valor máximo 1.0, pois não

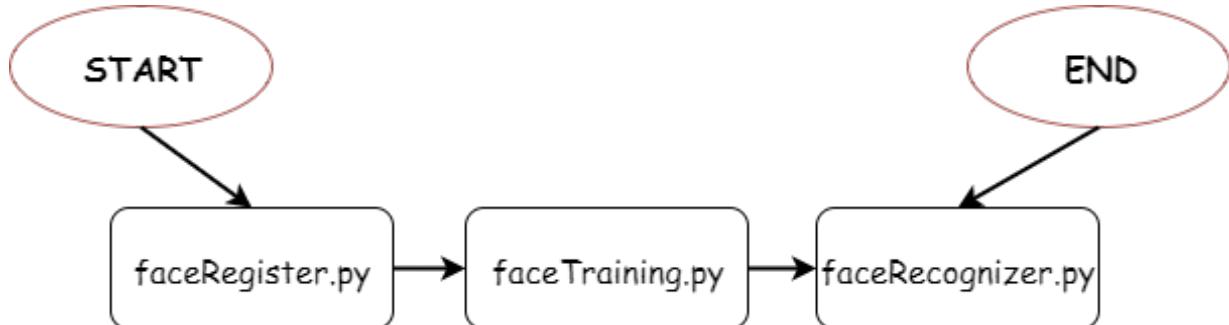
Tabela 4.2: Resultados das Métricas de Comparaçāo dos Algoritmos

Métodos de Reconhecimento Facial	FP	FN	VP	VN	Precisão	Revocação
EIGENFACES	1	2	2	1	0.67	0.5
FISHERFACES	0	2	1	2	1.0	0.5
LBP	0	1	3	1	1.0	0.75

existiram ocorrências de falsos positivos, ou seja, faces reconhecidas que não estavam no banco de imagens, ao contrário do *Eigenfaces* que se mostrou sensível a falsos positivos.

4.3 Etapa de Teste de Sistema de Monitoramento

Para que os testes de reconhecimento facial em ambiente privado fossem feitos, os *scripts* em *Python* mostrados na Figura 4.5, foram desenvolvidos, sendo executados em sequência. Os dois primeiros *scripts* são executados apenas uma vez, enquanto que o último, fica sendo executado repetidas vezes na placa *Raspberry Pi*, garantindo o monitoramento em tempo real.

Figura 4.5: *Scripts* Desenvolvidos.

4.3.1 Experimentos e Resultados do Sistema de Monitoramento

Para testar o sistema de monitoramento proposto em um ambiente real, primeiramente foi construída uma base de dados com 120 fotos de faces, sendo dose delas pertencentes a cada indivíduo, totalizando 10 pessoas cadastradas. A base de dados é local e fica armazenada na placa *Raspberry*. Para que esse cadastro fosse realizado, foi desenvolvido o *script* **faceRegister.py** que captura uma sequência de fotos. Para melhorar a precisão do reconhecimento, foi

sugerido a cada voluntário, durante o cadastro, que variasse ângulo da face, expressões faciais e retirasse ou colocasse os óculos, quando era o caso. A figura 4.6 mostra as fotos retiradas de um usuário.



Figura 4.6: Exemplo de Fotos Cadastradas por Usuário.

Após a construção da base de dados, foi implementado um *script* de treinamento das faces para o reconhecimento, possibilitando que o monitoramento de ambientes fosse testado. Para isto, a *webcam* foi posicionada em um ângulo no qual a captura da faces fosse feita de forma frontal, pois, do contrário, o reconhecimento seria ineficiente. Com o intuito de viabilizar a ideia do reconhecimento facial, foi solicitado que os voluntários, cadastrados e não cadastrados na base de dados, passassem em frente a câmera, como pode ser visto na figura 4.7.

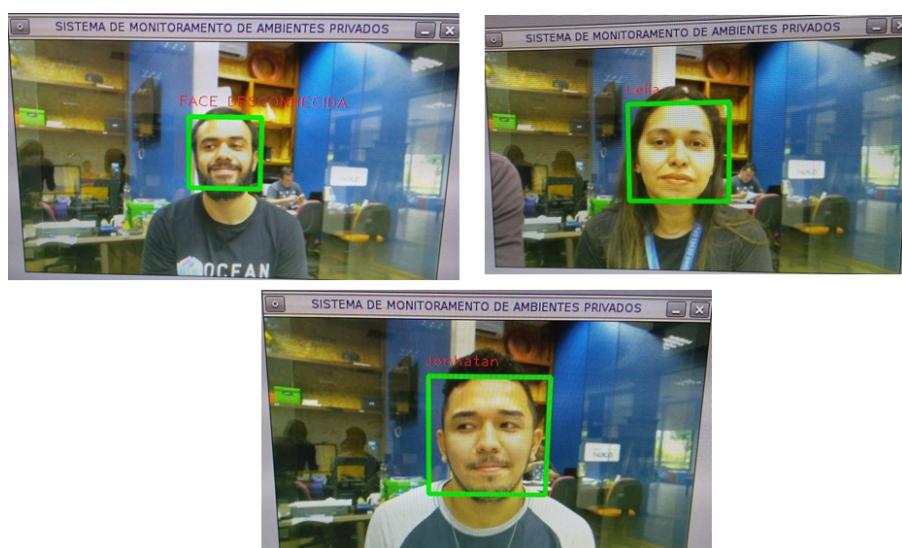


Figura 4.7: Monitoramento em Tempo Real.

Neste experimento, o *script* de reconhecimento facial ficou sendo executado na placa *Raspberry Pi* e o algoritmo de reconhecimento utilizado foi o LBP, devido aos resultados dos experimentos anteriores. Nesse *script*, além da biblioteca *OpenCV*, foi utilizada a *Eclipse Paho* para a comunicação entre os elementos de rede via MQTT. Para que o usuário pudesse acompanhar o monitoramento do ambiente, recebendo notificações em tempo real de invasão ou não, foi utilizado o aplicativo *MQTT-Dashboard*, por ser uma solução simples e gratuita. Dessa forma, qualquer usuário do ambiente privado que utilizasse o serviço *Eclipse IoT* e que inserisse “*/face/recognizer*”, tópico criado para o reconhecimento, no aplicativo poderia acompanhar o monitoramento. A Figura 4.8 ilustra a mensagem de monitoramento enviada para o aplicativo em tempo real.

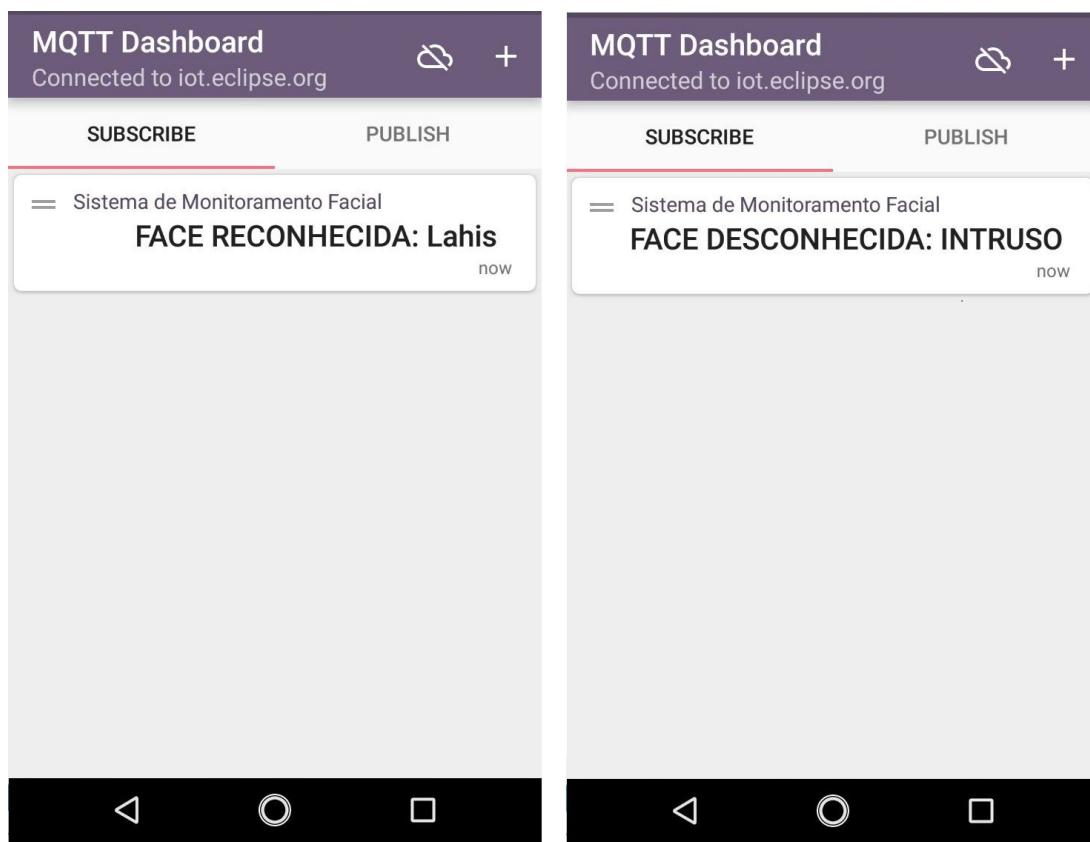


Figura 4.8: Exemplo de Fotos Cadastradas por Usuário.

Apesar da abordagem proposta ter sido viabilizada com os testes e mostrar-se próspera, quando aplicadas as métricas de avaliação os resultados ficaram aquém do esperado. Na tabela 4.3, é possível perceber que, no total de 13 voluntários, entre pertencentes e não pertencentes a

Tabela 4.3: Resultados do Monitoramento em Tempo Real

Algoritmo de Reconhecimento Facial	FP	FN	VP	VN	Precisão	Revocação
LBP	1	6	4	2	0.8	0.4

base de dados, os resultados da precisão e revocação foram, respectivamente 0.8 e 0.4. Resultados estes que representam que o algoritmo de reconhecimento facial e o de cadastro de imagens tem que ser melhorados.

Um dos fatores que ocasionaram esses resultados foi a pequena quantidade de experimentos, 13 no total, o que impacta consideravelmente os valores de precisão e revocação apresentados na tabela. Com mais voluntários e mais experimentos, esses valores poderiam ter se mostrado mais satisfatórios. Contudo, depois que essas métricas apresentarem valores melhores, o sistema de monitoramento pode ser utilizado em diversos ambientes controlando o acesso de usuários e quem sabe complementando a validação feita por outros métodos biométricos, como impressão digital e reconhecimento de íris.

Capítulo 5

Conclusão

Neste trabalho, foi apresentado um sistema de monitoramento de ambientes privados que utiliza algoritmos de reconhecimento facial e conceitos de *Internet das Coisas (IoT)*. Foi apresentado também, um estudo comparativo dos algoritmos de reconhecimento facial disponibilizados pela biblioteca *OpenCV*, com o objetivo de se escolher o melhor entre eles para ficar responsável pelo monitoramento.

Como resultado dos experimentos do reconhecimento facial com a base de dados AT&T, foi possível notar que na etapa de detecção facial, o algoritmo utilizado (*Viola-Jones*) se mostrou sensível a luz e a falsos positivos, tornando necessário implementações futuras de filtros de pele e de luz. Já nas etapas de extração de características e do reconhecimento em si, o algoritmo que teve melhor desempenho nas métricas de avaliação (precisão e revocação) foi o LBP. Já nos testes do monitoramento em tempo real houve 80% de precisão, mas um impacto ruim na revocação (40%), que significa muitos falsos negativos, ou seja, faces que estavam na base de dados mas não foram reconhecidas corretamente. Fato este que pode ser ocasionado à poucas faces cadastradas à base de dados de treinamento, sendo este o principal ponto de melhoria para refinamento da solução.

Outro resultado foi a publicação de artigo na IV Escola Regional de Informática (ERIN). Evento no qual o trabalho foi apresentado à comunidade acadêmica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas (EST-UEA) e teve boa receptividade dos participantes.

Como trabalhos futuros, serão corrigidos os erros na etapa de detecção facial, pois filtros que melhoraram sua precisão serão utilizados; e serão analisados mais algoritmos de reconhecimento facial, como por exemplo uso de redes neurais profundas (*deep learning*) (COLE, 2017), pois tal técnica vem sendo usada com muito sucesso em outras atividades relacionadas a visão computacional.

Referências Bibliográficas

- ACADEMY, D. S. *O que é Visão computacional? Principais Aplicações e Desafios Futuros.* 2017. Url = <<http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/>>. [Acessado em Junho/2017].
- ARAUJO, G. M. *Algoritmo para reconhecimento de características faciais baseado em filtros de correlação.* Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010.
- ARUBAS, E. *Face Detection and Recognition (Theory and Practice).* 2013. Url = <<http://eyalarubas.com/face-detection-and-recognition.html>>. [Acessado em Junho/2017].
- BARROS, M. *MQTT - Protocolos para IoT.* 2015. Url = <<https://www.embarcados.com.br/mqtt-protocolos-para-iot/>>. [Acessado em Junho/2017].
- BRAGA, L. F. Z. et al. *Sistemas de Reconhecimento Facial.* Tese (Doutorado) — UNIVERSIDADE DE SÃO PAULO, 2013.
- COLE, M. *Building a Facial Recognition Pipeline with Deep Learning in Tensorflow.* 2017. Url = <<https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8>>. [Acessado em Dezembro/2017].
- DA, B.; SANG, N. Local binary pattern based face recognition by estimation of facial distinctive information distribution. *Optical Engineering*, International Society for Optics and Photonics, v. 48, n. 11, p. 117203–117203, 2009.
- FILHO, O. M.; NETO, H. V. Processamento digital de imagens. *Bradspot*, 1999.
- GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Elsevier, v. 29, n. 7, p. 1645–1660, 2013.
- HANZRA, B. S. *Face Recognition using Python and OpenCV.* 2015.
- JUNIOR, J. C. S. J. Introdução do processamento de imagens. 2014.
- MACHADO, B. B. et al. Implementação de um algoritmo de reconhecimento facial usando eigenface. *e-xacta*, v. 2, n. 2, 2009.
- MATOS, P. F. et al. Um ambiente para análise de dados da doença anemia falciforme. 2009.
- OPENCV. *Face Recognition with OpenCV.* 2012. Url = <http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html>. [Acessado em Junho/2017].

- PATIL, A.; SHUKLA, M. Implementation of classroom attendance system based on face recognition in class. *International Journal of Advances in Engineering & Technology*, IAET Publishing Company, v. 7, n. 3, p. 974, 2014.
- QUEIROZ, J. E. R.; GOMES, H. M. Introdução ao processamento digital de imagens. *RITA*, v. 13, n. 2, p. 11–42, 2006.
- REIS, F. *Introdução aos Sistemas Embarcados*. 2015. Url = <<http://www.bosontreinamentos.com.br/electronica/electronica-geral/introducao-aos-sistemas-embarcados/>>. [Acessado em Junho/2017].
- RIZVI, Q. M.; AGARWAL, B. G.; BEG, R. A review on face detection methods. *Research Gate*, 2011.
- ROSEBROCK, A. Practical python and opencv. *Miami: pyimagesearch*, 2016.
- SAMARIA, F. S.; HARTER, A. C. Parameterisation of a stochastic model for human face identification. In: IEEE. *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. [S.l.], 1994. p. 138–142.
- SENTHILKUMAR, G.; GOPALAKRISHNAN, K.; KUMAR, V. S. Embedded image capturing system using raspberry pi system. *International Journal of Emerging Trends & Technology in Computer Science*, v. 3, n. 2, 2014.
- SILVA, A. L.; CINTRA, M. E. Reconhecimento de padrões faciais: Um estudo. 1999.
- SILVA, S. O.; SILVA, L. *A Linux Microkernel Based Architecture for OPENCV in the Raspberry PI Device*. 2305–1493 p. Tese (Doutorado), 2014.
- SORTE, L. X. B. Reconhecimento de faces: Aplicação de algoritmos de redes neurais. 2011.
- VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. [S.l.], 2001. v. 1, p. I–I.