CENTRO UNIVERSITÁRIO DINÂMICA DAS CATARATAS

THIAGO CORREA LIMA DA SILVA

# ESTUDO E IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE FACES EM VÍDEO COM O ALGORITMO *EIGENFACES* NA LINGUAGEM JAVA

FOZ DO IGUAÇU

2018

THIAGO CORREA LIMA DA SILVA

# ESTUDO E IMPLEMENTAÇÃO DE UM SISTEMA DE RECONHECIMENTO DE FACES EM VÍDEO COM O ALGORITMO *EIGENFACES* NA LINGUAGEM JAVA

Trabalho de conclusão de curso apresentado como requisito obrigatório para obtenção do título de Bacharel em Ciência da Computaçãodo Centro Universitário Dinâmica das Cataratas.

Orientador: Prof. Ma. Alessandra Bussador

FOZ DO IGUAÇU

2018

## SUMÁRIO

# 1 INTRODUÇÃO

O reconhecimento de face é um dos campos de pesquisa mais interessantes e importantes nas últimas duas décadas (CHAO, 2011). Grandes empresas estão na corrida para ver quem desenvolve o melhor algoritmo de reconhecimento de faces, e apenas à pouco tempo, duas delas (Google e Baidu) conseguiram taxas de erros abaixo das apresentadas por seres humanos (que se acerca aos 8%) (PORTAL, 2017).

As possibilidades de uso de um sistema de reconhecimento de faces são vastas (desde o uso comercial como empresas que querem identificar seus clientes e oferecer um serviço personalizado ao uso da polícia para o combate criminal e forense). Chega a ser um assunto polêmico em alguns círculos pois podem inferir em invasão de privacidade e privação da liberdade como a conhecemos.

As pesquisas sobre o assunto envolvem conhecimentos de disciplinas como neurociência, psicologia, computação visual, reconhecimento de padrões, processamento de imagens, matemática avançada (CHAO, 2011), por isso apresenta um grande problema computacional. Além do problema de manipulação de imagens, de identificação e reconhecimento das faces, deve-se considerar problemas de persistência destas, a questão de treinamento das faces, velocidade de verificação, de extração de dados de imagens e vídeos e desempenho em geral.

## 1.1 Breve História

O reconhecimento automático de faces é um conceito relativamente novo. Desenvolvido na década de 60, o primeiro sistema semi-automático requeria que um administrador do sistema localizasse pontos da imagem como olhos, boca, nariz, etc, e depois um algoritmos comparava as distâncias entre estes pontos em diferentes fotos para dar um resultado de comparação (SCIENCE; (NSTC), 2009). Nos anos 80, algoritmos simples foram criados para automatizar este processo. Em 1988, Kirby e Sirovich aplicaram Análise de Componente Principal, uma técnica padrão de álgebra linear, para o problema de reconhecimento de face. Este foi considerado um marco pois mostrou que menos que 100 valores são requeridos para codificar uma imagem de uma face normalizada e bem posicionada (SCIENCE; (NSTC), 2009). Em 1991, Turk e Pentland descobriu o erro residual das comparações feitas com a técnica *eigenfaces* poderiam ser usadas para detectar faces em uma imagem - uma descoberta que permitiu detecção em tempo-real. Apesar desta abordagem ser imitado a posição da face, qualidade da imagem e fatores de ambiente, ela criou um significante interesse no desenvolvimento de tecnologias automáticas para reconhecimento (SCIENCE; (NSTC), 2009). A tecnologia capturou a atenção da mídia e do público em janeiro de 2001 no evento *SuperBowl*, que capturou rostos das imagens

das câmeras de vigilância e comparou com fotos digitais "3x4"de uma base de dados apresentando rostos similares (SCIENCE; (NSTC), 2009).

Hoje, o reconhecimento de faces está sendo usado no combate à fraude de passaportes e cédulas de identidades onde a foto é padronizada com ambiente controlado, aplicativos de celulares e redes sociais para entretenimento, enquanto que grandes governos e organizações patrocinam, incentivam e desafiam empresas para conseguir o algoritmo ideal de reconhecimento (INTRONA, 2010).

## 1.2 Problemática

A face é um objeto 3D iluminado por de fontes de luz e envolvida por um fundo (*background*) com "dados arbitrários" (inclusive outras faces). Portanto, a aparência que uma face possui quando projetada para um modelo 2D pode variar tremendamente. Na verdade, problemas de iluminação e de segmentação "*foreground-background*" tem sido questões pertinentes no campo da computação gráfica e visual como um todo (JEBARA, 1995). Sendo assim o primeiro problema a enfrentar para o reconhecimento de faces seria a aquisição e processamento da imagem para que se possa detectar a face. As seções seguinte irão descrever procedimentos com mais detalhes.

### 1.2.1 Detecção e Aquisição da Face

Em geral, em uma imagem retirada de vídeo um digital de nível amador, o módulo de identificação da face deve encontrar condições luminosas descontroladas, alta variação de poses, maquiagem, mudanças nos pelos faciais, adoecimento, envelhecimento, oclusões da face por interferência, roupa ou cabelo, enfim, uma incontável gama de variáveis (JEBARA, 1995). De fato, há diversos desafios e fatores chaves que podem significantemente impactar a performance da identificação e reconhecimento da face e os pontos de verificação (*matching scores*).

Na Figura 1, lista-se exemplos de alguns destes desafios em imagens, respectivamente:

(a) Variação de iluminação;

(b) Variação de poses ou pontos de visão;

(c) Envelhecimento;

(d) Expressões faciais e estilo da face (pelos faciais ou maquiagem);

(e) Oclusão.

A detecção contínua de faces em vídeos (diz-se *tracking* ou rastreamento) segue a mesma lógica e tem os mesmo problemas de aquisições de faces estáticas. O rastreamento

FIGURA 1 – Exemplos de Variação
FONTE: *The Reseachgate.net*

nada mais é do que a detecção contínua da face em *frames* advindos de um vídeo, com um forte problema adicional de que deve-se manter a usabilidade do sistema em um computador contemporâneo de baixa a média performance. Em outras palavras, o processamento envolvido deve ser eficiente o bastante considerando o tempo de execução do vídeo (*frames* por segundo) e do sistema e tempo de armazenamento e consulta de dados.

A performance da detecção da face é uma questão-chave no processo de reconhecimento, porém considerada bem para poses de faces frontais, deste a década de 90. A detecção de faces pode ser considerada um caso específico da área de detecção de objetos, alcançando confiáveis (ELECTRICAL; IEEE, 1997).

Em um trabalho feito em 2003 por (KIM JOON HYUNG SHIM, 2003) na Universidade de Stanford - Califórnia, utilizaram a técnica *EigenFaces* para detecção, acrescido de algoritmo de identificação de gênero de sexo, com um *Pentium 3* de 700 Mhz e menos

de 500 megas de memória, alcançaram resultados que superam 93% de taxa de sucesso nas condições mais adversas, considerando iluminação e escala (numero grande de faces em uma imagem). Outros *benchmarks* mais recentes (2015) apresentam resultados que superam os 97.2% (YANG PING LUO, 2015).

O problema com estas técnicas relativamente antigas, é que também reconhecem fotos de fotos, ou desenhos de faces como legítimas faces, por vezes até com pontuação bastante para uma verificação de sucesso com uma face real. Outro problema é estas técnicas não funcionam num angulo de perfil, ou de qualquer outro ângulo que não seja o frontal.

Novas técnicas de detecção não-frontais estão sendo implementadas, bem como as de modelagem sub-espacial 3D e comparações com reconhecimento de padrões baseados em aprendizado de máquinas e redes neurais, são fundamentos para os sistemas de reconhecimento de face avançados, capazes de reconhecer não só a face, mas também a estrutura cranial do alvo (BARBU NATHAN LAY, 2015).

Após a face ser identificada e localizada na imagem, para que sirva ao processo de reconhecimento, deve-se recortar esta face da imagem e performar algumas operações gráficas sobre a mesma. Nesta fase são consideradas atividades de corte da face na região localizada, escalamento e correção de rotação, transformar em preto e branco, normalizar a imagem para minimizar as condições de ambiente da foto, deixando a face pronta para a próxima faze do processo: o treinamento.

### 1.2.2 Treinamento e Reconhecimento da Face

Esta fase consiste em manipular a face recortada da imagem ou *frame*, normalizada e tratada de forma a extrair informações e características desta para que se possa salvar de alguma forma e relacionar estas características com a pessoa ou alvo. Nesta fase é importante aquisitar fotos da mesma face mostrando diferentes expressões faciais e situações de lumiosidade e posição.

A forma que isso pode ser feito depende totalmente do algoritmo de reconhecimento envolvido. Há várias técnicas e algoritmos de reconhecimento, como (WóJCIK KONRAD GROMASZEK, 2016):

- reconhecimento baseado em redes neurais;
- reconhecimento baseado em processamento 3D;
- reconhecimento baseado em descritores de face;
- reconhecimento baseado em reconstrução;
- reconhecimentos tradicionais ou clássicos, dentre outros;

Este trabalho se focará em um algoritmo clássicos, famoso por ser pioneir e objeto de muito estudo, teste e documentação: *EigenFaces*. Os algoritmos convencionais podem ser divididos em duas categorias: caracterização holística ou linear, sendo que *EngenFaces* faze parte da primeira, que por sua vez faz parte de outra subdivisão de métodos de projeção linear chamada Análise de Componentes Principais - ACP (ou no inglês, *Principal Component Analysis* – PCA) (WóJCIK KONRAD GROMASZEK, 2016).

Basicamente, o método ACP consiste em tratar a imagem de uma forma uniforme, coletar características da mesma transformando-as em valores numéricos e disponibilizá-las em um plano cartesiano, que pode ter mais de três dimensões. Este processo visa destacar discrepâncias da face, ou seja, padrões de mudança de constaste, de relevo ou sombreamentos, ou diferença de cores. A transformação destes valores manipulados de volta em imagens, cria os chamados *EngenFaces* (Faces de fantasmas, em Alemão), pois como mostra a Figura 2, é devido a sua estranha aparência (DAVISON, 2013).

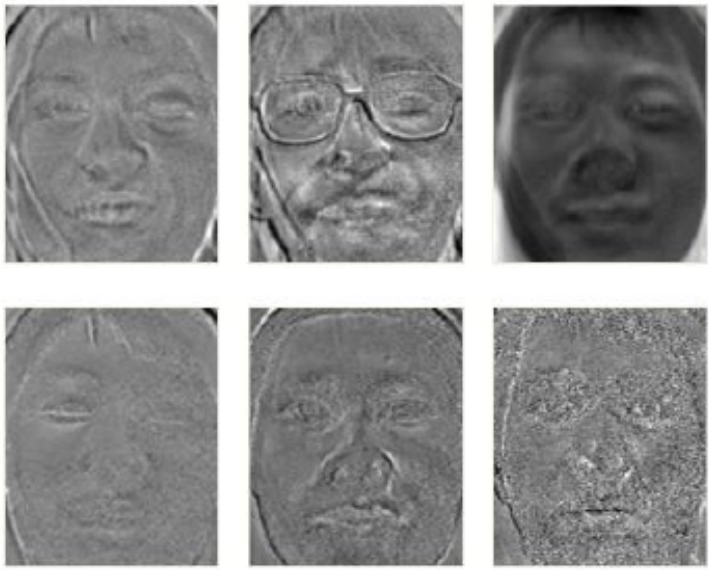

FIGURA 2 – Exemplos de EigenFaces
FONTE: (DAVISON, 2013)

O objetivo é que uma imagem de treinamento possa ser decomposta em uma soma múltiplas *eigenfaces* medidas e disponibilizadas em uma sequencia especial.

Uma maneira mais simples de representar o relacionamento entre *eigenfaces* e imagens das faces é quem estas faces são disponibilizadas em um espaço multidimensional, ou um plano cartesiano "$n$"dimensões onde os eixos do plano são as *eigenfaces* (DAVISON, 2013) como ilustra a Figura 3.

Tendo a face (ou faces) relacionada com o alvo, representado em valores e disponibilizada em um plano cartesiano de “n” dimensões, resta a verificação ou reconhecimento da face contra outra.

Para o reconhecimento da face propriamente dito, o mesmo processo do treina-

mento deve ser repetido e a face representada e disponibilizada no mesmo plano cartesiano das faces treinadas (DAVISON, 2013). A distância entre as duas faces representadas no plano cartesiano é medida, e basicamente, quando menor a distância, maior é a taxa de reconhecimento. Ou seja, uma distância zero entre as duas faces representadas no plano seria uma correspondência perfeita. A Figura 3 abaixo ilustra esta descrição:

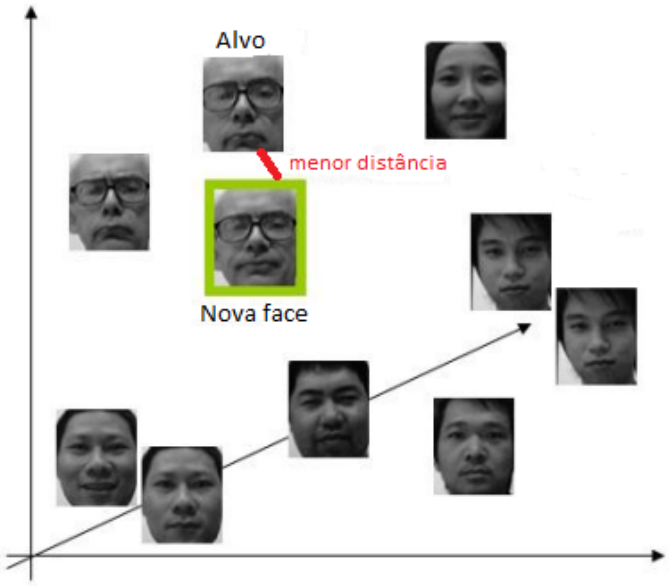


FIGURA 3 – Exemplos de EigenFaces
FONTE: Adaptado de (DAVISON, 2013)

Este plano cartesiano, na prática, pode ser um arquivo binário ou texto, comumente chamado de "*bundle*".

## 1.3 OBJETIVOS

Este projeto propõe a implementação de um Sistema de Reconhecimento de Faces em Vídeo um algoritmo desenvolvido para este fim: *EigenFaces*.

O algoritmo *EigenFaces* será implementado seguindo as instruções e exemplos de código de (DAVISON, 2013).

### 1.3.1 Objetivos Específicos

O desenvolvimento do sistema se fará na exploração das seguintes fases:

- Análise das bibliotecas e suas versões para melhor escolha;
- Preparação do ambiente;
- Implementação da interace e manipulação de imagens (e vídeos) utilizando componentes Java e as bibliotecas gráficas (para recorte de frames e imagens, aplicação de

filtros, mineração de dados da imagem, manipulação de cores, iluminação, normalização, etc);

- Implementação do algoritmo EigenFaces;
- Detecção de face em imagens tiradas com câmera;
- Treinamento / reconhecimento de face na imagem;
- Incrementar a implementação feita acima para o treinamento e reconhecimento em vídeo tirado da câmera em tempo real;
- Integrar os algoritmos de outras implementações;
- Performar uma bateria de testes, e comparar resultados.

O objetivo final é criar um sistema de reconhecimento de faces em vídeo, analisando seus resultados e finalmente disponibilizando todo o trabalho e o código aberto ao público desenvolvedor e acadêmico.

## 1.4 Justificativa

O reconhecimento de faces é umas das tecnologias mais estudadas e complexas de hoje em dia. As maiores empresas e governos do planeta estão na corrida para se acercarem aos 100% de acerto em seus reconhecimentos e governos e organizações como FBI e NSA investem e desafiam os produtores desta tecnologia (SCIENCE; (NSTC), 2009).

Eis na Tabela 1 algumas das empresas quem produzem tecnologias de reconhecimento de ponta:

As pioneiras estão avançando exponencialmente no assuntos e empresas a Google e a Baidu já conseguem taxas de erros menores do que as alcançadas por seres humanos,

TABELA 1 – Empresas que produzem tecnologias de reconhecimento de faces

| Empresas | Local | website |
|---|---|---|
| Acsys Biometrics Corp. | Burlington, Canada | http://www.acsysbiometrics.com |
| Animetrics, Inc. | Conway, NH | http://www.animetrics.com |
| Asia Software Ltd. | Almaty, Kazakhstan | http://www.asia-soft.com/frs/en/main |
| Aurora Computer Services Ltd. | Northampton, United Kingdom | http://www.facerec.com |
| Bioscrypt [Acquired by L-1 Identity Solutions in 2008; Bioscrypt acquired A4 Vision in 2007] | Toronto, Canada | http://www.bioscrypt.com |
| C-VIS Computer Vision und Automation GmbH [Acquired by Cross Match Technologies] | Palm Beach Gardens, FL | http://www.crossmatch.com |
| Carnegie Mellon University | Pittsburgh, PA | http://www.ri.cmu.edu/labs/lab_51.html |
| Cognitec Systems GmbH | Dresden, Germany | http://www.cognitec-systems.de |
| Cybula Ltd. | York, United Kingdom | http://www.cybula.com |
| Diamond Information Systems (DIS) | Beijing, China | http://www.disllc.net |
| FaceKey Corp. | San Antonio, TX | http://www.facekey.com |
| Neurotechnology [Formerly Neurotechnologija] | Vilnius, Lithuania | http://www.neurotechnology.com |
| Neven Vision [Acquired by Google in 2006; Formerly Eyematic Interfaces, Inc.] | Mountain View, CA | http://www.google.com/corporate |
| New Jersey Institute of Technology (NJIT) | Newark, NJ | http://www.cs.njit.edu/liu/facialrecognition_VPlab/index.html |
| Nivis, LLC | Atlanta, GA | http://www.nivis.com |
| Old Dominion University | Norfolk, VA | http://www.lions.odu.edu/org/vlsi/demo/vips.htm |
| OmniPerception Ltd. | Surrey, United Kingdom | http://www.omniperception.com |
| Omron | Kyoto, Japan | http://www.omron.com/r_d/coretech/vision |
| Panvista Limited | Sunderland, United Kingdom | http://www.panvista.co.uk |

FONTE: Adaptado de (SCIENCE; (NSTC), 2009)

com rapidez de verificação sem precedentes, como ilustra a Figura 4.

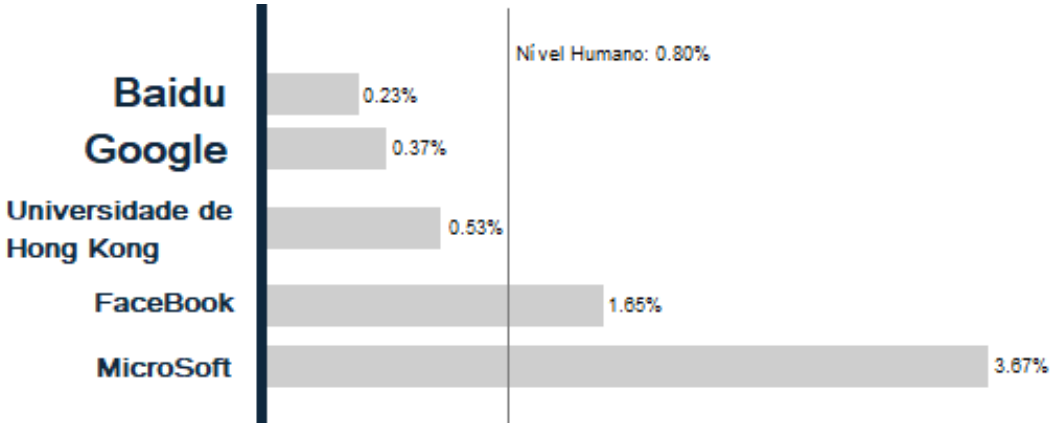


FIGURA 4 – Comparação de sistemas de reconhecimento de faces em percentagem (%) de erros

FONTE: Adaptado de (PORTAL, 2017)

A Figura 4 mostra que a tecnologia da empresa chinesa Baidu está liderando a corrida com apenas 0.23% de taxa de erro, contra 0.37% da empresa americana Google (PORTAL, 2017). Ambas já conseguiram ultrapassar as taxas alcançadas por seres humanos, que como mostra a Figura 4, atinge a taxa de 0.8% de erros. Como mostra (PORTAL, 2017), os humanos em geral não são uma fonte 100% precisa para reconhecimento de faces. Humanos podem facilmente confundir pessoas, ou não reconhecê-las pela idade ou por mudança de aparência ou estilo, ou até esquecê-las.

No final de 2016, a Baidu implantou seu sistema em sua histórica "Cidade de Água", substituindo totalmente o sistema de cartões e *tickets* com 99% de sucesso (VERGE.COM, 2016), e já estão no processo de implantação em outros parques temáticos. Policiais chineses estão usando óculos de reconhecimento que verificam instantaneamente as faces de turistas e procurados. Com os armazenamentos e reconhecimentos dominando aplicativos de celulares e redes sociais, e a polícia com grande interesse forense no assunto, e outras infinitas possibilidades, obviamente esta tecnologia fará parte do dia a dia do nosso futuro, sendo assim é necessário estuda-la, entendê-la e desenvolvê-la.

# 2 REVISÃO BIBLIOGRÁFICA

Vários trabalhos serviram como fonte de conhecimento e suporte para o entendimento e desenvolvimento deste tema. Alguns foram considerados mais importantes por abordar detalhadamente o problema de reconhecimento, outros foram buscados para se entender melhor, ou ter outra visão sobre alguns detalhes específicos do problema.

Trabalhos de autores como (SILVA, 2009) ajudaram a estruturar e organizar o tema por abordar tópicos similares e por possuir riqueza de detalhes no assunto de análise de componentes apresentando uma implementação em *MatLab* (um *software* interativo voltado para o cálculo numérico).

O trabalho de (BALAN, 2009) foi utilizado para se aprofundar no tema de imagens digitais, assim como (GONZALEZ, 2010) que também esclareceu o funcionamento de escalonamento, normalização e mono cromatização de imagens. O autor Dr. Andrew Davison em seu livro (DAVISON, 2013) se destacou pela ótima didática facilitando o entendimento de Análise de Componentes e sua relação com *EigenFaces*, disponibilizando ainda exemplos de implementação em *JAVA*.

Os trabalhos (SMITH, 2002) e (KITANI, ) foram estudados como tutoriais do processo de ACP e será usado posteriormente como guias de implementação do algoritmo.

Nas próximas seções serão explanados os conteúdos destes e de outros trabalhos disponibilizados em uma ordem didática.

## 2.1 CONCEITOS DE AQUISIÇÃO E PROCESSAMENTO DE IMAGENS E VÍDEO

Com a abrangência dos sistemas de comunicação, com difusão de conhecimentos informações pelos diversos meios, captar, armazenar e processar imagens se tornou necessidade fundamental. Para o reconhecimento de faces em vídeo, o sucesso deste processo é pré-requisito para o funcionamento dos algoritmos.

As sessões a seguir contemplam fundamentos básicos sobre imagem, vídeo, a alguns de seus processamentos necessários para a aplicação dos algoritmos de detecção e reconhecimento de faces.

### 2.1.1 Imagem Digital

A imagem digital são valores numéricos disponibilizados em uma matriz bidimensional. Basicamente existem dois tipos de imagens: os chamados *rasters* ou *bitmaps*, e vetorizadas. A primeira são representações de cada ponto da imagem com alguma cor usadas geralmente em fotografias, enquanto que o segundo é produzido por plotadores

que recebem os pontos e as distâncias entre eles como parâmetros, considerando retas, curvas, polígonos e etc, sendo assim não perdem sua qualidade quando redimensionados.

Converter uma imagem para o formato digital significa transferir os elementos que a compõem para elementos representativos de cada pequeno fragmento original. O menor elemento da imagem, o *pixel*, é identificado segundo sua intensidade de nível de cinza e as cores correspondentes. Identificados, estes elementos são armazenados por códigos que podem ser reconhecidos pelo dispositivo de visualização e apresentados novamente por um dispositivo de visualização, como um monitor de vídeo ou impressora (BALAN, 2009).

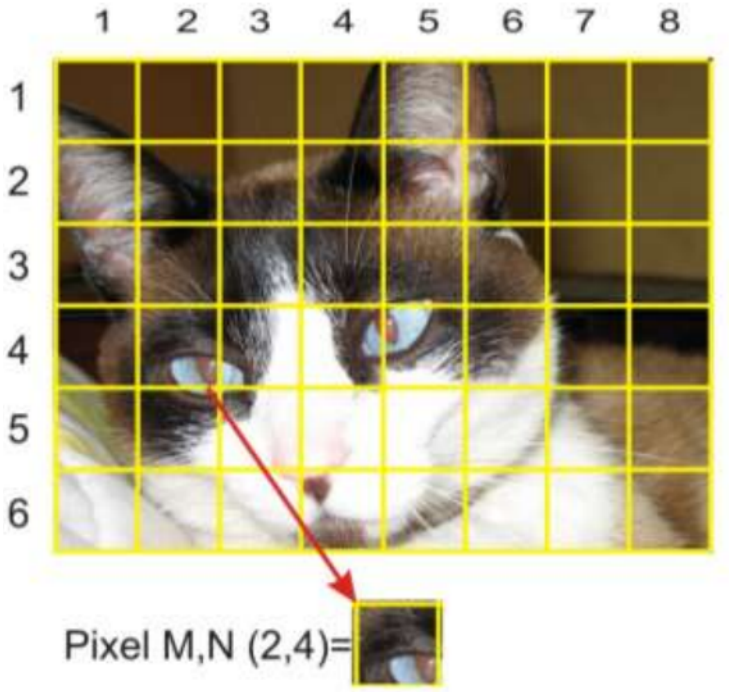


FIGURA 5 – Exemplo de imagem representada como uma matriz (M,N) de pixels, com destaque para o pixel (2,4).
FONTE: (BALAN, 2009)

Para que uma imagem analógica (representação real da cena) seja convertida, esta deve sofrer uma separação espacial (amostragem) e em amplitude (quantização) (BALAN, 2009).

A amostragem é a divisão do plano (x, y) em uma grade (ou matriz bi-dimensional) onde $x$ e $y$ serão números inteiros. Os pontos da matriz de são denominados *pixels* (*PICTure Elements*), como ilustra a Figura 5. Cada pixel representa uma parte da cena real, desta forma a resolução espacial da imagem é proporcional aos valores de M e N correspondentes na matriz (ao exemplo da Figura 5). Em geral a malha de amostragem, o formato dos pixels (x,y), é retangular, mas pode também ser triangular ou mais complexa. Os valores de cada ponto da matriz, coluna $x$ e linha $y$, que identifica um único pixel (M,N), devem ser escolhidos de forma a respeitar a relação qualidade da imagem *vs.* espaço de armazenamento, em função da aplicação para a qual a imagem se destina. Para

uma imagem digital com 256 níveis de cinza o número de bytes ocupados para armazenar a imagem é o produto da linha com a coluna da matriz (BALAN, 2009).

Matematicamente, toda imagem monocromática (preto e branco) é um função $f(x, y)$ da intensidade luminosa, em qualquer parte das coordenadas (x,y), proporcional ao brilho (tons de cinza) da imagem em um determinado ponto. A Figura 5) mostra uma imagem e como representamos os eixos x e y no plano cartesiano (GONZALEZ, 2010).

A função $f(x, y)$ é a multiplicação da iluminância i(x,y) (que é a quantidade de luz que incide sobre o objeto) pela refletância r(x,y) (que é fração de luz incidente que o objeto vai refletir ao ponto (x,y).

Sendo assim, podemos dizer que :

$f(x, y) = i(x, y) * r(x, y)$,

onde $0 < i(x, y)$ e $0 < r(x, y) < 1$ .

Quando se utiliza uma imagem colorida, no padrão RGB por exemplo, deve se usar uma função $f(x, y)$ para cada banda, R(*Red*), G(*Green*) e B(*Blue*) que são as cores primárias(GONZALEZ, 2010).

### 2.1.2 Formato PNG

O PNG (*Portable Network Graphics* ou "*PNG is Not Giff*") é um formato de representação de imagens do tipo *rasters* e foi desenhado para substituir o formato GIF e o TIFF em certa extensão. O PNG tem 3 vantagens principais: suporta canal alfa (transparência) de forma eficiente, tem a maior gama de profundidade de cores e alta compressão que pode ser regulável. Além disso, o formato é livre enquanto os outros possuem patentes (ROELOFS, 2017).

O trabalho da compressão é justamente retirar essas informações redundantes para diminuir o peso do arquivo. Por exemplo, dado um pixel qualquer de uma imagem, possivelmente, a cor ou o valor desse pixel será igual a de vários outros elementos dentro da mesma imagem. Quando a imagem é comprimida, para excluir os pixels de valores iguais, é guardado apenas um valor desse pixel, que é reproduzido para os outros semelhantes, economizando tempo de carregamento (FARIAS, 2017). Outras técnicas mais complexas também são utilizadas.

Este formato que ocupa pouco espaço em memória e de fácil manipulação foi escolhido neste trabalho para a representação de imagens digitais quando for necessário.

### 2.1.3 Vídeo Digital

Um vídeo é uma sucessão de imagens apresentadas sequêncialmente em um determinado ritmo. O olho humano pode distinguir cerca de 20 imagens por segundo. Deste

modo, quando se mostram mais de 20 imagens por segundo, é possível enganar o olho e criar a ilusão de uma imagem em movimento.

A fluidez de um vídeo se caracteriza pelo número de imagens por segundo (frequência de quadros), expresso em FPS (Frames per second - Quadros por segundo). o vídeo multimídia é geralmente acompanhado de som, ou seja, dados de áudio (SMITH, 2018).

### 2.1.4 Aquisição da Imagem e de Vídeo

No processo de aquisição de imagens, tradicionalmente, uma "cena"tridimensional (ou imagem analógica) deve ser capturada por um dispositivo eletrônico, que colhe uma amostragem convertendo-a para uma imagem digital de duas dimensões. Por exemplo, uma câmera digital captando uma cena 3D e convertêndo-a em 2D.

Atualmente o dispositivo de conversão mais usado é a câmera CCD (*charge coupled device*), que é uma matriz de células semi-condutoras fotossensíveis que trabalham com capacitadores, fazendo um armazenamento da carga elétrica proporcional à energia luminosa incidente (GONZALEZ, 2010).

A saída deste processo é a imagem do tipo *raster*, pronta pra ser formatada e compactada por alguma representação (PNG, JPG, BPM, etc).

Na aquisição de vídeos, o mesmo processo descrito acima para imagens é utilizado, com a diferença de que o vídeo é uma fluidez de imagens, sendo assim é necessário aplicar este processo iterativamente. Caso uma única imagem precise ser colhida do vídeo, basta escolher um quadro (descrito na Subseção 2.1.3) para o processo.

### 2.1.5 Processamento da Imagem

O processamento de imagem é todo o processo que tem uma imagem como entrada e saída, tais como fotografia ou quadros de vídeo (EYMAR LAéRCIO, 2018).

Usa-se para melhorar o aspecto visual de certas feições estruturais para o analista humano ou para performance e para fornecer outros subsídios para a sua interpretação, inclusive gerando produtos que possam ser posteriormente submetidos a outros processamentos (EYMAR LAéRCIO, 2018).

O processamento de imagens pode ser dividido em 3 fases básicas: pré-processamento, realce e classificação. Pré-processamento refere-se ao processamento inicial de dados brutos para calibração radiométrica da imagem, correção de distorções geométricas e remoção de ruído. Realce visa melhorar a qualidade da imagem, permitindo uma melhor discriminação dos objetos presentes na imagem. Na classificação são atribuídas classes aos objetos presentes na imagem (EYMAR LAéRCIO, 2018).

Nas próximas seções algumas técnicas de processamento de imagens serão abordadas pela sua importância e sua necessidade de uso em algoritmos de detecção e reconhe-

cimento de faces. São elas:

- Conversão em Escala de Cinza: considerada uma técnica pré-processamento, é usada para transformar a imagem em "preto e branco"(na escala de cinza) , o que torna outras operações de processamento mais rápidas e eficientes;
- Escalonamento: pré-processamento de correção linear geométrica 2D (WANGENHEIM, 2017);
- Equalização: uma técnica de realce de contraste (GONZALEZ, 2010)
- Segmentação: recorte da imagem, onde se extraem os objetos relevantes para a aplicação desejada (EYMAR LAéRCIO, 2018).
- Classificação: uso de classificadores para reconhecer padrões em imagens (DAVISON, 2013)

#### 2.1.5.1 Conversão em Escala de Cinza

Em fotografia, computação e colometria, uma imagem em escala de cinza é uma imagem em que cada valor de pixel é uma única amostra da representação de intensidade de luz (JOHNSON, 2006). Em outras palavras, o valor R, G e B de cada pixel devem ser iguais para se obter uma escala de cinza que varia de preto em sua intensidade mais baixa para o branco absoluto.

A conversão de uma cor para escala de cinza não é algo único. Existem diversas maneiras e possibilidades de inserção de parâmetros que dariam resultados diferentes. A maneira mais usual é calcular a média dos valores RGB de cada pixel e atribuir o resultado ao pixel novamente. Exemplo:

$Gray = (Red + Green + Blue)/3$, reconhecendo que o algoritmo seria algo como:

```
For Each Pixel in Image {
   Red = Pixel.Red
   Green = Pixel.Green
   Blue = Pixel.Blue
   Gray = (Red + Green + Blue) / 3
   Pixel.Red = Gray
   Pixel.Green = Gray
   Pixel.Blue = Gray
}
```

Converter a imagem em escala de é necessário para tornar eficientes os subsequente processos, como equalização e aplicação de classificadores (DAVISON, 2013). Na Figura 6 apresenta-se um exemplo desta tipo de conversão.

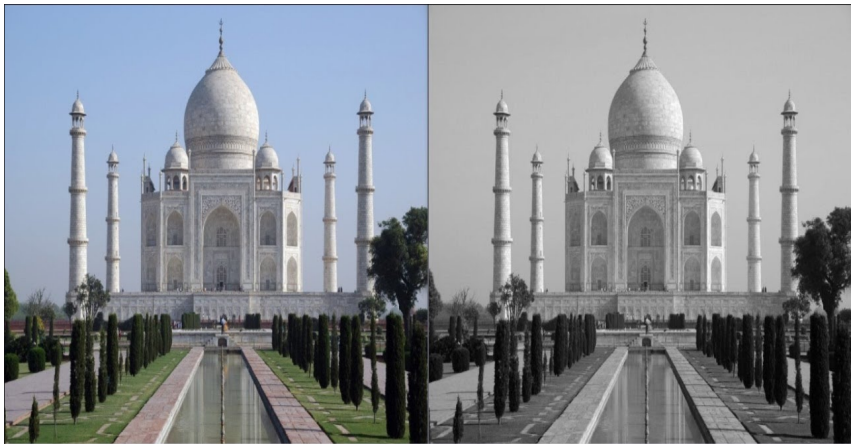

FIGURA 6 – Imagem convertida para escalada de cinza.

#### 2.1.5.2 Escalonamento

O escalonamento de imagens é uma transformação geométrica 2D. Pontos no Plano $(x, y)$ podem ser escalonados (esticados) por fatores de escala, definida como $S_x$ e $S_y$, através de multiplicação (WANGENHEIM, 2017):

$x' = x.S_x$, $y' = y.S_y$. Onde o novo ponto é resultado da multiplicação do ponto pela escala.

No exemplo da figura abaixo $S_x = 2$ e $S_y = 1$ os valores de $x$ foram escalonados em 2 ($S_x$) enquanto y foi escalonado em 1 ($S_y$). Observe o resultado a seguir.:

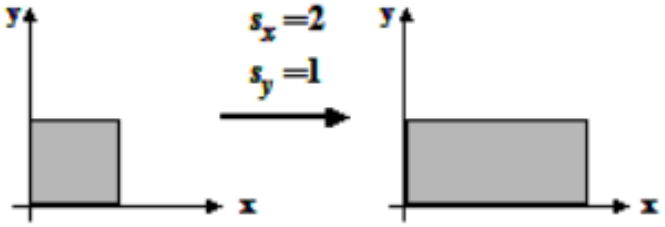


FIGURA 7 – Escalonamento com $S_x = 2$ e $S_y = 1$.
FONTE: (WANGENHEIM, 2017)

O escalonamento pode causar distorções na imagens se não for feita linearmente, e perda de qualidade caso os valores sejam alterados de forma a aumentar a imagem. É muito usado para padronização de tamanhos, e se o escalonamento for feito com a subtração de valores pequenos, a qualidade da imagem pode ser melhorada. Outra vantagem de escalonar a imagem subtraindo valores é o fato de posteriores processamentos performarem mais rapidamente, pois a imagem apresenta-se menor (DAVISON, 2013).

#### 2.1.5.3 Equalização de Histograma

O histograma de uma imagem digital é definido como "uma função discreta h(rk)=nk onde rk é o k-ésimo valor de intensidade e nk é o número de pixels da imagem com in-

tensidade rk e cujos níveis de intensidade desta imagem estejam no intervalo [0, L - 1]". Os Histogramas são a base para várias técnicas de processamento no domínio espacial, onde sua manipulação pode ser utilizada para realce de imagens, além de fornecer dados estatísticos a seu respeito (GONZALEZ, 2010).

Em outras palavras, esta técnica realça o contraste da imagem com o objetivo de melhorar a qualidade da imagem em um ponto de vista humano. Do ponto de vista computacional, a equalização examina o intervalo de valores em escala de cinza (ou outra disposição de cores, dependendo da imagem) e tonifica mais a parte que existe mudança abrupta do branco para o preto (ou cinza escuro). O resultado é uma imagem com maior contraste entre áreas sombreadas, iluminadas e contornos, tornando a detecção de padrões mais fácil posteriormente (DAVISON, 2013). A Figura 8 mostra o resultado depois da aplicação deste procedimento.

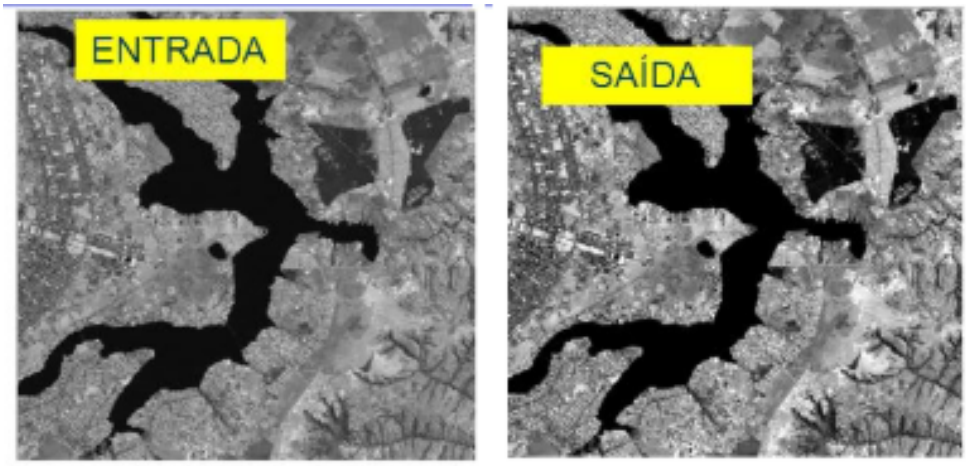


FIGURA 8 – Exemplo de aplicação de equalização de histograma.

#### 2.1.5.4 Segmentação

Em visão computacional, segmentação se refere ao processo recortar uma imagem digital, ou seja, recolher um conjunto de pixels, para simplificar, mudar a representação de uma imagem ou extrair áreas relevantes para facilitar a sua análise.

Como resultado, cada um dos pixels em uma mesma região é similar com referência a alguma característica ou propriedade computacional, tais como cor, intensidade, textura ou continuidade. Este processo é pré requisito para que reconhecimentos de objetos tenham grandes chances de sucesso (GONZALEZ, 2010).

#### 2.1.5.5 Classificação

Segundo (VELASQUEZ, 2017), classificação de imagem contextual é um tópico de reconhecimento de padrões em Visão Computacional. Chama-se "contextual"pois significa que esta abordagem é focada na proximidade com os pixels e sua relação com o "classifi-

cador", (contemplado na seção Subseção 2.2.1). É utilizado efetivamente na detecção de objetos e também pode ser aplicado para a detecção de faces em uma imagem.

## 2.2 DETECÇÃO DE FACES COM CARACTERÍSTICAS HAAR

Um classificador HAAR pode ser treinado para detectar vários tipos de objetos rígidos, definidos, como carros, motos ou partes do corpo humano como olhos ou boca, com altas taxas de sucesso. Não é muito eficiente em reconhecer objetos com ramos tipo arvores, mãos ou objetos camuflados ou contendo pouca textura, contorno e sub-regiões que variam em cor e iluminação (DAVISON, 2013).

Um classificador bem treinado pode envolver *milhares* de fotos de alta qualidade com imagens positivas. Para a detecção de faces, isto significa imagens de reostos tiradas de perto que devem ter posições similares com muito pouca variação de fundo (*background variation*). Olhos bocas e narizes devem estar na mesma posição em todas as fotos, e estas devem ser do mesmo tamanho. Também é necessário treinar o classificador com um número similar de imagens negativas (isto é, sem rostos).

Existem diversas bibliotecas com classificadores pré-treinados para diferentes objetos, incluindo faces.

### 2.2.1 As Características Haar (*Haar Features*)

Os classificadores Haar (ou cacarcterísticas de haar, ou filtro haar) recebem esse nome pela sua similaridade com a *Haar wavelet* (ou ondulação Haar), que consiste em uma sequência de funções quadradas redimensionadas que formam uma familia de ondulações das mais simples possíveis.

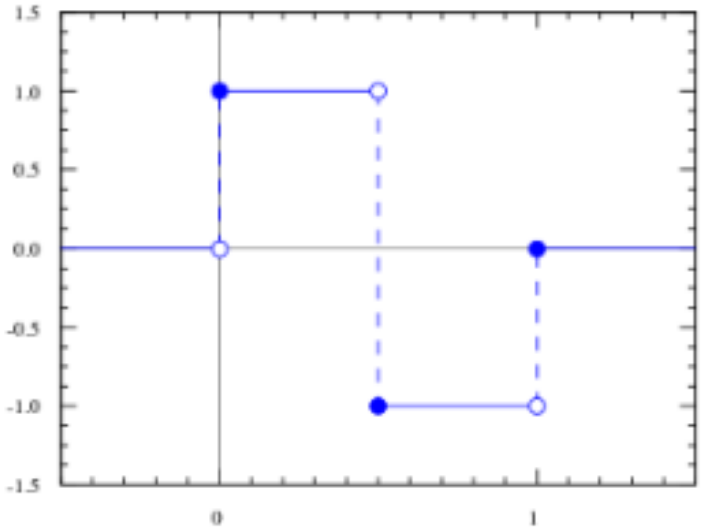


FIGURA 9 – Um exemplo de *Haar wavelet*

Características Haar são basicamente imagens treinadas e usadas para achar objetos similares em outras imagens. Viola e Jones em seu algoritmo *Haar Cascades* (contemplado na seção seguinte), por exemplo, utilizam classificadores retangulares. Cada classificador pode indicar a existência ou a ausência de uma característica em outra imagem. A maior motivação para o uso de características em um objeto ao invés do uso de pixel é que a velocidade da análise de uma imagem baseada no conjunto de sua principais características é muito maior do que a análise baseada sobre seus pixels, devido ao fato do numero de características ser substancialmente inferior em relação ao número de pixels (MOREIRA, 2008).

A Figura 10 mostra um classificador Haar em ação.

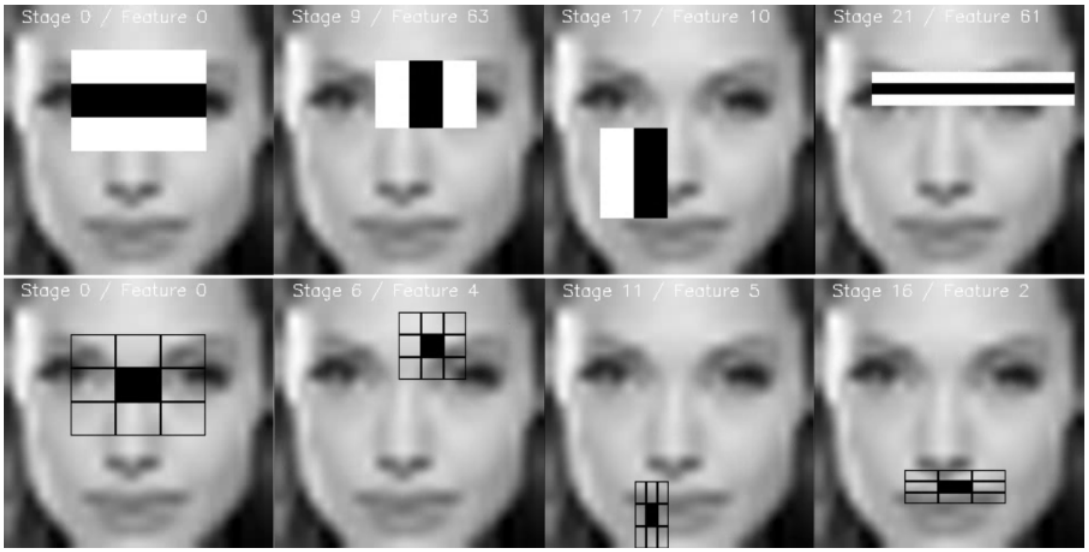


FIGURA 10 – Classificadores Haar representando características relevantes face
FONTE: Biblioteca OpenCV

Estes padrões retangulares podem ser escalonados para que caracterísricas de diferentes tamanhos na imagem possam ser detectados usando a mesma abordagem.

### 2.2.2 Algoritmo Viola-Jones (*Haar Cascades Classifier*)

O *framework* de detecção de objetos Viola-Jones, apelidado em inglês de *Haar Cascades Classifier*, ou em português Cascata de Classificadores Haar (MOREIRA, 2008), foi o primeiro algoritmo de detecção a fornecer detecção de objetos com taxas de sucesso competitivas e em tempo real. Foi proposto em 2001 por Paul Viola e Michael Jones. Apesar de poder detectar uma variedade de classes de objetos, foi desenhada com o objetivo primordial de detectar faces.

Com a utilização das características Haar, o objetivo a ser alcançado é classificar corretamente um dado objeto a partir do conjunto de suas principais características.

De acordo com (DAVISON, 2013), o algoritmo funciona com o uso de integrais que são rápidas e eficientes, possibilitando ainda reduzir drasticamente o número de atributos

que precisam ser testados para decidir se a imagem contem um objeto (por exemplo uma face). O teste de atributos da imagem são organizados em "cascatas"(também pode ser representado por uma árvore binária), representando o uso das integrais iterativas.

O nó-raiz da árvore binária deve contér um valor representando o teste que se provou ser o melhor em encontrar um objeto durante o treino. Se uma imagem não é rejeitada por este valor teste entao a imagem segue para o nó com o segundo melhor valor de teste, e assim sucessivamente. Apenas se a imagem alcançar o fim de todos os nós (o fim da cascata ou da árvore) sem ser rejeitado, a imagem certamente deverá conter o objeto característico.

O grande ponto negativo é que este algoritmo também detecta os objetos que não são realmente os objetos que se tinha a intenção de detectar(DAVISON, 2013). Por exemplo, o desenho de um rosto certamente vai ser detectado com uma face original, e este desenho não precisa ser muito bem elaborado. As vezes uma disposição de sombras e regiões luminosas aleatórias podem er detectadas como objetos característicos .

Segundo (MOREIRA, 2008), os estágios dentro de uma cascata são criados através da combinação de funções de classificadores Haar previamente treinados. O principal objetivo do uso de cascatas é fazer com que seus estágios iniciais descartem um grande numero de regiões que contem o objetivo desejado, e estágios mais avançados sejam melhores em evitar um falso positivo na região analisada.

A Figura 11 representa os $N$ estágios de uma cascata de classificadores. Cada um deles deve descartar ao máximo o número de regiões da imagem que não contém o objeto, a fim de diminuir a quantidade de processamento de outras sub-áreas da imagem original (MOREIRA, 2008).

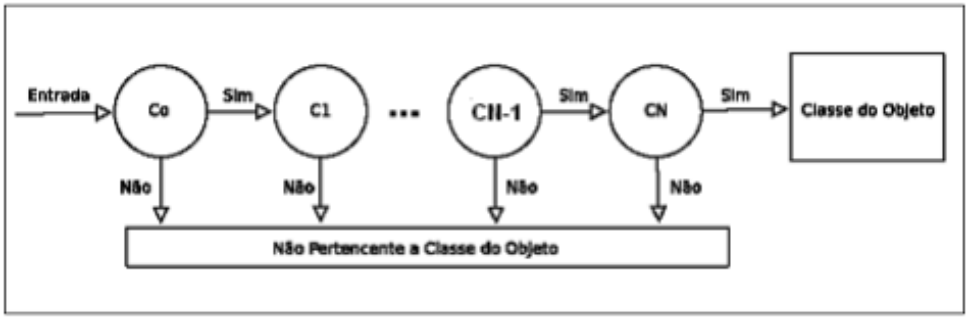


FIGURA 11 – Cascata de classificadores com estágios (C0, C1, ..., Cn)

.

FONTE: (MOREIRA, 2008)

O processo para geração da cascata é guiado por um conjunto de metas de detecção e de desempenho. Na prática, uma arquitetura muito simples pode ser usada para produzir uma eficiente cascata de classificadores. O usuário pode informar as taxas mínimas aceitáveis e cada estágio da cascata é treinada junto ao conjunto de características Haar contando com cada vez mais elementos até que se atinjam as taxas de detecção de

sucesso e falsos positivos. As taxas são encontradas testando o detector corrente sobre um conjunto de validação. Se a taxa de falsos positivos ainda não for atendida, então um novo estágio subsequente é ativado, coletando todas as falsas detecções encontradas (MOREIRA, 2008).

Para que o algoritmo possa detectar o objeto de interesse, é necessário que a imagem seja percorrida diversas vezes em várias direções, assim a cada iteração é analisado uma região de tamanho diferente, chamada **janela de busca** (MOREIRA, 2008).

O usuário pode definir o tamanho desta janela que deve ser maior ou igual ao tamanho das imagens que foram treinadas na cascata, ou seja, se a cascata foi treinada com imagens positivas de 24x24 pixels, o tamanho mínimo da janela de busca deve seguir a mesma resolução (MOREIRA, 2008). As janelas devem funcionar em *thread* (multiplas locações), sendo que $n$ janelas devem se deslocar no eixo X da imagem e $m$ janelas no eixo Y. A quantidade de janelas deve ser o suficiente para ocupar todo o espaço da imagem.

Outro valor que o usuário pode informar é número de detecções subsequentes necessários para que a região seja considerada a ter o objeto alvo. Reduzir este valor pode aumentar a velocidade de processamento, mas também aumenta as chances de taxas de falsos positivos (DAVISON, 2013).

O resultado do algoritmo são as posições dos objetos classificados com as características as quais se recebeu treinamento, podendo ser identificados na imagem.

## 2.3 *EIGENFACES* E ANÁLISE DE COMPONENTES PRINCIPAIS (ACP)

O processo de reconhecimento de faces depende de uma fase de treinamento precedente envolvendo imagens faciais com suas identidades associadas. Imagens típicas de treinamento são mostradas na Figura 12.

É importante que as imagens sejam todas colhidas e orientadas de maneira similar, para que as variações entre as imagens sejam causadas por diferenças entre as faces e não por diferenças em plano de fundo ou posição facial. Também deve haver uniformidade na posição, clareamento, tamanho e resolução da imagem. É recomendado incluir várias imagens da mesma face mostrando diferentes expressões, como sorrindo ou abrindo a boca. A relação da imagem com o indivíduo pode ser feita de diversas maneiras, uma delas é codificar alguma chave único para o indivíduo junto ao nome da imagem (DAVISON, 2013).

O processo de treinamento cria as *EigenFaces*, como mostra a Figura 2 explicada na Subseção 1.2.2, que são composições das imagens de treinamento que tem a propriedade de acentuar elementos que podem ser distinguidos mais facilmente pelo algoritmo. Uma imagem de face pode gerar várias *Eigenfaces*. As imagens de treino da face podem ser representadas por uma sequência de "pesos"(p1, p2, p3, p4, p5, p6), como ilustra a figura


FIGURA 12 – Exemplo de imagens típicas de treinamento.

Figura 13.

O objetivo é que a imagem de treinamento pode ser decomposta por uma soma dos pesos das multiplas *engeifaces*, sendo todos estes pesos salvos como uma sequência.

Nem todas as eigenfaces são igualmente importantes - algumas pode contar elementos faciais mais importantes para distinguir faces. Isto quer dizer que não é necessário utilizar todas as eigenfaces para reconhecer uma face, o que permite que uma imagem seja representada por uma pequena quantidade de eigenfaces, o que ocupa menos espaço e torna a execução do algoritmo mais rápida. O ponto negativo é que com menos pesos, a probabilidade de se perder importantes elementos faciais também aumenta(DAVISON, 2013).

Outra maneira de se entender a relação entre eigenfaces e as imagens é representa-las em um plano cartesiado multidimensional (ou um *eigenspace*), como mostra a Figura 14.

Os pesos agora podem ser observados como as coordenadas da imagem de treinamento em um plano cartesiano multidimensional (*eigenspace*). Na Figura 14, existem apenas 3 eixos representando as eigenfaces p1, p2 e p3, porém se houvessem 10 eigenfaces, haveria 10 eixos neste plano cartesiano.

Após o processo de treinamento das faces vem o processo de reconhecimento. Neste novo processo, uma nova imagem para reconhecimento deve ser decomposta em eigenfaces (na mesma maneira da Figura 13). A sequência de pesos resultante é comparada com

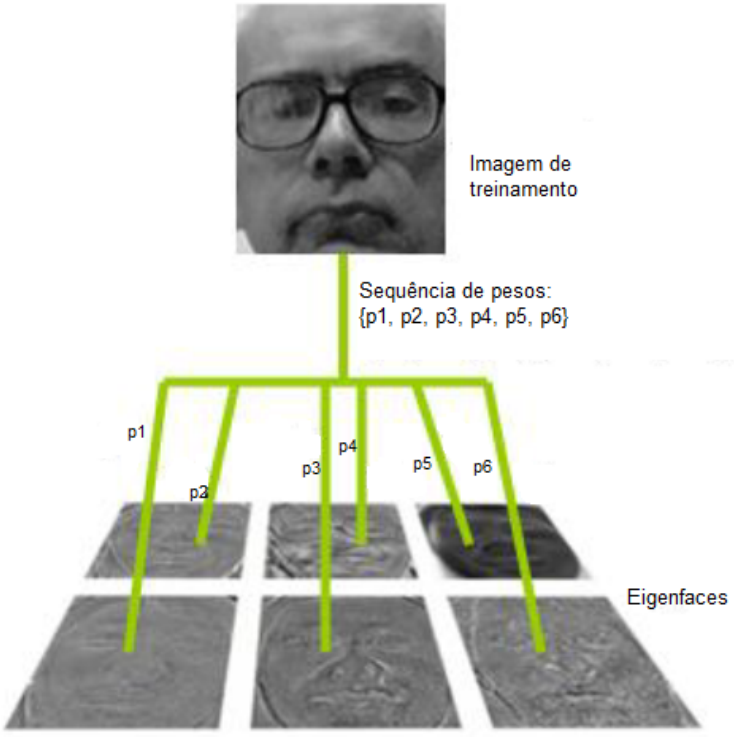


FIGURA 13 – Exemplo de uma imagem de treinamento representada em uma sequência de pesos de engenfaces.
FONTE: Adapdato de (DAVISON, 2013)

cada sequência de pesos que foram treinadas previamente, e o nome associado com o as características mais próximas dos eigenfaces treinados é associado (DAVISON, 2013).

Em outras palavras, em termos de *engenspaces*, a nova imagem é posicionada no plano cartesiano com suas próprias coordenadas (pesos). Então uma técnica de medida de distância (usualmente a distância Euclidiana, que será abordada na Subseção 2.3.2.1) é utilizada para encontrar e figura de treinamento correspondente mais próxima. Sendo assim, uma distância Zero seria uma correspondência (reconhecimento) perfeita (DAVISON, 2013). A Figura 3 na Subseção 1.2.2 ilustra este processo.

As Eigenfaces são geradas utilizando um procedimento matemático chamado Análise de Componentes Principais - ACP (em inglês Principal Components Analysis - PCA). Esta técnica acentua as similaridades e as diferenças nos dados.

### 2.3.1 Análise de Componentes Principais (ACP)

Para que esta técnica seja explicada, será utilizado vetores de valores numéricos: o vetor $x$ e $y$, como mostra a Tabela 2.

Os dados de $x$ e $y$ poderiam representar a altura de estudantes, preços de frutas, ou valores numéricos extraídos de uma imagem, por exemplo. O objetivo da demonstração é acentuar as diferenças e similaridades entre os dois vetores de dados em uma abordagem

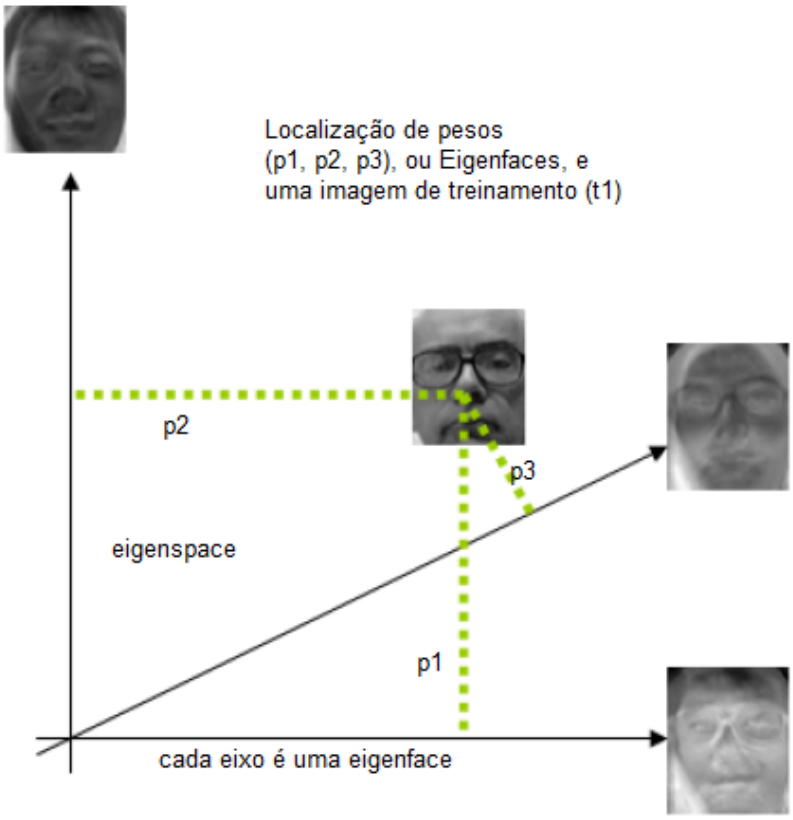


FIGURA 14 – Exemplo de eigenfaces representadas em um plano cartesiano (*eigenspace*)de 3 dimensões.
FONTE: Adapdato de (DAVISON, 2013)

TABELA 2 – Dados dos vetores $x$ e $y$.

| x | y |
|---|---|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

FONTE: (DAVISON, 2013)

numérica (DAVISON, 2013).

Para tal, medidas estatísticas como média e desvio padrão (uma medida que tem o objetivo de espalhar valores acerca da média). Variância é outra medida de espalhamento, que é igual ao desvio padrão ao quadrado:

A variável $n$ é o número de itens dos vetores (10 de acordo com a Tabela 2), e $\bar{x}$ é

$$\text{var}(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{(n-1)}$$

a média dos valores d conjunto de dados.

Contudo, a média e a variância fornecem informação apenas sobre a forma dos dados em um único vetor (por exemplo, e média e a variância do vetor $x$), sendo que o objetivo é quantificar as diferenças **entre** os dois vetores (DAVISON, 2013). Justamente a covariância é uma medida estatística, generalizada da variância, que compara dois conjuntos de dados (DAVISON, 2013):

$$\text{cov}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

Se a fórmula da covariância for aplicada nos dados dos vetores $x$ e $y$ dados na Tabela 2, o resultado será 0.6154. A parte em que se deve estar atendo é o sinal: se for negativo significa que quando um vetor aumenta em dados, o outro diminui. Quando o valor é positivo, que é o caso, então os dois conjuntos de dados aumentam juntas. Para ilustrar este conceito, na Figura 15 plota-se os valores correspondentes dos vetores $x$ e $y$ como pontos em um gráfico (DAVISON, 2013).

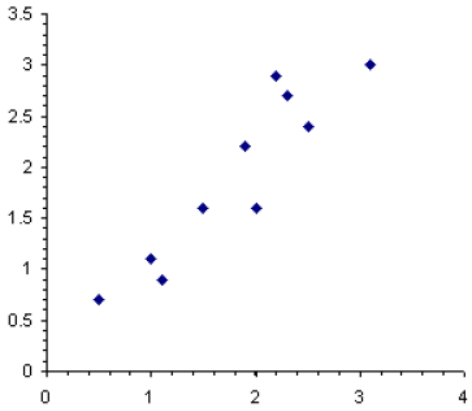


FIGURA 15 – Plotagem dos valores de $x$ e $y$.
FONTE: Adapdato de (DAVISON, 2013)

Se houverem mais de dois vetores (por exemplo o vetor $x$, $y$ e $z$) então a covariância deve ser calculada entra todas os conjuntos de dados, ou seja, calculando cov(x, y), cov(x, z) e cov(y, z).

Não há necessidade de calcular cov(y, x), cov(z, x) e cov(z, y) sendo que a definição da equação garante que cov(A, B) é igual a cov(B, A) .

A maneira padrão de armazenar a covariância entre múltiplos vetores de dados é por uma matriz onde os conjuntos de dados tornam-se os índices das colunas e linhas. Por exemplo, a covariância para os conjuntos de dados x,y e z se tornariam uma matriz de dimensões 3x3 (DAVISON, 2013):

$$\begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix}$$

Na diagonal principal, pode-se observar que a covariância é entre o conjunto de dados a própria covariância, que é equivalente à variância nestes dados. Podemos provar isto comparando a variância de x ( var(x) ) e a covariância de x com x ( cov(x) ). A matriz também apresenta simetria em volta da diagonal principal.

Os conjuntos de dados $x$ e $y$ da Tabela 2 seriam representados em uma matriz de dimensões 2x2:

$$\begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$$

A matriz de covariância é um meio útil de demonstrar a relação entre os conjuntos de dados, mas deve-se obter ainda mais informações para que sejam geradas as *eigenfaces* calculando dois vetores que serão contemplados na seção seguinte: *eigenvectors* e *eigenvalues*.

### 2.3.2 ACP E *Eigenfaces*

Para que as eigenfaces sejam geradas, deve-se transformar a covariância ou outros conjuntos de dados chamados de *eigenvectors* e *eigenvalues*.

Segundo (DAVISON, 2013), o *eigenvector* é um vetor de valores ordinário que quando multiplicado por uma dada matriz, muda de magnitude, enquanto que *eigenvalue* é apenas um termo para esta magnitude.

Por exemplo, dada a matriz:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}$$

O eigenvector para esta matriz é $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$, pois quando multiplicado pela matriz acima, o mesmo vetor é retornado o mesmo vetor é retornado se multiplicado por 4:

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = 4 \times \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Sendo assim, 4 é o *eigenvalue* para o *eigenvector* $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$.

Os eigenvectores podem ser encontradas apenas para matrizes quadradas, porém nem todas. Para uma dada matriz n x n, se realmente houver eigenvectores, haverá n. A matriz 2x2 usada de exemplo acima tem dois eigenvetores (e seus valores eigenvalues correspondentes).

A relação eigenvalue pode ser escrita matematicamente como:

$\Lambda\ \nu = \lambda\ \nu$

onde $\Lambda$ é a matriz quadrada, $\nu$ é um eigenvetor para A e o valor $\lambda$ é um eigenvalue.

Para usa-los no algoritmo PCA, os eigenvetores devem ser normalizados para que todos eles tenham a mesma unidade de tamanho. Isto é, o eigenvetor acima, $\begin{pmatrix}3\\2\end{pmatrix}$, tem a unidade de tamanho "falso"de $\sqrt{(3^2+2^2)} = \sqrt{13}$. Se o vetor for dividido por este valor resultara em um tamanho de unidade (DAVISON, 2013):

$$\begin{pmatrix}3\\2\end{pmatrix} \div \sqrt{13} = \begin{pmatrix}3/\sqrt{13}\\2/\sqrt{13}\end{pmatrix}$$

Se recuperarmos a matriz de covariância que geramos acima:

$$\begin{pmatrix}0.6166 & 0.6154\\0.6154 & 0.7166\end{pmatrix}$$

pode-se observar que, com é uma matriz 2x2, possui dois eigenvetores e eigenvalues:

$$\begin{pmatrix}-0.7352\\0.6779\end{pmatrix} \text{ e } 0.049$$

$$\begin{pmatrix}0.6779\\0.7352\end{pmatrix} \text{ e } 1.284$$

Ambos os eigenvetores possuem unidade de tamanho.

#### 2.3.2.1 Utilizando Eigenfaces como Componente Principal

Observando novamente a plotagem dos vetores x e y contemplados na Figura 15, se a média destes conjuntos de dados ($\bar{x}$ e $\bar{y}$) for subtraído de seus valores, estes dados então seriam **normalizados**. Isto também quer dizer que as linhas que representam os vetores $x$ e $y$ são transladados para o centro como mostra a Figura 16.

Os dois engenvetores pode ser adicionados ao gráfico como linhas convertendo os vetores em equações. $\begin{pmatrix}-0.7352\\0.6779\end{pmatrix}$ torna-se $y = \frac{-0.7352}{0.6779}x$ e $\begin{pmatrix}0.6779\\0.7352\end{pmatrix}$ torna-se $y = \frac{0.6779}{0.7352}x$.

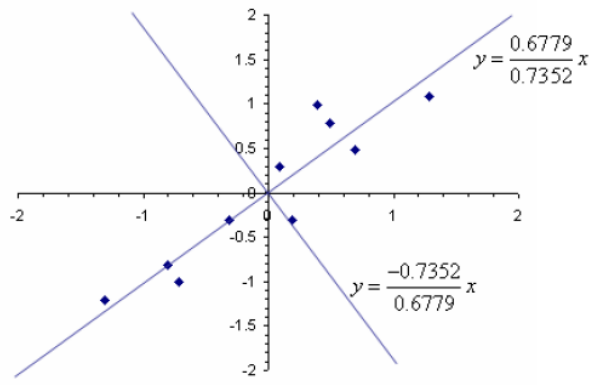


FIGURA 16 – Plotagem normalizada dos dados de $x$ e $y$ com eigenvetores.
FONTE: (DAVISON, 2013)

Os dados normalizados e as duas equações são ilustradas na Figura 16 como linhas azuis.

Observa-se que na Figura 16 demonstra como os eigenvectores acentuam as relações entre os conjuntos de dados- os vetores indicam como os conjuntos de dados se espalham nas coordenadas do espaço cartesiano. Este espalhamento facilita a distinção dos dados (DAVISON, 2013).

Dos dois eigenvetores marcados em azul, a linha $\frac{0.6779}{0.7352}x$ é a mais útil, pois os dados estão mais espalhados acerca do trajeto desta linha. Isto pode ser confirmado numericamente observando o eigenvalue associado com seu eigenvetor: o eigenvalue (1.284) é um melhor indicador de espalhamento pois é o maior dentre os dois eigenvalues.

A outra linha, $\frac{-0.7352}{0.6779}x$, contribui com a informação demonstrando como os dados são espaçados para direita ou esquerda do eigenvetor principal (a linha $\frac{0.6779}{0.7352}x$). Entretanto, trata-se de um componente menos importante no espalhamento de dados, assim como indica seu eigenvalue (0.049) (DAVISON, 2013).

O eigenvetor com o maior eigenvalue (no caso a linha $\frac{0.6779}{0.7352}x$, ou o vetor $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$) é chamado de **componente principal** do conjunto de dados. Isto é, os eigenvetores e eigenvalues são efetivamente utilizados para performar uma análise de componente principal nos conjuntos de dados (DAVISON, 2013).

Todos os eigenvetores extraídos da matriz são perpendiculares. Isto significa que é possível rotacionar (talvez refletir) os dados para que os eigenvetores tornem-se alinhados com os eixos. Se este componente principal $\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$ for rotacionado e refletido para que seja alinhado com o eixo y, resulta-se em uma plotagem mostrada na Figura 17:

Observando a Figura 17, o ponto mais próximo da origem se situa na coordenada (0.13, 0.27). Pode-se afirmar que também que este ponto tem uma sequência de pesos relativos a cada eigenvetor, no caso, a sequência 0.13, 0.27.

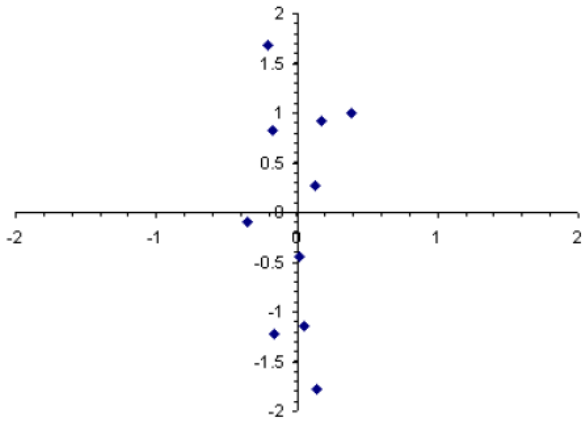


FIGURA 17 – Versão rotacionada e refletida da Figura 16
FONTE: (DAVISON, 2013)

Tanto a denotação de coordenadas quanto a de pesos pode ser usada para representar as eigenfaces - como contemplado anteriormente, na Figura 13 foi dado o exemplo em que uma imagem é uma sequência de pesos de seis eigenfaces, enquanto que, alternativamente, na Figura 14 é representada em coordenadas de um plano cartesiado de 3 dimensões, sendo que cada eixo é definido por uma *eigenface* (ou *eigenvector*).

#### 2.3.2.2 O Reconhecimento

O processo de reconhecimento consiste em recolher novos conjutos de dados (no caso, uma nova face), e utilizando os dados processados até agora e representados na Figura 17, deve-se comparar os dois novos conjuntos de dados (a nova face e os eigenfaces ou eigenvetores) fazendo uso de medidas de distância.

Existem algumas técnicas de medidas de distância, dentre elas, há uma simples chamada **Distância Euclidiana**. Esta medida nada mais é do que a distância mínima entre dois pontos no espaço (uma linha reta). A distância euclidiana entre os pontos $P = (p_1, p_2, ..., p_n)$ $Q = (q_1, q_2, ..., q_n)$ é definida como:

$$\sqrt{(p_1 - q_1)^2 + (p_1 - q_1)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

O resultado desta medida é utilizada para calcular a taxa de sucesso de uma comparação, sendo assim uma distância nula representaria uma equivalência perfeita.

Contudo, em aplicações na vida real, deve-se reduzir o tamanho dos dados, pois existem informações irrelevantes e a tarefa de reconhecimento pode ser executada mais rapidamente enquanto retém informações suficientes para distinguir entre os pontos quando comparados com os novos conjuntos de dados.

A técnica consiste em reter todas as coordenadas dos dados, mas reduzir a dimensionalidade do eigenspace, isto é, subtrair o número de eixos do plano cartesiano. Este

processo é equivalente a subtrair algumas eingenfaces, mas como citado anteriormente, os enigenvalues de menor intensidade não tem influência no espalhamentos dos dados no plano, sendo pouco significativos para o reconhecimento, e portanto podem ser removidos.

Portanto, os dados mostrados na Figura 17 possuem os seguintes eigenvetores e eigenvalores:

$$\begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix} \text{ e } 0.049$$

$$\begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \text{ e } 1.284$$

Observando que o primeiro *eigenvector* foi rotacionado em direção ao eixo $x$, e o segundo *eigenvector* em direção ao eixo $y$, se o primeiro *eigenvector* que contribui muito pouco para a informação de espalhamento (devido ao seu pequeno *eigenvalue*) for descartado junto ao seu eixo, os dados deste *eigenvcetors* são projetados para o eixo $y$, resultando na plotagem da Figura 18

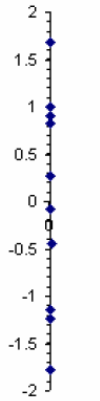


FIGURA 18 – Dados da Figura 17 projetados em direção ao eixo y.
FONTE: (DAVISON, 2013)

Apesar dos eixos serem eliminados, seus dados ainda estão presentes e espalhados o suficiente para que o novo conjunto de dados seja comparado com o uso da distância euclidiana.

Considerando que um novo conjunto de dados adicionado com os valores do vetor $z = (2.511,\ 2.411)$, e estes conjuntos passarem pelos mesmos processos que os vetores $x$ e $y$, e por fim normalizado e projetado assim como mostra a Figura 16 e Figura 17, plotando-o no mesmo espaço resulta na Figura 19.

A Figura 19 é a mesma que a Figura 17 com a adição de um novo ponto vermelho que indica a ccoordenada do novo conjunto de dados do vetor $z$, posicionada logo acima do ponto (-0.1751, 0.8280), que por sua vez começou como um ponto de treinamento

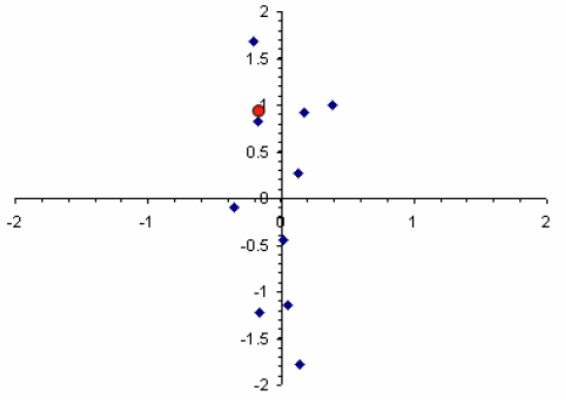


FIGURA 19 – Plotagem dos novos dados com os dados de treino transformados.
FONTE: (DAVISON, 2013)

de pesos (2.5, 2.4), informada nos conjuntos de dados $x$ e $y$. Isto quer dizer que o novo conjunto de dados $z$ (2.511, 2.411) possui a menor distância euclidiana com o conjunto de treinamento (2.5, 2.4).

A real vantagem dos eigenvetores é quando a dimensionalidade do espaço (eigenspace) torna-se extremamente grande. No presente exemplo foi utilizado uma dimensionalidade dois, enquanto que em aplicações reais com imagem os eixos podem aumentar para valores como 40 mil (DAVISON, 2013).

Com tamanha dimensão e portanto, quantidade de eixos, como citado anteriormente, pode-se eliminar alguns eixos dependendo de seus valores eingenvalues (se forem pequenos, podem ser considerados dispensáveis). Se por uma hipótese isto for feito neste exemplo incluindo o novo conjunto dados $z$, a plotagem do resultado seria como na Figura 20 abaixo:

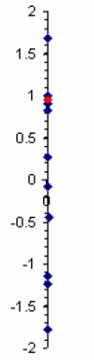


FIGURA 20 – Plotagem dos conjuntos de dados $x$, $y$ e $z$, trasnformados e projetados no eixo do Componente Principal
FONTE: (DAVISON, 2013)

A Figura 20 é similar à Figura 18 com exceção do ponto vermelho (novo dado). Pode-se observar nesta figura o problema potencial em se reduzir a dimensionalidade: a probabilidade de erros aumenta tornando difícil de decidir qual ponto de treinamento é que possui a menor distância euclidiana. Outro fator da equação da distância euclidiana é

que se utiliza de um parâmetro para definir quando dois pontos estão perto. Se um valor alto é definido então diversos pontos podem estar dentro do alcance aceitável do novo dado (DAVISON, 2013).

### 2.3.3 EigenVectors para EigenFaces

O transformação dos valores numéricos gerados na ACP para imagens EigenFaces nao são necessário para os algoritmos apresentados. Essa transformação serve apenas para satisfazer olhos humanos, ou seja, uma tentativa de observar em imagens os valores gerados pela teoria.

Essencialmente, deve-se converter a imagem para um conjunto de dados, como os vetores $x$ e $y$ apresentados na Tabela 2, e performar o algoritmo ACP como descrito anteriormente. A translação de uma imagem para um conjunto de dados é feita tratando esta como um vetor de duas dimensões, e em seguida mapear para um vetor unidimensional como sugere a Figura 21:

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & & p_{2N} \\ \vdots & & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix} \rightarrow \begin{pmatrix} p_{11} \\ \vdots \\ p_{1N} \\ p_{21} \\ \vdots \\ p_{2N} \\ \vdots \\ p_{NN} \end{pmatrix}$$

FIGURA 21 – Conversão de imagem para conjunto de dados. A variável $P$ seria a representação de um pixel.
FONTE: (DAVISON, 2013)

Supondo que a imagem tem o tamanho de 200x200 pixels em escala de cinza, o conjunto de dados resultante residirá em um *eigenspace* de 40 mil dimensões, o que acarretará na geração de 40 mil *eigenvectors*, cuja a maioria possui um *eigenvalue* desconsiderável ou insignificante, podendo assim ser descartados. Isto facilitará o processamento da imagem e a utilização da técnica.

Em geral, supondo que uma imagem quadrada em escala de cinza tem o tamanho de $N$ pixels, então $N^2$ eigenvetors serão criados. No seguinte exemplo, será assumido que existem $M$ imagens:

A técnica consiste em "dividir"a matriz de covariância que é $(N^2xN^2)$ em duas matrizes de $(N^2xM)$ e $(MxN^2)$.

## 2.4 Considerações Finais

Neste capítulo foram apresentados os conceitos necessários para o entendimento de todo o processo de Reconhecimento de Faces, constituído de outros sub processos como

aquisição de imagem e vídeo, processamento e classificação de imagens, detecção da face, treinamento e por fim o reconhecimento efetivo da face.

Os algoritmos de detecção de face Viola-Jones, e o de reconhecimento de faces EigenFaces que utiliza o método matemático ACP (Análise de Componentes Principais), foram detalhados na ordem sequêncial de seus processos utilizando exemplos.

No próximo capítulo abordará o plano de materiais e métodos elaborado a partir do conhecimento adquirido no Capítulo 2.

# 3 MATERIAIS E MÉTODOS

Após os estudos realizados sobre processamento e classificação de imagens, sobre o algoritmo de detecção *Haar-cascades* e o de reconhecimanto *EigenFaces* contemplado no capítulo passado, é necessário definir o material e meio para desenvolver o sistema aqui proposto.

Este capítulo apresentará os materiais e metodos que serão utilizados para o desenvolvimento do sistema proposto neste trabalho.

## 3.1 MATERIAIS

As seguintes seções irão detalhar os equipamentos que serão usados para o desenvolvimento do sistema e as tecnologias de software escolhidas baseado na revisão bibliográfica realizada e na experiência do autor.

### 3.1.1 Hardwares

Lista-se a seguir, os equipamentos necessários para a realização deste trabalho:

- Câmera digital do tipo *Web* (*Webcam*) de 720p de resolução (ou 1280x720 *pixels*);
  - será utilizada para captar o vídeo analógico e disponibilizá-lo no formato digital
- Microcomputador pessoal com processador Intel i7, 32 gigabytes de memória, e placa de vídeo externa de 16 megabytes (alto rendimento);
  - será utilizado como principal instrumento de desenvolvimento
- Microcomputador pessoal do tipo notebook com processador Intel i3 de versão antiga, 2 gigabytes de memória, sem placa de vídeo externa, utilizando a própria memória RAM para renderização (baixo à médio rendimento);
  - será utilizado como instrumento de desenvolvimento secundário, para testar questões de consumo de memória e processamento (desempenho)

Estes *hadwares* são considerados suficientes para a realização deste trabalho.

### 3.1.2 Softwares

As subseções seguintes irão citar e detalhar as tecnologias que serão utilizadas para a implementação do trabalho aqui proposto, descrevendo a maneira que serão utilizados.

#### 3.1.2.1 Linguagem Java

A tecnologia Java é usada para desenvolver aplicativos para uma ampla variedade de ambientes, de dispositivos consumidores a sistemas corporativos heterogêneos (PERRY, 2016).

A linguagem Java deriva da linguagem C, portanto suas regras de sintaxe assemelham-se às regras de C. Por exemplo, os blocos de códigos são modularizados em métodos e delimitados por chaves ( e ) e variáveis são declaradas antes que sejam usadas (PERRY, 2016).

Estruturalmente, a linguagem Java começa com pacotes. Um pacote é o mecanismo de *namespace* da linguagem Java. Dentro dos pacotes estão as classes e dentro das classes estão métodos, variáveis, constantes e mais (PERRY, 2016).

Java é uma linguagem Orientada a Objetos. A Programação Orientada a Objetos (POO) diz respeito a um padrão de desenvolvimento que consiste na representação de cada elemento em termos de um objeto, ou classe. Esse tipo de representação procura aproximar o sistema que está sendo criado ao que é observado no mundo real, e um objeto contém características e ações, assim como vemos na realidade (GASPAROTTO, 2014).

O Java possui classes prontas para a solução dos mais diversos problemas. A seguinte lista descreve as classes ou pacotes que este trabalho fará uso, dentre outras:

- Pacote ***java.swing*** e ***java.awt***: possui classes para a construção da interface (ou *front-end*) do sistema. Botões, janelas, manipulação de eventos e outros componentes visuais;
- Pacote ***java.io***: utilizado para manipular entrada e saida de informações, manipulação de arquivos e pastas, etc;
- Pacote ***java.util***: possui classes de estrutura e manipulação de listas e filas, além uma variedade de classes úteis para problemas esporádicos;
- outros pacotes utilizados casualmente de acordo com a necessidade como o ***java.lang*** provê classes que são fundamentais para a programação Java, por exemplo as classes de criação e execução de *threads*;

#### 3.1.2.2 Biblioteca OpenCV, JavaCV e JavaCPP

A biblioteca **OpenCV** (Open Source Computer Vision Library) foi originalmente desenvolvida pela Intel em 2000 e hoje é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão computacional (OPENCV..., 2018).

Seu código foi escrito em nas linguagens C e C++, porém possui interface para várias outras linguagens como Python, Visual Basic, Java, dentre outras. Incorpora algoritmos para a resolução de vários problemas na área de visão computacional, inclusive o algoritmo Haar-Cascades contemplados na Subseção 2.2.2, o qual será utilizado.

A biblioteca **JavaCPP**, atualmente mantida pelo grupo ByteDeco, é considerada uma *bridge* ("ponte") entre a linguagem C++ e Java, disponibilizando classes de configurações e interfaces para várias bibliotecas escritas em C++, inclusive a OpenCV (BYTEDECO, 2018a).

Por sua vez, a biblioteca **JavaCV**,inicialmente desenvolvida pela Google e agora mantida pelo grupo ByteDeco, utiliza as interfaces e *wrapers* da biblioteca JavaCPP e provê classes úteis que podem ser usadas para manipular a OpenCV de maneira mais fácil para plataforma Java (BYTEDECO, 2018b). Também oferece drivers de aceleração gráfica que otimizam o funcionamento da OpenCV na plataforma Java, dentre outras utilizades.

Estas três bibliotecas funcionam de maneira conjunta para resolver a problemática de aquisição, processamento de imagens, e classificação de objetos, e será utilizada neste trabalho.

A lista a seguir disponibiliza e detalha os principais pacotes, classes e métodos ou funções que se fará uso destas bibliotecas de desenvolvimento. As definições de suas funções foram retiradas das documentações dispostas nos *links* de (OPENCV..., 2018), (BYTEDECO, 2018b) e (BYTEDECO, 2018a):

- Pacotes/classes:
  - ***javacpp.opencv_core.CvMemStorage***: classe que representa um bloco de memória para armazenamento de vários componentes da OpenCV. Trata problemas alocação e desalocação de memória;
  - ***javacpp.opencv_core.CvRect***: esta classe é utilizada para representar um segmento de uma imagem com pontos x e y, altura e largura do segmento;
  - ***javacpp.opencv_core.CvSeq***: classe que representa uma lista (ou sequência) de classes do tipo *CvRect*;
  - ***javacpp.opencv_core.IplImage***: utilizada para representar uma imagem no fomrato *raster*;
  - ***javacv.VideoInputFrameGrabber***: classe responsável por estabelecer conexão com câmeras conectadas a um dispositivo (microcomputador, por exemplo);

  - ***javacv.FrameGrabber***: esta classe tem a função de monitorar o fluxo de *frames* de uma *webcam* conectada a partir da classe *VideoInputFrameGrabber* e disponibilizar uma imagem advinda deste fluxo quando requisitado;
  - ***javacv.Java2DFrameConverter***: tem a função de converter as classes de representações de imagens *IplImage* e *java.awt.image*;
  - ***javacv.OpenCVFrameConverter***: tem a função de converter as classes de representações de imagens *IplImage* e *javacv.Frame*, disponibilizada pela classe *FrameGrabber* acima listada;
  - ***javacpp.opencv_ objdetect.CvHaarClassifierCascade***: classe que contém as configurações e o algoritmo de classificação Haar-Cascades, utilizado para a detecção de objetos (ou faces);

- Métodos ou funções:

  - ***javacpp.opencv_ core.cvClearMemStorage***: esta função desaloca o espaço de memória utilizada pela classe *CvMemStorage* listada acima;
  - ***javacpp.opencv_ core.cvCreateImage***: responsável por criar objetos da classe *IplImage*, aceitando parâmetros como produndidade de cores, tamanho da imagem, dentro outros;
  - ***javacpp.opencv_ core.cvReleaseImage***: este método libera o espaço de memória ocupado pelos objeto das classe *IplImage*;
  - ***javacpp.opencv_ core.cvGetSeqElem***: manipula objetos da classe *CvSeq* listada acima;
  - ***javacpp.opencv_ core.cvGetSize***: função dispõe o tamanho em *bytes* de um objeto das classes *CvSeq*, *CvRect* e *IplImage*;
  - ***javacpp.opencv_ imgproc.cvCvtColor***: função que tem a habilidade de mudar a disposição e profundidade de cores de uma imagem da classe *IplImage*, por exemplo, convertendo-as em escala de cinza como contempla a Subseção 2.1.5.1;
  - ***javacpp.opencv_ imgproc.cvEqualizeHist***: função com a habilidade de performar equalização de histograma, contemplada na Subseção 2.1.5.3;
  - ***javacpp.opencv_ imgproc.cvResize***: função com a habilidade de escalonamento, contemplada na Subseção 2.1.5.2
  - ***javacpp.helper.opencv_ objdetect.cvHaarDetectObjects***: este método recebe um objeto configurado do tipo *CvHaarClassifierCascade*, uma imagem do tipo *IplImage*, dentre outros parâmetros, e efeticamente detecta objetos (no caso, face) na imagem retornando objetos da classe *CvSeq* ou *CvRect*;
  - outras funções ou métodos para manipular ou converter classes de objetos irão eventualmente ser utilizadas;

Estas três bibliotecas também possuem um conjunto de parâmetros e constantes pré estabelecidas para facilitar o estabelecimento do ambiente desenvolvimento que irão ser utilizadas.

#### 3.1.2.3 Biblioteca Colt

A biblioteca Colt foi desenvovida pela CERN (sigla francesa que corresponde à *Conseil européen pour la recherche nucléaire*, em portugês Conselho Europeu de Pesquisas Nucleares) e provê uma gama de bibliotecas para Computação Científica e Técnica de Alta Performance em Java (COLT, 2018). Será utilizada na implementação do algoritmo *EigenFaces* para performar as operações necessárias descritas na Subseção 2.3.1 e Subseção 2.3.2. Lista-se a seguir as classes que serão utilizadas desta biblioteca (COLT, 2018):

- ***cern.colt.matrix.DoubleMatrix2D***: uma classe abstrata para representrar matrizes de duas dimensões, contendo elementos do tipo primitivo *double*;
- ***cern.colt.matrix.impl.DenseDoubleMatrix2D***: uma classe que herda e implementa os métodos abstratos da classe *DoubleMatrix2D* acima descrita. Possui métodos para clonar a matriz, reaver algum elemento, buscar algum elemento, conversão em vetores de *double* po exemplo, dentre outras utilidades;
- ***cern.jet.math.Functions***: contém classes métodos que definem funções matemáticas a serem aplicadas em matrizes como respresentam a classe *DenseDoubleMatrix2D* acima citada, como soma, multiplicação, subtração de matrizes, etc;
- ***cern.colt.matrix.linalg.EigenvalueDecomposition***: classe que contém métodos para a extração dos *eigenvalues* e *eigenvectors* de uma matriz recebida, necessários para certas etapas do algorítio *EigenFaces* descrito na Subseção 2.3.2;

#### 3.1.2.4 Sistema GIT

Git (uma gíria inglesa que referencia uma "pessoa teimosa") é um sistema de gerenciamento e controle de código livre criado pelos desenvolvedores Linus Tornalds (o criador do Sistema Operacional Linux) e Junio Hamano sob a licensa GNU (GPLv2) (GIT, 2018). Este sistema é extremamente útil para se ter cada etapa da escrita do código salva, com possibilidade de reversão de ações, criação de ramos de desenvolvimento dentre outras utilidades, e é utilizado tando na escrita desta monografia quanto na escrita do código do sistema de reconhecimento aqui proposto.

#### 3.1.2.5 NetBeans IDE

O NetBeans é um IDE (*Integrated Development Environment*, ou em português, Ambiente Integrado de Dezemvolvimento) livre licensiado pela *Common Development and Distribution License* (CDDL) e *GNU General Public License* (GPL).

Este editor de código tráz facilidades como auto-complemento de código, desenho visual de interface, atalhos de teclados para maior produtividade na escrita, *debugger*, dentre outras vantagens, e foi escolhido para a implementação neste trabalho.

#### 3.1.2.6 Sistemas Linux e Windows

O ambiente de desenvolvimento deste trabalho será configurado sob o sistema operacional Linux, que estará executando na máquina de alto rendimento descrita na Subseção 3.1.1, item "2". Já o sistema operacional Windows se executará na maquina de baixo à medio rendimento descrito também na Subseção 3.1.1, item "3", para testes. O sistema Linux é livre, e o sitema operacional Windows possui licensa para a máquina (*notebook*) na qual estará sendo executada.

## 3.2 Métodos

O método define-se com um processo organizado, lógico e sistemático de pesquisa, instrução, desenvolvimento, etc (MéTODO, 2018). Existem vários métodos de pesquisa como o observacional, comparativo, histórico, experimental, dentre outros. Neste trabalho terá uma abordagem experimental, descrita na seção a seguir (FACHIN, 2017).

### 3.2.1 Método Experimental

No método experimental, variáveis são tratadas de maneira pré estabelecida e seus efeitos suficientemente controlados e conhecidos pelos pesquisador. O princípio central da aplicação do método experimental é que deve-se aceitar os resultados como eles são apresentados, com tudo de imprevisto e acidental que possa haver diante dos resulatos obtidos, ignorando as próprias opiniões e também as alheias (FACHIN, 2017).

### 3.2.2 Desenvolvimento Ágil

O Desenvolvimento Ágil foi padronizado pelo **Manifesto Ágil** como um método independente que reúne características de outras metodologias antigas as quais eram despadronizadas. Este manifesto foi criado para valorizar os seguintes pontos (ARAUJO, 2016):

- Indivíduos e suas interações são mais importantes que processos e ferramentas
- Software que funciona importa mais que uma documentação abrangente

- Colaboração com o cliente vale mais do que negociação de contratos
- Responder à mudanças é melhor do que seguir planos

No desenvolvimento ágil, os projetos adotam o modelo iterativo e em espiral (figura 1.5). Neste processo, as fases do projeto são executadas diversas vezes, produzindo ciclos curtos que se repetem ao longo de todo o desenvolvimento, sendo que, ao final de cada ciclo, sempre se tem um software funcional. Os ciclos são chamados de iterações e crescem em número de funcionalidades a cada repetição, sendo que, no último ciclo, todas as funcionalidades desejadas estariam implementadas, testadas e aprovadas (CASTRO, 2018).

A **XP** (*eXtremeProgramming*), explicada na seção seguinte, será utilizada neste projeto por ser incremental, focar-se no desenvolvimento ao invés da documentação e gerar, a cada iteração, um protótipo, essencial para testes, refinamento até que se chegue à um resultado satisfatório.

#### 3.2.2.1 XP (*eXtremeProgramming*)

Apesar de ser uma metodologia dos anos 90, faz parte dos métodos ágeis, já que se encaixa no que define o que é um método ágil, conforme o Manifesto Ágil. Ela é baseada em três pilares: **agilidade no desenvolvimento, economia de recursos e qualidade do produto final** (CASTRO, 2018).

### 3.2.3 Modelagem UML

A modelagem UML (*Unified Modeling Language*, em português Linguagem de Modelagem Unificada), por vezes chamada de linguagem UML, procura fornecer meios para auxiliar no levantamento dos requisitos que irão constituir um sistema, além de recursos para a modelagem de estruturas que farão parte do mesmo. O fato da UML ser um padrão de grande aceitação no mercado também se deve, em grande parte, à forte integração desta com conceitos da Orientação a Objetos (OO) (GROFFE, 2013). Como este trabalho propões a implementação na liguagem Java que é orientada a objetos, este padrão de modelagem será utilizado na elaboração de documentos modelando os componentes esperados para o funcionamento do sistema.

Dois dos diagramas que fazem parte da linguagem UML serão utilizados (GROFFE, 2013):

- Diagrama de Classes: Permite a visualização do conjunto de classes, detalhando atributos e operações (métodos) presentes, assim como prováveis relacionamentos entre essas estruturas. Considerado um diagrama estrutural;

- Diagrama de Sequência: Demonstra as interações entre diferentes objetos na execução de uma operação, destacando ainda a ordem em que tais ações acontecem num intervalo de tempo. A sequência em que as diversas operações são executadas ocorre na vertical, de cima para baixo. Considera,do um diagramas de interação;

Estes dois diagramas serão utilizados para ilustrar as classes Java que serão implementadas, as interações de seus objetos e os processos do sistema que consiste na aquisição do vídeo, deteção e reconhecimento da face, ou seja, toda a entrada, processamento e saida de dados do sistema detalhando seus componentes e interações.

## 3.3 Considerações Finais

Neste capítulo foram abordados os materiais e métodos necessários para o desenvolvimento do sistema de reconhecimento de faces em vídeo, utilizando o algorítimo *EigenFaces*. Pode-se destacar a modelagem que trará esclarecimento quanto à estrutura do sistema e ao processo que deve ser percorrido, o desenvolimento ágil considerado essencial para se ter resultados rápidos para que sejam analizados e iterados num ciclo de desenvolvimento espiral com melhoras a cada iteração. Outro destaque são as bibliotecas OpenCV, JavaCV e Colt, que juntamente com suas documentações facilitarão o processo de implementação do algorítimo *eigenfaces* e o desenvolvimento do sistema como um todo e por fim, o sistema GIT que grava e assegura cada etapa do processo de escrita da monografia e do cógido.

# 4 IMPLEMENTAÇÃO

Neste capítulo serão abordadas questões do planejamento e da implementação do sistema de reconhecimento de faces em vídeo. Primeiramente, os requisitos do sistemas serão analisados e descritos, seguidos da modelagem destes requisitos em forma de fluxograma, seguidos da modelagem das classes que serão criadas por meio dos diagramas descritos na Subseção 3.2.3, e por fim, as questõs de configuração do ambiente de desenvolvimento e a implementação efetiva do código. Todo este processo é feito seguindo conceitos da metodologia ágil XP, descrita na Subseção 3.2.2.

## 4.1 Análise de Requisitos

Para que se possa iniciar o planejamento da implementação do sistema aqui proposto por meio de diagramação, deve-se colher os requisitos necessários para o funcionamento do sistema. Ou seja, analisar o problema descrevendo entredas e saidas de dados, todas as situações e seus fluxos de informações. Sendo assim lista-se por ordem de execução os requisitos que atendem à proposta do sistema, discriminados por **entrada**, **processamento** e **saída** de dados ou informações:

1. **entrada:** uma câmera do tipo *webcam* descrita na Seção 3.1 deve estar filmando uma área com iluminação controlada, produzindo o fluxo de *frames* e disponibilizando em formado digital para o sistema;

2. **processamento:** o sistema adquire controle do fluxo de *frames* da *webcam*, manipulando-o para que se possa ser exibido na tela do microcomputador e realiza a detecção de uma face, em tempo real, utilizando o algoritmo descrito na Subseção 2.2.2 com os materiais contemplados na Subseção 3.1.2.2;

3. **saída:** o sistema exibe, em tempo real, o fluxo de *frames* processado como define o item acima, na tela do microcoputador. Ao mesmo tempo, caso seja detectada uma face, o sistema deverá desenhar um retângulo em volta da face detectada neste fluxo de *frames*, ativando um campo para que o usuário possa entrar com uma identificação da face detectada;

4. **entrada:** o usuário entra com uma identificação da face detectada, caso haja, em um campo disponibilidado pelo sistema (por exemplo, um nome);

5. **processamento:** o sistema inicia o processo de treinamento da face definido na Seção 2.3, criando um *eigenspace*, e no momento seguinte à criação deste "plano cartesiado", o sistema já deve realizar o processo de reconhecimento descrito na Subseção 2.3.2.2;

6. **saída:** caso seja reconhecida uma face no fluxo de *frames*, tal como descrito no item acima, o sistema deve exibir a identificação introduzida pelo usuário com define o item "4";

## 4.2 Fluxograma

O fluxograma é uma técnica que apresenta de forma gráfica, sequêncial, as atividades de um processo para facilitar a visualização de suas etapas (CéSAR, 2011).

Na elaboração deste fluxograma, deve-se considerar que os requisitos listados na seção anterior ocorrem em tempo-real. Sendo assim, todas as tarefas necessárias para atender os requisitos devem ser executados em um *loop*, exceto para o requisito de entrada identificação da face (item "4") e o requisito de treinamento (item "5", de processamento), que ocorrerão apenas caso o usuário introduza a informação.

A fluxograma da Figura 22 dispõe os requisitos e o fluxo de informação que deve ser realizados:

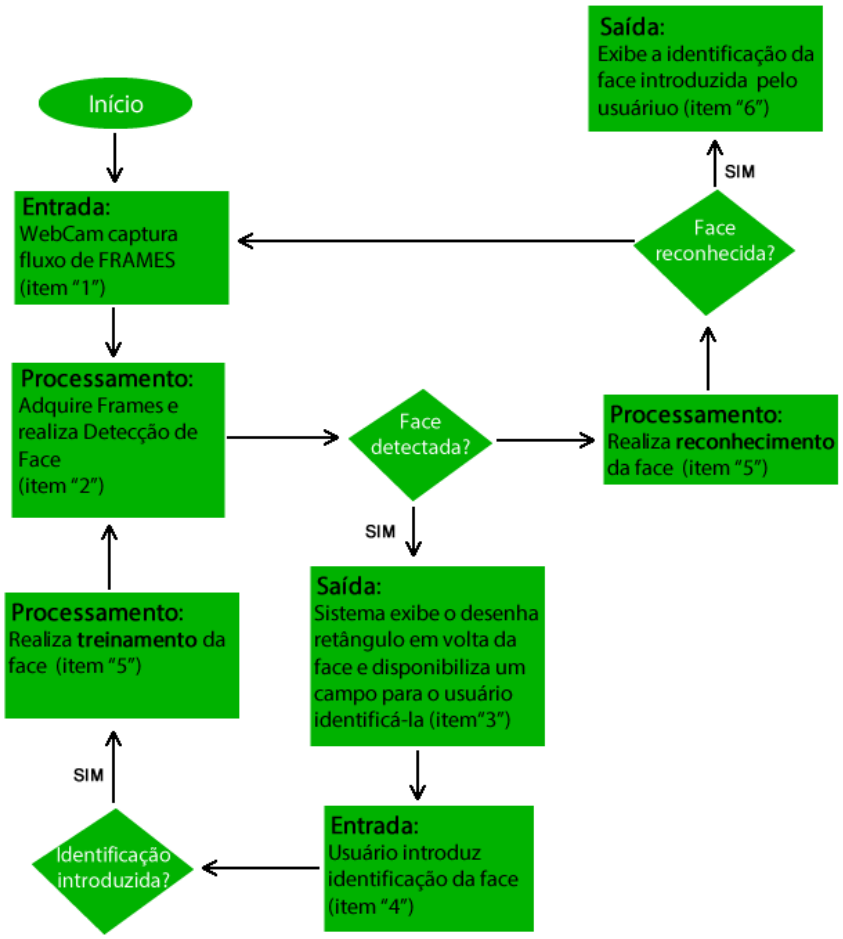


FIGURA 22 – Fluxograma representando os requisitos que devem ser atendidos.

# REFERÊNCIAS


ARAUJO, A. D. Manifesto Ágil e as top 3 metodologias Ágeis de desenvolvimento! BeCode, 2016. Disponível em: <https://becode.com.br/manifesto-agil-e-top-metodologias-ageis/>. Acesso em: 06.2018.

BALAN, W. C. A imagem digital. **Espaço para apoio didático do Curso de Comunicação social - Radialismo**, 2009. Disponível em: <http://willians.pro.br/textos/>. Acesso em: 05.2018.

BARBU NATHAN LAY, G. G. A. Face detection with a 3d model. 2015. Disponível em: <https://pdfs.semanticscholar.org/2bbf/047993f08a608ce4ca22720d374d4cf76f98.pdf>.

BYTEDECO. Javacpp - the missing bridge between java and native c++ libraries. 2018. Disponível em: <https://github.com/bytedeco/javacpp-presets>. Acesso em: 05.2018.

BYTEDECO. Javacv - java interface to opencv, ffmpeg, and more. 2018. Disponível em: <https://github.com/bytedeco/javacv>. Acesso em: 05.2018.

CASTRO, V. A. Introdução ao desenvolvimento Ágil. DevMedia, 2018. Disponível em: <https://www.devmedia.com.br/introducao-ao-desenvolvimento-agil/5916>. Acesso em: 06.2018.

CHAO, W.-L. **Face Recognition**. Dissertação (Mestrado) — National Taiwan University, GICE, Taiwan, Maio 2011. Disponível em: <http://disp.ee.ntu.edu.tw/~pujols/Face%20Recognition-survey.pdf>.

COLT. 2018. Disponível em: <http://dst.lbl.gov/ACSSoftware/colt/>. Acesso em: 05.2018.

CéSAR, F. I. G. In: _____. **Ferramentas Básicas de Qualidade, 1a Ed. 2011**. [S.l.]: Biblioteca 24hrs, 2011.

DAVISON, D. A. In: _____. **Java Prog. Techniques for Games**. Vienna: Universidade Prince of Songkla, Thailândia, 2013. Disponível em: <hhttp://fivedots.coe.psu.ac.th/~ad/index.html>.

ELECTRICAL, I. of; IEEE, E. E. Face recognition: eigenface, elastic matching, and neural nets. **Proceedings of the IEEE ( Volume: 85, Issue: 9, Sep 1997 )**, 1997. Disponível em: <http://ieeexplore.ieee.org/document/628712/>.

EYMAR LAéRCIO, R. Teoria : Processamento de imagens. **Instituto Nacional de Pesquisas Espaciais - INPE**, 2018. Disponível em: <http://www.dpi.inpe.br/spring/teoria/realce/realce.htm>. Acesso em: 05.2018.

FACHIN, O. In: _____. **Fundamentos De Metodologia - 6a Ed. 2017**. [S.l.]: Saraiva, 2017.

FARIAS, F. Como compactar imagens sem perder qualidade e aumentar a velocidade de seu site. **Resultados Digitais**, 2017. Disponível em: <https://resultadosdigitais.com.br/blog/compactar-imagens-sem-perder-qualidade/>. Acesso em: 05.2018.

GASPAROTTO, H. M. Os 4 pilares da programação orientada a objetos. Devmedia, 2014. Disponível em: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 06.2018.

GIT. 2018. Disponível em: <https://git-scm.com/>. Acesso em: 05.2018.

GONZALEZ, R. C. W. R. C. In: _____. **Processamento Digital de Imagens 3ªed**. São Paulo: Pearson Prentice Hall, 2010.

GROFFE, R. J. Artigo modelagem de sistemas através de uml: uma visão geral. Devmedia, 2013. Disponível em: <https://www.devmedia.com.br/modelagem-de-sistemas-atraves-de-uml-uma-visao-geral/27913>. Acesso em: 06.2018.

INTRONA, H. N. L. D. Facial recognition technology - a survey of polycy and implementation issues. 2010. Disponível em: <https://www.nyu.edu/projects/nissenbaum/papers/facial_recognition_report.pdf>.

JEBARA, T. S. **3D Pode Estimation and Normalization for Face Recognition**. Dissertação (Mestrado) — McGill University, Columbia University, Montréal, Québec - Canada, 1995. Disponível em: <http://www.cs.columbia.edu/~jebara/htmlpapers/UTHESIS/node7.html>.

JOHNSON, S. In: _____. **Stephen Johnson on Digital Photography**. [S.l.]: O'Reilly Media, Incorporated, 2006.

KIM JOON HYUNG SHIM, J. Y. I. **Final Project - Face Detection Project**. Dissertação (Mestrado) — Stanford University, Stanford, California, 2003. Disponível em: <https://web.stanford.edu/class/ee368/Project_03/Project/reports/ee368group02.pdf>.

KITANI, C. E. T. E. C. Um tutorial sobre análise de componentes principais para o reconhecimento automático de faces. **Departamente de Engenharia Elétrica**, Centro Universitário da FEI. Disponível em: <http://fei.edu.br/~cet/Tutorial_ReconhecimentoFaces.pdf>. Acesso em: 05.2018.

MOREIRA, G. C. G. **Reconhecedor de Objetos em Vídeos Digitais para Aplicações Interativas**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro - PUCRJ, Rio de Janeiro, 2008. Disponível em: <https://www.maxwell.vrac.puc-rio.br/13069/13069_1.PDF>.

MéTODO. Dicio, 2018. Disponível em: <https://www.dicio.com.br/metodo/>. Acesso em: 06.2018.

OPENCV - Open Source Computer Vision. 2018. Disponível em: <https://opencv.org/about.html>. Acesso em: 05.2018.

PERRY, J. S. Aprenda: Tecnologia java. IBM - Developer Works, 2016. Disponível em: <https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/index.html>. Acesso em: 06.2018.

PORTAL, S. T. S. Face recognition software comparison as recognition. **Digital Economy Compass 2017**, 2017. Disponível em: <https://www.slideshare.net/statista_com/statista-digital-economy-compass-2017>.

ROELOFS, G. A basic introduction to png features. **PNG LIB**, 2017. Disponível em: <http://www.libpng.org/pub/png/pngintro.html>. Acesso em: 05.2018.

SCIENCE, N.; (NSTC), T. C. Face recognition. **National Science and Technology Consil (NSTC)**, National Science and Technology Consil (NSTC), p. 92–99, 2009. Disponível em: <https://www.fbi.gov/file-repository/about-us-cjis-fingerprints_biometrics-biometric-center-of->. Acesso em: 04.2018.

SILVA, G. N. da. **Estudo das Técnicas PCA (Análise de Componentes Principais) e Auto-faces Aplicadas ao Reconhecimento de Faces Humanas**. Dissertação (Mestrado) — Centro Universitário Eurípides Soares - UNIVEM, Marília, 2009. Disponível em: <http://aberto.univem.edu.br/bitstream/handle/11077/284/Estudo+da+t%E9cnica+PCA+(An%E1lise+de+Componentes+Principais)+e+Auto-faces+aplicadas+ao+reconhecimento+de+faces+humanas.pdf;jsessionid=2D900F25381260DCCB85EC832736A216?sequence=1>.

SMITH, L. I. A tutorial on principal components analysis. 2002. Disponível em: <http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf>. Acesso em: 05.2018.

SMITH, L. I. Vídeo digital. 2018. Disponível em: <https://br.ccm.net/contents/739-video-digital>. Acesso em: 05.2018.

VELASQUEZ, H. C. Advanced vision module. **The University of Edinburgh**, 2017. Disponível em: <http://www.inf.ed.ac.uk/teaching/courses/av/index.html>. Acesso em: 05.2018.

VERGE.COM, T. • baidu swaps tickets for facial recognition in historic chinese 'water town'. novembro 2016. Disponível em: <https://www.theverge.com/2016/11/17/13663084/baidu-facial-recognition-park-wuzhen>.

WANGENHEIM, D. A. von. Transformações geométricas 2d e coordenadas homogêneas. **Image Processing and Computer Graphcs Lab - LAPiX**, Universidade Federal de Santa Catarina – UFSC, 2017. Disponível em: <http://www.lapix.ufsc.br/ensino/computacao-grafica/transformacoes-geometricas-2d-e-coordenadas-homogeneas>. Acesso em: 05.2018.

WóJCIK KONRAD GROMASZEK, M. J. W. Face recognition: Issues, methods and alternative applications. 2016. Disponível em: <https://www.intechopen.com/books/face-recognition-semisupervised-classification-subspace-projection-and-evaluation-methods/face-recognition-issues-methods-and-alternative-applications>.

YANG PING LUO, C. C. L. e. X. T. S. Wider face: A face recognition benchmark. 2015. Disponível em: <https://arxiv.org/pdf/1511.06523.pdf>.