UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Rafael Augusto Barbieri Lobato

Implementação de um algoritmo de visão computacional para localização de centros de olhos

Blumenau
2022

Rafael Augusto Barbieri Lobato

Implementação de um algoritmo de visão computacional para localização de centros de olhos

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Marcos Vinicius Matsuo, Dr.

Blumenau
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lobato, Rafael Augusto Barbieri
Implementação de um algoritmo de visão computacional para localização de centros de olhos / Rafael Augusto Barbieri Lobato ; orientador, Marcus Vinicius Matsuo, 2022.
83 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2022.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Visão Computacional. 3. Centros de olhos. 4. Aprendizado de Máquina. I. Matsuo, Marcus Vinicius. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Rafael Augusto Barbieri Lobato

Implementação de um algoritmo de visão computacional para localização de centros de olhos

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 22 de Setembro de 2022.

Banca Examinadora:

Documento assinado digitalmente
Marcos Vinicius Matsuo
Data: 29/09/2022 13:11:04-0300
CPF: ***.580.029-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Marcos Vinicius Matsuo, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 29/09/2022 14:25:54-0300
CPF: ***.999.369-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Maiquel de Brito
Data: 30/09/2022 09:58:00-0300
CPF: ***.657.150-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Maiquel de Brito, Dr.
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, por todo o apoio incondicional fornecido ao longo de toda a minha jornada pela universidade. Agradeço também aos meus amigos de faculdade e minha namorada Ana Clara, por sempre estarem ao meu lado fornecendo contínuo suporte e motivação. Por fim, agradeço à meu professor e orientador Marcos Vinícius Matsuo, cujo conhecimento, auxílio e aconselhamentos foram imprescindíveis para a realização desse trabalho.

O rosto é a imagem da alma e os olhos são os seus intérpretes (Marco Túlio Cícero)

RESUMO

A localização de olhos em imagens digitais consiste de uma importante etapa de processamento em diversos sistemas que utilizam visão computacional, que variam desde aplicações na indústria do entretenimento até aplicações que auxiliam na inclusão de pessoas com deficiência. O presente trabalho descreve o desenvolvimento de um algoritmo no software Matlab, utilizando técnicas de visão computacional e aprendizado de máquina, para realizar a localização de centros de olhos em uma imagem que contenha um rosto humano. Inicialmente, é realizada uma revisão teórica dos principais conceitos utilizados no desenvolvimento e então uma apresentação de todas as etapas de processamento do algoritmo implementado. No desenvolvimento essas etapas são detalhadas, desde os estágios de identificação do rosto e características críticas dos olhos até o treinamento de um classificador SVM responsável por localizar os centros de olhos. Ao final, o algoritmo implementado é testado utilizando diferentes bancos de imagens com seus resultados sendo avaliados e discutidos.

Palavras-chave: Visão Computacional; Centros de olhos; Machine Learning.

ABSTRACT

Locating eyes in digital images consists of an important processing step in several systems that use computer vision, ranging from applications in the entertainment industry to applications that assist in the inclusion of people with disabilities. The present work describes the development of an algorithm in Matlab software, using computer vision and machine learning techniques, to find the location of eye centers in an image that contains a human face. Initially, a theoretical review of the main concepts used in the development is carried out and then a presentation of all the processing steps of the implemented algorithm. These steps are then further detailed, from the stages of face identification and critical eye features to the training of an SVM classifier responsible for locating eye centers. At the end, the implemented algorithm is tested using different image datasets with their results being evaluated and discussed.

Keywords: Computer Vision; Eye Centers; Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CMI	Cell Mean Intensity
HOG	Histogram of Oriented Gradients
LBP	Local Binary Pattern
NSE	Negative Slope Edge
PSE	Positive Slope Edge
sCES	semi-Circular Edge Slope
sEES	semi-Eliptical Edge Slope
SVM	Support Vector Machine

SUMÁRIO

1 INTRODUÇÃO

Em diversas aplicações de visão computacional (envolvendo, por exemplo, rastreamento de olhos e reconhecimento facial) é necessário localizar em alguma etapa do processamento os olhos presentes nas imagens. A correta identificação de olhos em imagens digitais permite introduzir soluções ou melhorias a uma gama considerável de aplicações. Por exemplo, tal tecnologia pode ser utilizada no desenvolvimento de jogos, com o objetivo de amplificar a imersão de seus jogadores, permitindo uma rápida interação com a aplicação (CORCORAN *et al.*, 2012). Em particular, a localização de olhos pode auxiliar também na questão da acessibilidade e inclusão de pessoas com deficiência. Como exemplo, pode-se citar o dispositivo chamado de *mouse ocular*, que permite que usuários sejam capazes de navegar por um computador apenas utilizando os olhos (TRINDADE, F. H. V., 2018). Outro exemplo bastante interessante, onde a localização de olhos é necessária é em um sistema de monitoramento facial para identificação de motoristas sonolentos que dispara alarmes e ativa dispositivos de segurança assim que detectado que o motorista encontra-se em estado de sonolência (CHANG; CHEN, 2014).

De modo geral, localizar olhos por meio de técnicas de visão computacional é uma tarefa desafiadora. As imagens contendo rostos estão suscetíveis a diversas variáveis que podem dificultar a correta identificação dos olhos, como por exemplo a variação de luminosidade do ambiente, resolução da imagem, variações de escala em rostos ou posição da face, as quais podem dificultar a correta localização dos olhos. Existe também a questão fisiológica, há muita variabilidade no formato e tamanho dos olhos, variando de indivíduo a indivíduo (FARKAS *et al.*, 2005).

Em particular, há diferentes abordagens propostas na literatura para localização de olhos. Algumas dessas abordagens buscam identificar os olhos com base em uma modelagem matematicamente precisa de seu formato (VILLANUEVA; CABEZA; PORTA, 2006), outras baseiam-se em uma estratégia de segmentação da região do olho por meio de análise de cores (SKODRAS; FAKOTAKIS, 2015), ou ainda há outras abordagens que se utilizam de características topográficas dos olhos para sua detecção (VILLANUEVA *et al.*, 2013).

Neste contexto, este trabalho tem como foco a implementação de um algoritmo de localização dos olhos de um rosto humano por meio do reconhecimento de características geométricas semi-elípticas das pálpebras e semi-circulares da íris, baseado no trabalho desenvolvido em (AHMED; LASKAR, 2019). Essa abordagem foi escolhida pois essas são características que se preservam bem diante de muitas das variabilidades de uma imagem, principalmente em questões de variabilidade de luz e resolução de imagem. Inicialmente, a intenção deste trabalho era a reprodução fiel do algoritmo descrito em (AHMED; LASKAR, 2019), porém durante o desenvolvimento foi necessário complementar o trabalho original com implementações e etapas de processamento não descritas inicialmente, a fim

de simplificar determinadas etapas do processo ou devido à escassez de informações de referentes à determinadas etapas de processamento. Especificamente, dada uma imagem digital contendo um rosto humano, a ideia do algoritmo implementado é a realização das seguintes etapas: *i*) identificação do rosto e das características semi-elípticas e semi-circulares presentes para a seleção de possíveis candidatos a pares de olhos; *ii*) classificação dos candidatos a olhos através de um classificador SVM (*Support Vector Machine*), previamente treinado; e *iii*) definição de um único par de olhos, obtido a partir dos múltiplos candidatos de pares de olhos identificados previamente.

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo geral a implementação de um algoritmo de visão computacional, baseado em (AHMED; LASKAR, 2019), que seja capaz de identificar as coordenadas dos centros dos olhos de um rosto humano presente em uma imagem digital.

1.2 OBJETIVOS ESPECÍFICOS

Este trabalho almeja atingir os seguintes objetivos específicos:

- Compreender as etapas de processamento necessárias para localização de centros de olhos em imagens.
- Implementar cada etapa de processamento avaliando os resultados obtidos.
- Avaliar o desempenho do algoritmo desenvolvido em diferentes *datasets* contendo imagens de rostos.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este documento está organizado como segue. O Capítulo 2 apresenta uma revisão dos principais conceitos teóricos utilizados como base para o desenvolvimento deste trabalho. O Capítulo 3 aborda todo o desenvolvimento do algoritmo de localização de centro de olho implementado. No Capítulo 4 são discutidos os resultados obtidos pelo algoritmo implementado. Finalmente, no Capítulo 5 são apresentadas as conclusões do trabalho, além de propostas de melhorias e desenvolvimentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção tem como finalidade introduzir definições e conceitos teóricos fundamentais, de forma a facilitar e auxiliar na compreensão do algoritmo desenvolvido.

2.1 IMAGEM DIGITAL

Uma imagem digital consiste em uma representação de uma imagem bidimensional através da codificação de números binários ou valores discretos (CORKE, 2017). Em particular, existem dois tipos principais de imagem digital, a saber: imagem *vetorial* e imagem *raster*.

Uma imagem digital vetorial é gerada a partir de elementos geométricos primitivos, como retas ou curvas, cujas características como magnitude, comprimento ou orientação podem ser facilmente manipuladas e alteradas a fim de se obter um resultado final desejado (CONGALTON, 1997). Esse tipo de imagem não perde qualidade com sua ampliação, pois é redimensionável devido à em sua composição primitiva geométrica.

As imagens do tipo *raster* são compostas por uma matriz densa de pequenos elementos denominados *pixels*, onde cada *pixel* é um elemento primitivo, com propriedades distintas que lhes conferem diferentes níveis de intensidade (FIUME, 1989). As imagens *raster* podem ainda ser classificadas como binária, escala de cinza (ou *grayscale*) e colorida. Os *pixels* podem ser quantificados de diversas formas, porém mais comumente seguem a escala de valores baseada no tipo de dado *UINT8* (inteiro sem sinal de 8 bits), podendo assumir valores no intervalo 0 até 255; ou ainda, podem ser quantificados utilizando o tipo de dado *double*, podendo neste caso assumir valores de $0,0$ a $1,0$.

2.1.1 Imagens binárias e em Escala de Cinza

Uma imagem do tipo *raster* binária é aquela cujos *pixels* que a compõem podem assumir apenas dois valores, isto é, 0 (denotando preto) e 1 (denotando branco), considerando dados do tipo *double*. Na Figura 1(a) é apresentado um exemplo de imagem binária.

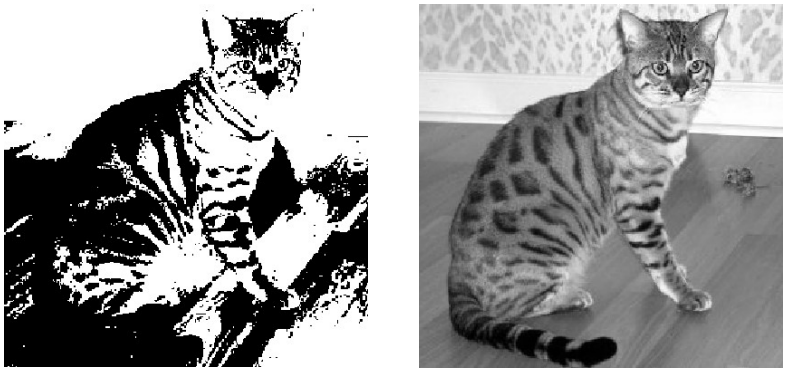
Já uma imagem do tipo *raster* em escala de cinza é similar à imagem binária, porém os *pixels* podem assumir quaisquer valores dentro do intervalo limitado pelo tipo de dado, o que concede à imagem uma maior acurácia na representação de seus detalhes (Veja Figura 1(b)).

2.1.2 Imagem Colorida

Ao se tratar de imagens binárias e imagens em escala de cinza, considera-se apenas uma matriz no qual os *pixels* compositores da imagem estão agrupados, com seus respectivos valores. Entretanto, para representar uma imagem colorida, são necessários

Figura 1 – Imagem Digital

(a) Imagem binária. (b) Imagem em escala de cinza.



Fonte: The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

três matrizes de igual tamanho, de forma a utilizar informação contida em cada uma para a formação da imagem colorida. Em outras palavras, cada *pixel* da imagem colorida é composto por três valores, cada um contido em uma matriz diferente.

Especificamente, as informações representadas em cada matriz dependem do espaço de cor utilizado para representação da imagem colorida. Dentre os espaços de cores mais comuns destacam-se os espaços RGB e HSV.

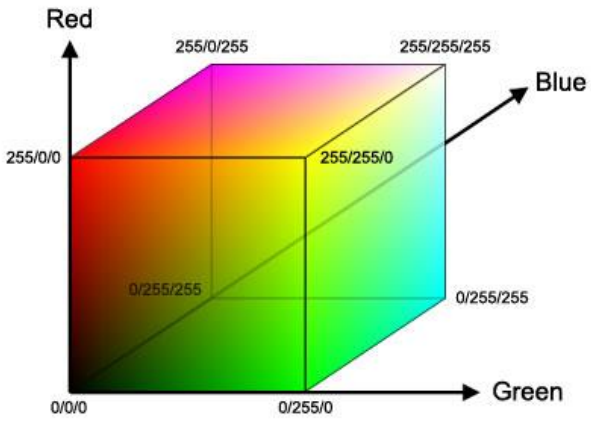
2.1.2.1 Espaço RGB

No espaço RGB cada cor é representada como uma combinação das cores vermelho (R), verde (G) e azul (B). Dessa forma, um único pixel de uma imagem colorida é constituído por uma intensidade de vermelho, verde e azul, o que lhe confere uma caracterização única de cor. O valor de cada componente RGB varia de 0 a 255 considerando dados do tipo UINT8. A representação do espaço de cor RGB é ilustrada na Figura 2, onde é possível observar diversas cores geradas a partir da combinação de diferentes níveis de vermelho (R), verde (G) e azul (B).

2.1.2.2 Espaço HSV

Diferentemente do espaço RGB, que utiliza as cores primárias para a composição de outras cores, o espaço HSV busca codificar as cores de uma forma mais próxima de como as cores são percebidas aos humanos. No espaço de cor HSV cada cor é definida pelos valores H (*Hue*, ou Matiz), S(Saturação) e V(*Value*, ou Intensidade). Especificamente, a Matiz (cujo valor pode variar de 0 até 360) define a característica cromática da cor; a Saturação (cujo valor pode variar de 0 a 1) define a pureza da cor, isto é, oferece uma medição do quanto a cor original foi diluída por branco; e a Intensidade (cujo valor pode variar de 0 até 1) é análoga à brilho, onde seus valores mínimo e máximo representam maior distância

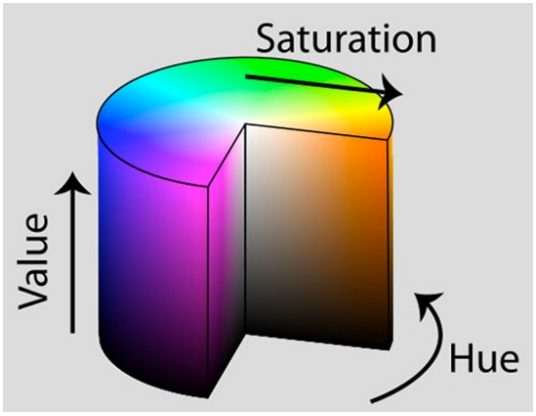
Figura 2 – Modelo RGB esquematizado.



Fonte: (MAIA; TRINDADE, R., 2016)

das cores branca e preta, respectivamente, (CHERNOV; ALANDER; BOCHKO, 2015), como ilustrado pela Figura 3.

Figura 3 – Modelo HSV esquematizado.



Fonte: CorelDRAW

O espaço HSV é bastante utilizado em aplicações onde se deseja realizar a segmentação de uma imagem com base nas cores dos objetos que a compõem, devido à clara distinção entre crominância e intensidade no modelo. Uma aplicação desse modelo é a segmentação (GHAZALI; MA; XIAO, 2011) ou detecção (RAHMAN; EDY PURNAMA; PURNOMO, 2014) da pele humana.

2.2 PROCESSAMENTO DE IMAGENS

Processamento de imagem é um processo computacional que transforma uma ou mais imagens de entrada em uma imagem de saída (CORKE, 2017). Especificamente

envolve submeter imagens a uma série de operações matemáticas, de forma a manipular os valores e/ou as características espaciais de seus *pixels*.

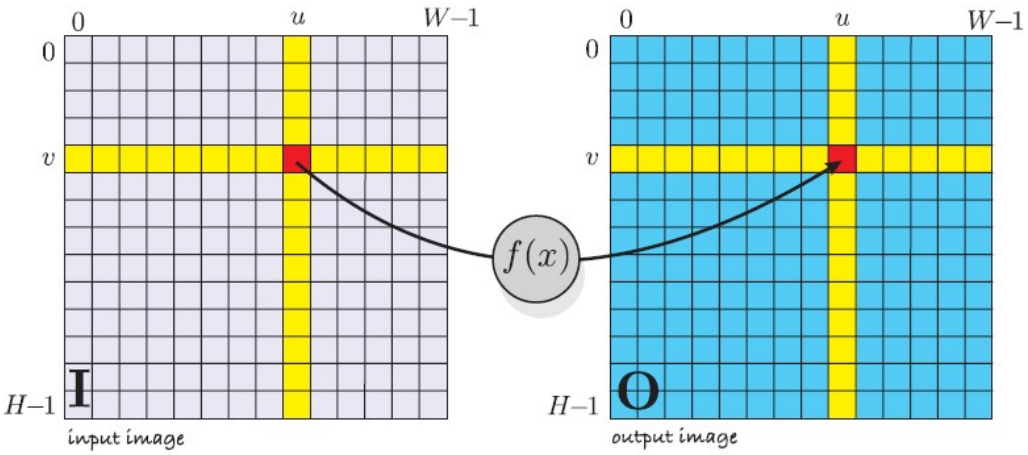
2.2.1 Operações Monádicas

Uma operação de transformação monádica é aquela onde cada pixel de coordenada (u, v) da imagem de saída **O** é resultado do processamento do correspondente pixel (u, v) da imagem de entrada **I**. Matematicamente, uma operação monádica pode ser expressa como

$$\mathbf{O}[u, v] = f(\mathbf{I}[u, v]), \forall (u, v) \in \mathbf{I} \tag{1}$$

onde $f(\cdot)$ é a função de transformação. Conforme ilustrado na Figura 4 é possível observar que a operação é realizada *pixel* a *pixel*.

Figura 4 – Operação de processamento monádica.



Fonte: (CORKE, 2017)

Consistem de operações monádicas, por exemplo, a conversão de uma imagem do tipo *double* para *UINT8* e vice-versa, descrita pelas funções de transformação

$$f_{double \to uint8}(x) = 255x \tag{2}$$

$$f_{uint8 \to double}(x) = \frac{x}{255} \tag{3}$$

onde x é o valor do *pixel*.

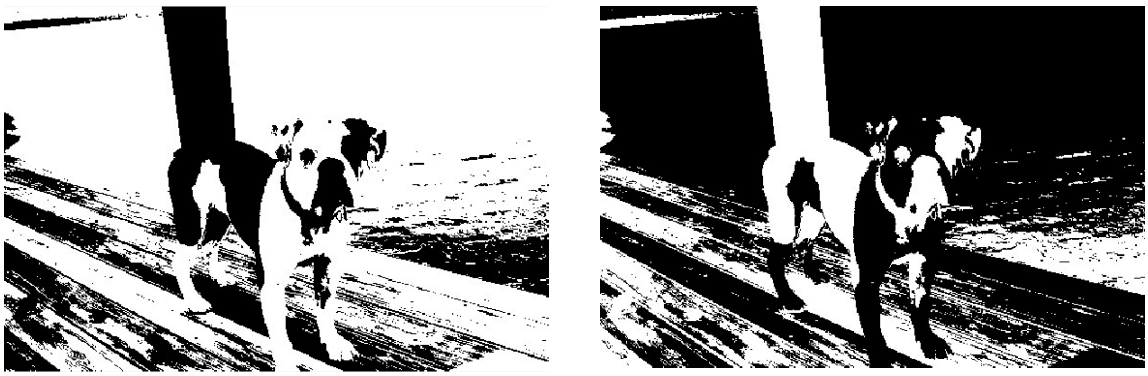
É possível também extrair o negativo de uma imagem por meio de uma operação monádica, transformando todos os *pixels* da imagem de entrada em seu valor oposto na imagem de saída. Em uma imagem *double* binária, a função de transformação é dada por

$$f_{neg}(x) = 1 - x \tag{4}$$

onde x é o valor do *pixel*. O resultado desta operação está ilustrado pela Figura 5.

Figura 5 – Comparação entre imagem binária e imagem negativa

(a) Imagem binária. (b) Imagem após transformação monádica



Fonte: The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

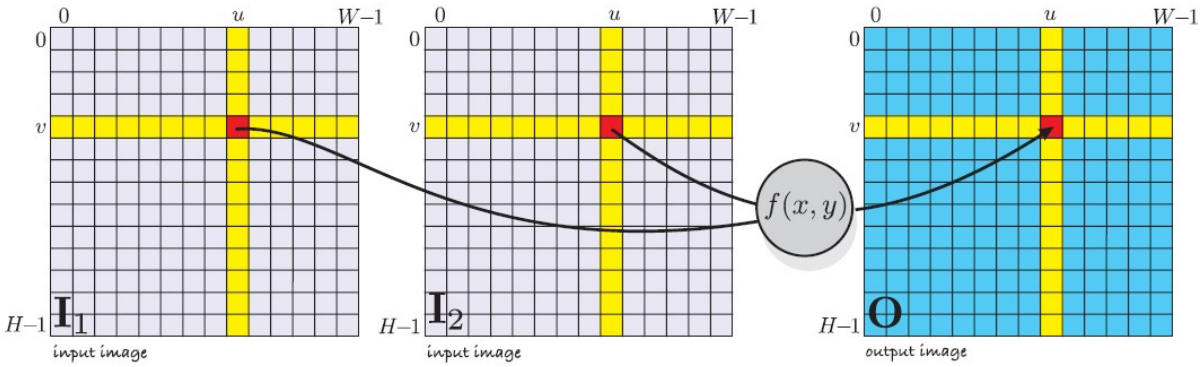
2.2.2 Operações Diádicas

Uma operação de transformação diádica é aquela onde cada pixel de coordenada (u, v) da imagem de saída $\mathbf{O}$ é resultado do processamento do correspondente pixel (u, v) de duas imagens de entrada distintas $\mathbf{I}_1$ e $\mathbf{I}_2$. Matematicamente, uma operação diádica pode ser expressa como

$$\mathbf{O}[u, v] = f(\mathbf{I}_1[u, v], \mathbf{I}_2[u, v]), \forall (u, v) \in \mathbf{I}_1, \mathbf{I}_2 \tag{5}$$

onde $f(\cdot, \cdot)$ é a função de transformação. Diferentemente das operações diádicas, a imagem final $\mathbf{O}$ resulta do processamento *pixel* a *pixel* entre duas imagens, como ilustra a Figura 6.

Figura 6 – Operação de processamento diádica.



Fonte: (CORKE, 2017)

As operações diádicas podem ser, por exemplo, a soma, subtração ou multiplicação ponto a ponto dos *pixels* de uma matriz, descrita pelas funções de transformação:

$$f(\mathbf{I}_1[u,v],\mathbf{I}_2[u,v]) = \mathbf{I}_1[u,v] + \mathbf{I}_2[u,v], \forall (u,v) \in \mathbf{I}_1,\mathbf{I}_2 \quad (6)$$

$$f(\mathbf{I}_1[u,v],\mathbf{I}_2[u,v]) = \mathbf{I}_1[u,v] - \mathbf{I}_2[u,v], \forall (u,v) \in \mathbf{I}_1,\mathbf{I}_2 \quad (7)$$

$$f(\mathbf{I}_1[u,v],\mathbf{I}_2[u,v]) = \mathbf{I}_1[u,v] * \mathbf{I}_2[u,v], \forall (u,v) \in \mathbf{I}_1,\mathbf{I}_2 \quad (8)$$

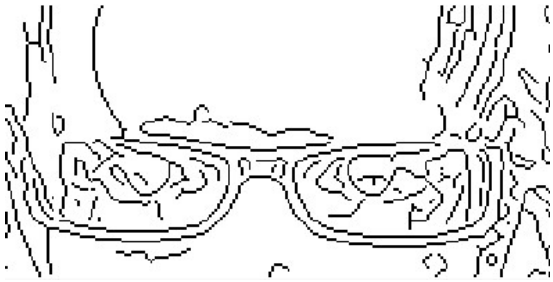
Possíveis aplicações dessas operações envolvem a extração de fundo colorido (conhecido por *Chroma Key*) e fusão de imagens.

A Figura 7 ilustra um exemplo de fusão de imagens, onde as principais características de borda de um rosto (Figura 7(b)) sofrem uma operação diádica de multiplicação *pixel* a *pixel* com a imagem original (Figura 7(a)) a fim de realçar as bordas detectadas (Figura 7(c)).

Figura 7 – Sobreposição de imagens

(a) Imagem de rosto

(b) Principais traços de borda do rosto



(c) Imagem resultante da operação diádica

Fonte: GI4E Database - Universidad Pública de Navarra

2.2.3 Transformações Geométricas

Uma transformação geométrica, caracterizada como uma transformação espacial de imagem, modifica o arranjo espacial dos *pixels* em uma imagem. (GONZALEZ; WOODS, 2008). Em outras palavras, em uma transformação geométrica cada *pixel* de coordenada (u, v) de uma imagem de entrada I são mapeados para uma nova posição (u', v') na imagem

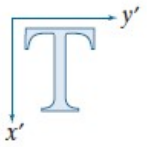
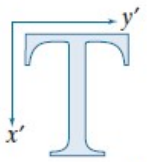
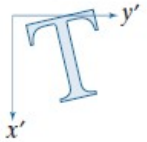
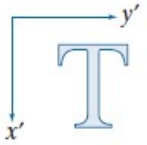
de saída **O**, através de uma matriz de mapeamento **A**. Matematicamente, tem-se

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{9}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Especificamente, os valores dos elementos da matriz **A** definem a transformação geométrica a ser sofrida pelo par de coordenadas (u, v). A Figura 8 apresenta a matriz de transformação de algumas das principais transformações (identidade, reflexão, rotação e translação), juntamente com exemplos dessas transformações.

Figura 8 – Tabela de Transformações Geométricas. Considere a notação $x = u$ e $y = v$

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to −1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	

Fonte: (GONZALEZ; WOODS, 2008)

2.2.4 Filtragem espacial

Uma operação de filtragem modifica a imagem digital por meio da substituição dos valores de cada *pixel* por uma função dos valores deste *pixel* e de seus vizinhos (GONZALEZ; WOODS, 2008).

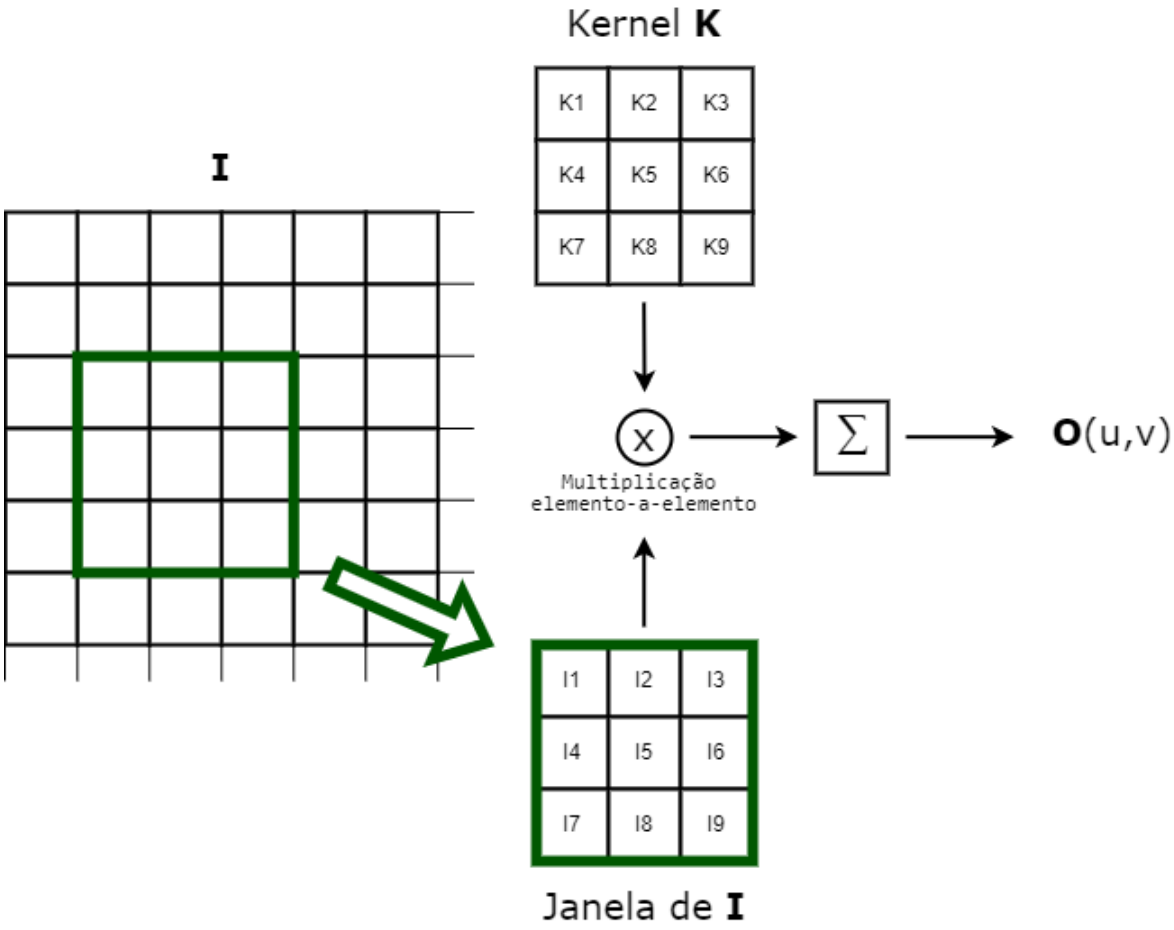
Em uma operação de filtragem espacial, cada *pixel* (u, v) da imagem de saída **O** é o resultado da soma dos produtos entre os elementos de um kernel (ou filtro) **K** com um

conjunto de *pixels* (dentro de uma janela centrada na coordenada (u, v)) da imagem de entrada **I**. Matematicamente, tem-se

$$\mathbf{O}(u,v) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} K(s,t)\mathbf{I}(u+s, v+t) \tag{11}$$

onde $(a, b) \neq 0$ e $\mathbf{O}(u, v)$ é a posição espacial de um *pixel* na imagem filtrada. Na Figura 9 é ilustrado o processo de filtragem espacial, descrito por (11).

Figura 9 – Ilustração do processo de filtragem



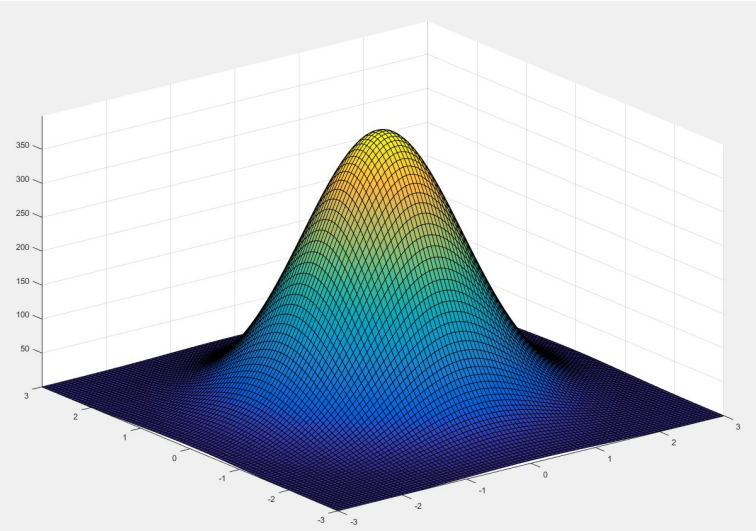
Fonte: O autor

Um exemplo de aplicação de filtragem espacial é a redução da definição de uma imagem por meio da aplicação de um kernel gaussiano, cujos elementos são computados como

$$K(s,t) = \frac{1}{2\pi\sigma^2} e^{\frac{-(s^2+t^2)}{2\sigma^2}} \tag{12}$$

onde σ é o desvio padrão da distribuição gaussiana, que pode ser observada na Figura 10. A finalidade deste kernel, que constitui o filtro passa-baixa Gaussiano, é gerar na imagem resultante um efeito de borrão, deixando-a menos nítida. É possível observar na Figura 11 um comparativo entre uma figura e sua contraparte filtrada por um filtro Gaussiano.

Figura 10 – Superfície de distribuição Gaussiana



Fonte: O autor

Figura 11 – Filtragem utilizando kernel Gaussiano

(a) Imagem em escala de cinza (b) Imagem pós-filtragem

Fonte: The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

Outra aplicação de filtragem espacial é a extração de gradientes horizontais e verticais de uma imagem, isto é, a variação na intensidade dos *pixel* de uma imagem para uma determinada direção.

Os filtros mais utilizados para computar gradientes horizontais e verticais são os filtros de Sobel (CORKE, 2017), matematicamente definidos como

$$\mathbf{I}_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{13}$$

$$\mathbf{I}_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{14}$$

onde $\mathbf{I}_u$ e $\mathbf{I}_v$ representam as imagens contendo vetores gradientes na orientação horizontal e vertical, respectivamente (Veja Figura 12(b) e 12(c)).

2.2.5 Operações morfológicas

Operações morfológicas, são operações espaciais que processam uma imagem de entrada binária $\mathbf{I}$, tendo como objetivo a alteração das formas das regiões em branco presentes da imagem. Para tal, utiliza-se um elemento estruturante $\mathcal{S}$, que corresponde a uma matriz binária de dimensões arbitrárias, com um *pixel* definido como origem (veja Figura 14) (CORKE, 2017). Matematicamente, uma operação morfológica pode ser definida como

$$\mathbf{O}[u, v] = f(\mathbf{I}[u + i, v + j]), \quad \forall (i, j) \in \mathcal{S}, \forall (u, v) \in \mathbf{I} \tag{15}$$

onde $\mathbf{O}$ é a imagem de saída após a operação.

As operações morfológicas consistem, primeiramente, em transladar o elemento estruturante $\mathcal{S}$ sobre a imagem de entrada $\mathbf{I}$, utilizando como referência o seu *pixel* de origem. Para cada coordenada $\mathbf{I}(u, v)$, os *pixels* de $\mathbf{I}$ e $\mathcal{S}$ sobrepostos naquela região serão comparados elemento a elemento, e geram na imagem resultante de saída $\mathbf{O}(u, v)$ um valor de 0 ou 1. Esse valor depende da quantidade de *pixels* sobrepostos entre $\mathbf{S}$ e $\mathbf{I}$ na região analisada e também da operação morfológica sendo realizada. Em particular, as operações morfológicas mais comuns são a erosão e dilatação, as quais são explicadas com mais detalhes nas seções seguintes.

2.2.5.1 Erosão

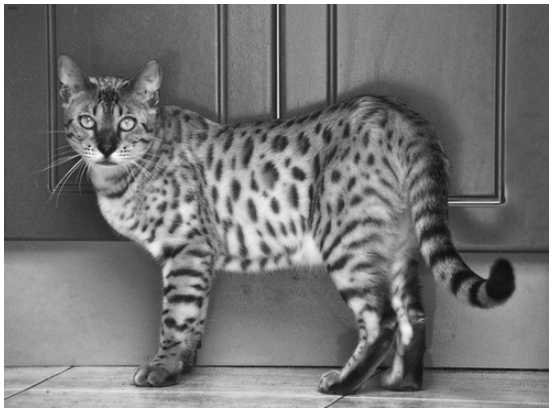
Matematicamente, a erosão da imagem de entrada $\mathbf{I}$ pelo elemento estruturante $\mathcal{S}$ é representada pela equação

$$\mathbf{O} = \mathbf{I} \ominus \mathcal{S} \tag{16}$$

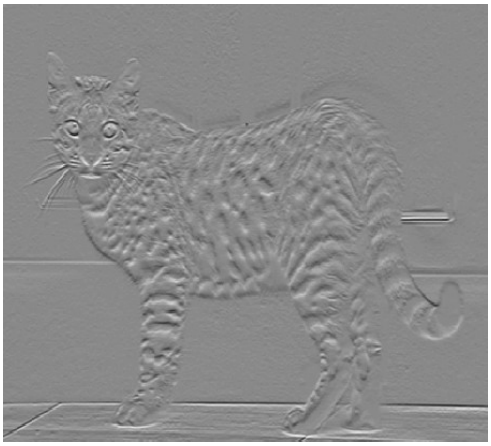
onde $\mathbf{O}$ denota a imagem de saída. Note que as dimensões das imagens $\mathbf{O}$ e $\mathbf{I}$ são idênticas.

Figura 12 – Filtragem de imagem utilizando kernels de Sobel.

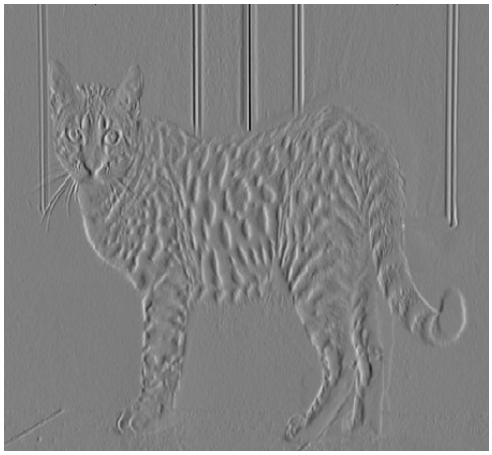
(a) Imagem original **I**



(b) Imagem $\mathbf{I}_u$, contendo gradientes horizontais



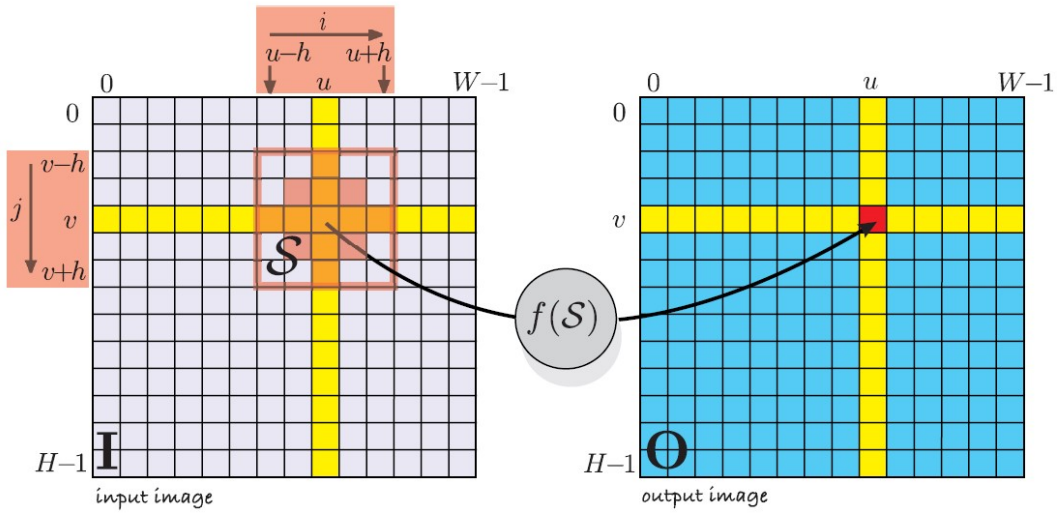
(c) Imagem $\mathbf{I}_v$, contendo gradientes verticais



Fonte: Adaptado de The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

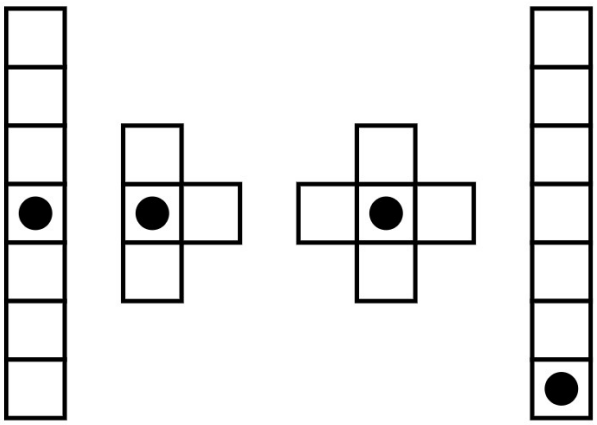
Um exemplo de resultado da operação morfológica de erosão é ilustrado na Figura 15(b), onde a imagem de entrada da Figura 15(a) é processada utilizando um elemento estruturante quadrado de 15×15 *pixels*. Especificamente, na operação de erosão o *pixel* (u, v) da imagem de saída **O** recebe valor 1, somente se todos os pixels em branco do elemento estruturante (centralizado) na coordenada (u, v) estiverem sobrepostos a pixels brancos da imagem de entrada **I**. Tal operação, tem o efeito de reduzir as dimensões das regiões em branco da imagem de entrada, como pode ser observado analisando as Figuras 15(a) e 15(b). A erosão pode ser utilizada para remover ruídos indesejados de uma imagem ou eliminar características indesejadas, como observado na Figura 15(b).

Figura 13 – Processamento morfológico de uma imagem



Fonte: (CORKE, 2017)

Figura 14 – Exemplos de elementos estruturantes, com pontos de origem destacados



Fonte: O autor

2.2.5.2 Dilatação

Matematicamente, a dilatação da imagem $\mathbf{I}$ pelo elemento estruturante $\mathcal{S}$ é dada por

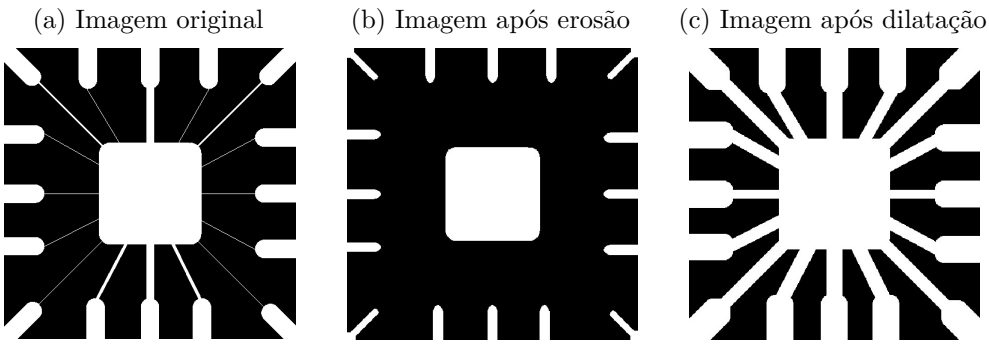
$$\mathbf{O} = \mathbf{I} \oplus \mathcal{S} \tag{17}$$

onde $\mathbf{O}$ denota a imagem de saída. Note novamente que as dimensões de $\mathbf{O}$ e $\mathbf{I}$ são idênticas.

Um exemplo de resultado da operação morfológica de dilatação é ilustrado na Figura 15(c), onde imagem de entrada da Figura 15(a) é processada utilizando um elemento estruturante de 15×15 *pixels*. Para a operação de dilatação, o *pixel* (u, v) da imagem

de saída **O** recebe valor 1 se ao menos um *pixel* em branco do elemento estruturante (centralizado) na coordenada (u, v) estiver sobreposto a um *pixel* branco na imagem de entrada **I**. Essa operação gera o efeito de aumentar as dimensões das regiões em branco da imagem de entrada, como pode ser analisado nas Figuras 15(a) e 15(c). Note que, na imagem de saída, a dilatação gera o efeito contrário da erosão. Enquanto a erosão visa a redução e a eliminação de partes indesejadas da imagem, a dilatação visa ampliar ou aumentar determinadas características dessas imagem.

Figura 15 – Operações morfológicas utilizando *kernel* quadrado 15×15



Fonte: Adaptado de (GONZALEZ; WOODS, 2008)

2.2.5.3 Transformada Hit-or-Miss

A transformada Hit-or-Miss (HMT) é um procedimento morfológico que utiliza uma versão alterada dos elementos estruturantes. Para essa transformada, os *pixels* de um elemento estruturante podem conter valor 0, 1 ou nulo (independente) (CORKE, 2017). Esse elemento modificado pode ser utilizado tanto para operações morfológicas de erosão quanto para operações morfológicas de dilatação. A Figura 16 ilustra o procedimento de dilatação utilizando HMT.

A HMT é eficaz na identificação de formas em uma imagem. Dada uma imagem $\mathbf{I}(u, v)$, a HMT pode ser definida como (GONZALEZ; WOODS, 2008):

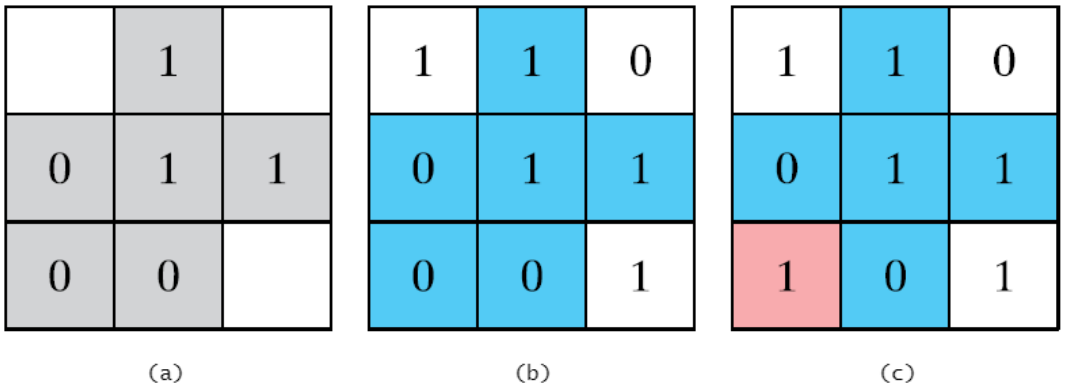
$$HMT(\mathbf{I}; \mathcal{S}) = (\mathbf{I} \ominus \mathcal{S}_1) \cap (\mathbf{I}^c \ominus \mathcal{S}_2)) \tag{18}$$

onde $\mathcal{S}_1$ é o elemento estruturante que percorre os *pixels* de primeiro plano (na imagem **I**) e $\mathcal{S}_2$ é o elemento estruturante que percorre os *pixels* de plano de fundo (na imagem $\mathbf{I}^c$). A Figura 17 ilustra esse procedimento.

2.2.6 Análise de componentes conectados

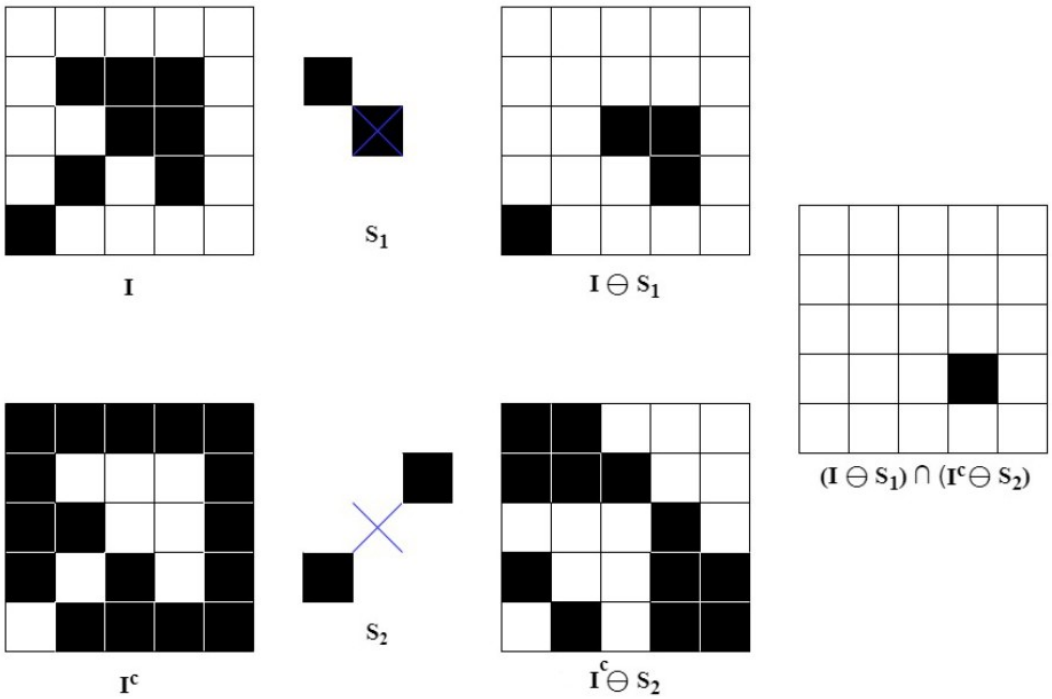
Componentes conectados podem ser definidos como regiões de *pixels* adjacentes que possuem o mesmo valor (SZELISKI, 2011). Existem duas maneiras de classificar a

Figura 16 – Elemento estruturante na Transformada Hit-or-Miss. (a) Elemento Estruturante com valores 0, 1 e nulos. (b) Sobreposição válida sobre uma janela de *pixels*. (c) Sobreposição não válida sobre uma janela de *pixels*.



Fonte: Adaptado de (CORKE, 2017)

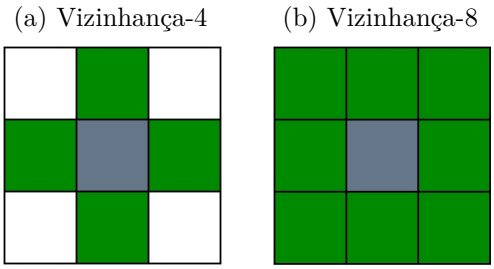
Figura 17 – Procedimento de Transformada Hit-or-Miss



Fonte: O autor, inspirado em (GONZALEZ; WOODS, 2008)

conectividade entre dois *pixels* em uma imagem 2D, isto é, determinar se eles são ou não adjacentes: conectividade por Vizinhança-4 ($\mathcal{N}^4$) (Figura 18a), que considera como adjacentes os *pixels* acima, abaixo, à esquerda e à direita do *pixel* analisado, e Vizinhança-8 ($\mathcal{N}^8$) (Figura 18b), onde todos os *pixels* vizinhos, inclusive diagonalmente, são considerados adjacentes.

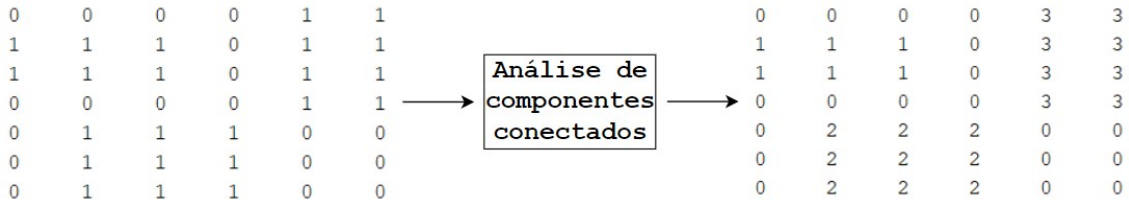
Figura 18 – Conectividade de *pixels*



Fonte: O autor

Por análise de componentes conectados entende-se o processo de rotular componentes conectados em uma imagem binária. Neste processo, os *pixels* de cada região em branco da imagem binária são rotulados com um número inteiro de modo diferenciar cada componente conectado. Um exemplo, de resultado de análise de componentes conectados é apresentado na Figura 19.

Figura 19 – Processo de rotulação de componentes conectados



Fonte: O autor

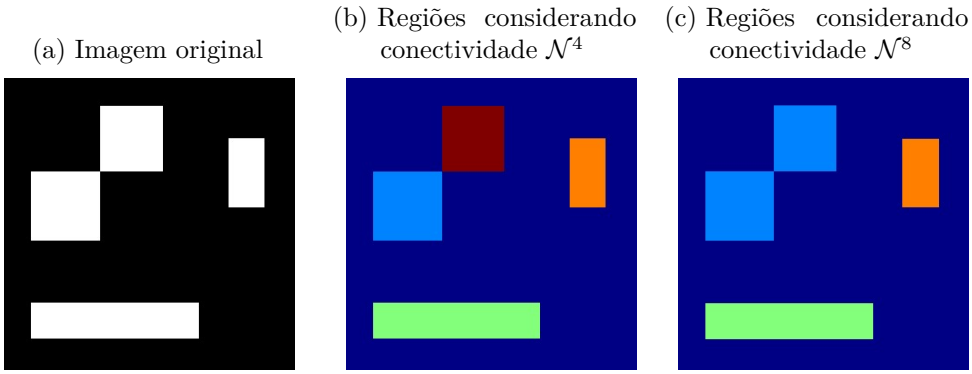
A seleção da conectividade dos *pixels* pode alterar a quantidade de componentes detectadas em uma dada imagem, como ilustra a Figura 20 (regiões de mesma coloração indicam um único componente conectado).

2.2.7 Algoritmo de Detecção de Bordas de Canny

As bordas dos objetos presentes em uma imagem podem ser identificados visualmente por meio de mudanças significativas de luz, textura ou cor (MARTIN; FOWLKES; MALIK, 2004). Uma abordagem para a definição identificação de borda é pela ocorrência de uma rápida variação de intensidade, o que pode ser matematicamente descrito pela variação do gradiente local (SZELISKI, 2011). Esse conceito é a base para diversos algoritmos de detecção de borda, que possuem como finalidade identificar os *pixels* de borda presentes em uma imagem.

O algoritmo de detecção de bordas de Canny, desenvolvido em 1986 por John Canny (CANNY, 1986), é dos algoritmos de detecção de bordas mais utilizados devido a

Figura 20 – Análise de componentes conectados em uma imagem



Fonte: O autor

sua baixa taxa de erros, boa localização de *pixels* de borda fortes e definição das bordas com apenas 1 *pixel* de largura.

Especificamente, para computar a borda de uma imagem através do algoritmo de Canny, inicialmente a imagem de entrada $\mathbf{I}$ é suavizada por um filtro gaussiano, como exemplificado pela Figura 11. À partir desse imagem, obtém-se o gradiente (magnitude e fase) da imagem suavizada, através das equações

$$\mathbf{M} = \sqrt{\mathbf{I}_u^2 + \mathbf{I}_v^2} \tag{19}$$

$$\alpha = tan^{-1}\left(\frac{\mathbf{I}_u^2}{\mathbf{I}_v^2}\right) \tag{20}$$

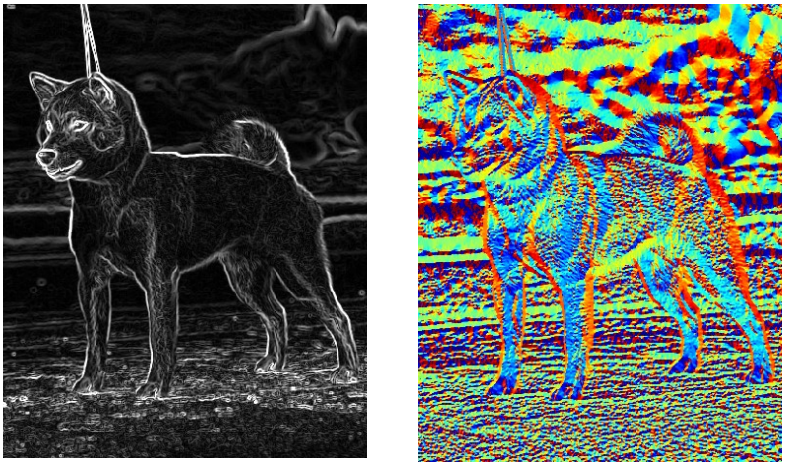
onde M é a magnitude, α é a fase e $\mathbf{I}_u$, $\mathbf{I}_v$ são as imagens representando os gradientes horizontais e verticais de $\mathbf{I}$, respectivamente (veja Seção 2.2.4). Na Figura 21 são apresentadas as imagens de magnitude (Figura 21(a)) e fase (Figura 21(b)) obtidas a partir do processamento da imagem suavizada apresentada na Figura 11(b)).

Uma vez computadas as imagens de magnitude $\mathbf{M}$ e fase α, cria-se uma imagem $\mathbf{G}_n$, com as mesmas dimensões de $\mathbf{I}$. Para cada *pixel* $\mathbf{I}(u, v)$, examina-se primeiramente a direção do seu gradiente (ou seja, a fase $\alpha(u, v)$), conforme ilustrado na Figura 22. Na sequência, verifica-se se a magnitude do gradiente do *pixel* em análise (ou seja, $\mathbf{M}(u, v)$) é maior do que a magnitude do gradiente dos vizinhos que estão na mesma direção da fase $\alpha(u, v)$, como ilustrado na Figura 22, onde fica evidente que os *pixels* vizinhos são os *pixels* $P3$ e $P7$. Se a magnitude do *pixel* em análise for maior do que a magnitude dos *pixels* vizinhos, então considera-se que na coordenada (u, v) existe possivelmente um *pixel* de borda, neste caso, faz-se $\mathbf{G}_n(u, v) = 1$ (caso contrário, tem-se $\mathbf{G}_n(u, v) = 0$). Esse processo é denominado de supressão de não máximos locais, com os *pixels* com valor 1 da imagem $\mathbf{G}_n$ (veja Figura 23) representando candidatos a *pixels* de borda.

Por fim, compara-se cada *pixel* de $\mathbf{G}_n$ com dois limiares (um alto e outro baixo), de forma a identificar *pixels* de borda considerados fortes ou fracos. Com a imagem de

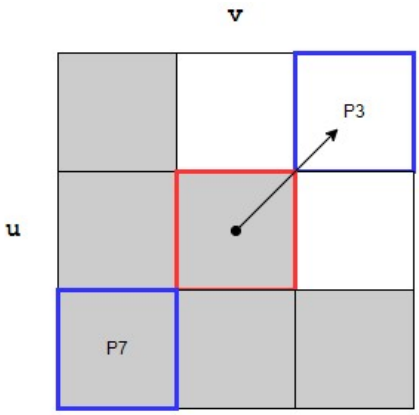
Figura 21 – Imagens de (a) e fase (b) obtidas a partir do cálculo do gradiente da imagem apresentada na Figura 11(b).

(a) Imagem de Magnitude (b) Imagem de Fase



Fonte: Modificado de The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

Figura 22 – Magnitude e fase de um gradiente em $\mathbf{I}(u, v)$



Fonte: O autor

de borda final sendo é constituída apenas por *pixels* de borda considerados fortes, como ilustrado na Figura 24.

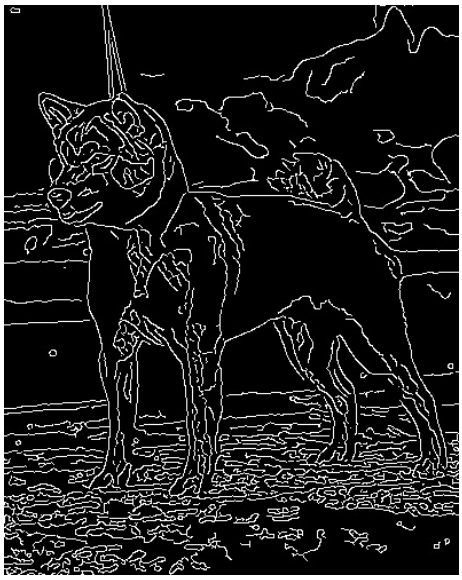
2.2.8 Vetores de características

A detecção e extração de características em uma imagem é um componente essencial para diversas aplicações de Visão Computacional (SZELISKI, 2011). Usualmente, entende-se por característica um vetor ou escalar contendo alguma informação (de forma, orientação, posição ou de identificação) de um objeto ou região de interesse presente em uma imagem. Em particular, existem diversos tipos de características que podem ser obtidas de um

Figura 23 – Imagem $\mathbf{G}_n$

Fonte: Modificado de The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

Figura 24 – Imagem resultante do algoritmo de detecção de Canny



Fonte: Modificado de The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

objeto em uma imagem. Assim, um primeiro passo é definir quais características se está interessado em extrair e quais métodos devem ser utilizados.

No sistema de localização de centro de olho desenvolvido neste trabalho, são utilizados três tipos de vetores de características para a classificação de regiões de interesse, os quais são apresentados e discutidos na sequência.

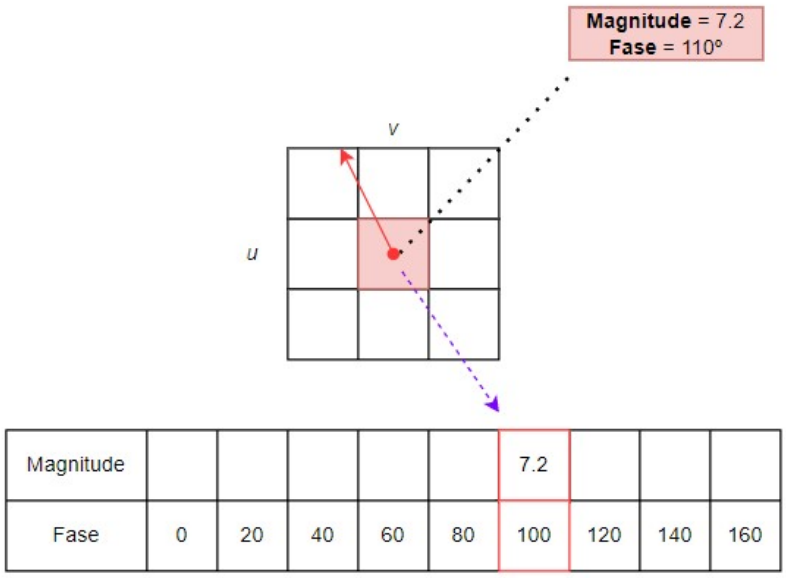
2.2.8.1 HOG - Histogram of Oriented Gradients

O HOG, ou histograma de gradientes orientados (DALAL; TRIGGS, 2005), é um algoritmo que se baseia no fato de que um objeto em uma imagem pode ser descrito pela intensidade dos gradientes ou pela direção de suas bordas (SANTOS *et al.*, 2012).

De forma simplificada, o procedimento para a extração de características HOG inicia-se pelo cálculo do gradiente da imagem de entrada $\mathbf{I}$ que se deseja extrair as informações HOG, de maneira similar à etapa inicial do algoritmo de Canny (Seção 2.2.7), utilizando as equações (19) e (20).

A imagem é então sub-dividida em janelas normalmente de tamanho 8×8, também chamadas de células, onde cada *pixel* de uma célula tem o valor da magnitude do seu gradiente armazenado em um histograma de 9 regiões, com cada região denotando um intervalo de fase do gradiente, como ilustra a Figura 25 (note nesta imagem que o gradiente do *pixel* está representado pela seta vermelha).

Figura 25 – Construção do vetor de características HOG da célula 8×8



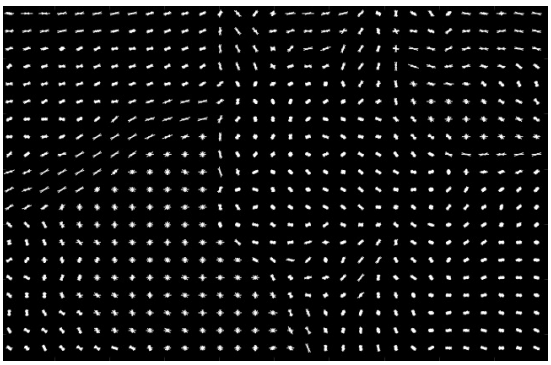
Fonte: O autor

O vetor de características HOG da imagem $\mathbf{I}$ é a concatenação de todos os vetores computados de todas as células na imagem. O tamanho deste vetor HOG depende diretamente do tamanho da imagem e também do tamanho da janela da célula, que por padrão é definido como 8×8, porém pode ser considerada 16×16 ou 32×32, dependendo da aplicação e do grau de detalhes que se deseja obter com as informações dos gradientes. Um exemplo de imagem $\mathbf{I}$, juntamente com a visualização de seus gradientes, pode ser observada na Figura 26.

Figura 26 – Visualização dos gradientes de uma imagem

(a) Imagem de entrada

(b) Gradientes resultantes das células da imagem



Fonte: Modificado de The Oxford-IIIT Pet Dataset (PARKHI *et al.*, 2012)

2.2.8.2 LBP - Local Binary Pattern

A característica LBP foi proposta em 1996 (OJALA; PIETIKÄINEN; HARWOOD, 1996), sob a premissa de descrever uma região bidimensional utilizando padrões de espaço locais e contrastes na escala de cinza.

Para rotular um *pixel* com um valor LBP, é necessário inicialmente realizar uma avaliação dos *pixels* vizinhos, considerando Vizinhança-8, em uma janela 3×3 da imagem analisada, como ilustra a Figura 27(a). Todos os *pixels* vizinhos com magnitude maior que o *pixel* de referência recebem o rótulo 1, enquanto os demais recebem rótulo 0(veja Figura 27(b)). A cada vizinho é atribuído um peso diferente, seguindo potências de base 2 (conforme ilustrado na Figura 27(c)). Por fim, os rótulos atribuídos aos *pixels* anteriormente são multiplicados por seu valor em potência de 2 (Figura 27(d)). O valor LBP do *pixel* de referência é a soma dos valores de todos os *pixels* vizinhos. Matematicamente, define-se

$$P_{LBP} = \sum_{p=0}^{J-1} s(P_r - P_v)2^p, \tag{21}$$

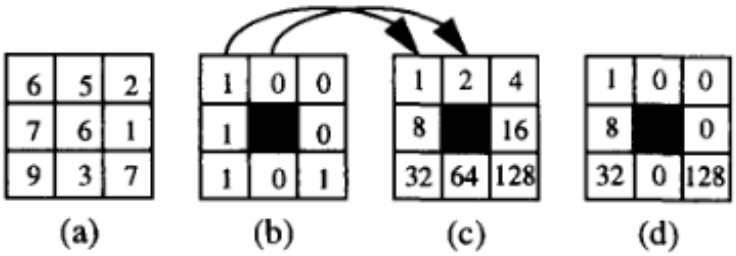
$$s(x) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

onde J é o número de *pixels* da janela analisada (por padrão, utiliza-se uma janela de tamanho 3×3, resultando em $J = 9$), P_r é o *pixel* de referência e P_v é o *pixel* vizinho analisado.

2.2.8.3 CMI - Cell Mean Intensity

Um vetor de características CMI é gerado a partir da informação da intensidade média dos *pixels* de múltiplas janelas da imagem de entrada $\mathbf{I}$ (KIM *et al.*, 2017). Para tal, inicialmente divide-se a imagem de entrada $\mathbf{I}$ em janelas de tamanho 3×3. Para

Figura 27 – Extração de características LBP de um *pixel*



Fonte: (OJALA; PIETIKÄINEN; HARWOOD, 1996)

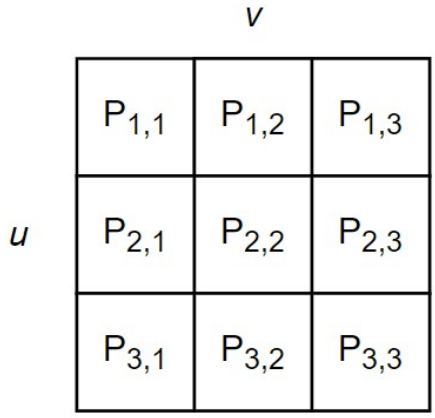
cada janela, computa-se a intensidade média da região. O vetor CMI é composto pelas intensidades médias de todas as janelas da imagem. Matematicamente, tem-se

$$V_{CMI} = [Mw_1, Mw_2, ...Mw_n], \tag{22}$$

$$Mw = \frac{\sum_{i=1}^{3} \sum_{j=1}^{3} P_{i,j}}{9}$$

onde $P_{i,j}$ são as intensidades dos *pixels* presentes em uma janela 3×3 da imagem **I**, como ilustrado na Figura 28.

Figura 28 – Janela de tamanho 3×3



Fonte: O autor

2.2.9 Algoritmo de Viola-Jones

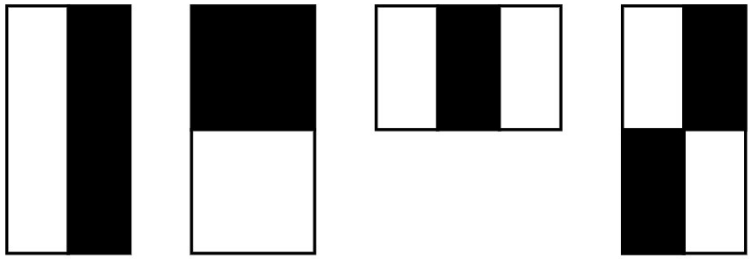
O algoritmo de detecção de objetos de Viola-Jones (VIOLA; JONES, 2001), proposto em 2001, realiza a detecção de objetos por meio de uma abordagem baseada em aprendizado de máquina, sendo muito utilizado para detecção e localização de faces devido a sua alta velocidade de processamento. Especificamente, o algoritmo de Viola-Jones faz

uso dos conceitos de características de Haar, imagens integrais e algoritmo de AdaBoost, os quais são abordados na sequência.

2.2.9.1 Características de Haar

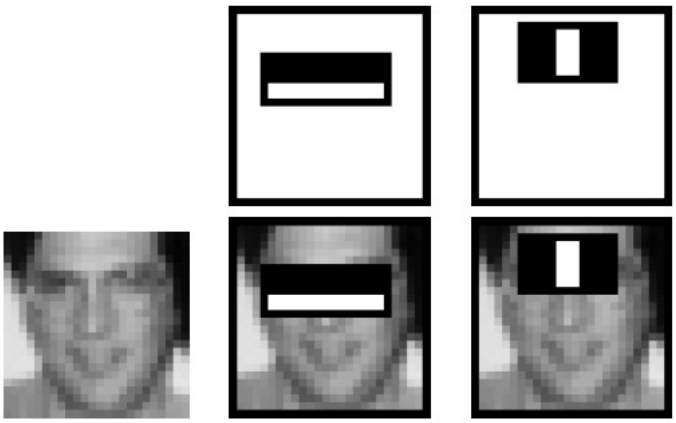
As características de Haar consistem de matrizes de *pixels* quadradas ou retangulares, com regiões brancas e pretas (veja Figura 29). Especificamente, para encontrar o valor de uma característica de Haar, aplica-se a matriz em uma região de interesse na imagem (veja Figura 30), somam-se os *pixels* localizados na região branca e na região preta e, por fim, calcula-se a diferença dessa soma. Características de Haar podem ser utilizadas para extrair informações importantes de uma imagem, como, por exemplo, as características de um rosto humano. Especificamente, uma característica de Haar pode ser utilizada para identificar as bordas na região ao redor do nariz, ou então a variação de intensidade dos *pixels* de uma sobrancelha com a pele, como ilustrado na Figura 30.

Figura 29 – Exemplos de características de Haar



Fonte: O autor

Figura 30 – Aplicação de características de Haar em um rosto



Fonte: (VIOLA; JONES, 2001)

2.2.9.2 Imagem Integral

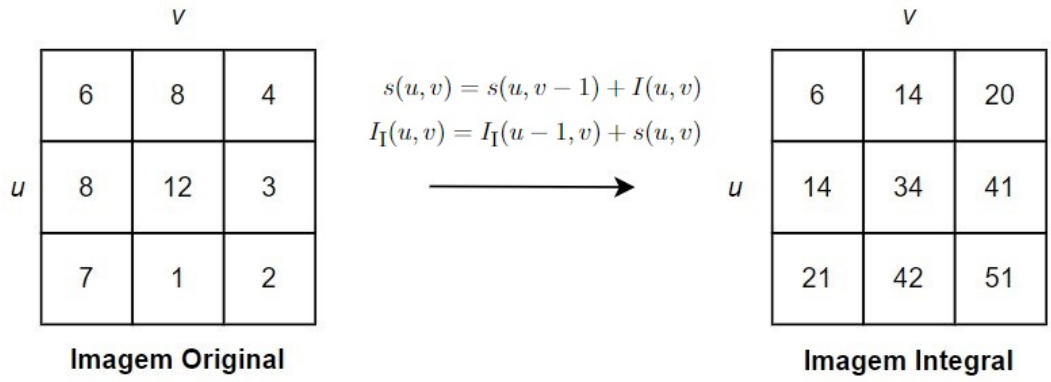
Uma imagem integral consiste de uma representação de uma imagem, onde um *pixel* na posição (u, v) corresponde ao somatório do valor de todos os *pixels* acima e à esquerda (incluso) na imagem original. Matematicamente, essa conversão é representada por

$$s(u, v) = s(u, v - 1) + I(u, v) \tag{23}$$

$$I_{\mathrm{I}}(u, v) = I_{\mathrm{I}}(u - 1, v) + s(u, v) \tag{24}$$

onde $s(u, v)$ é a soma cumulativa das linhas, $I_{\mathrm{I}}(u, v)$ é a imagem integral e $\mathbf{I}$ denota a imagem original (note que $s(u, -1) = 0$ e $I_{\mathrm{I}}(-1, v) = 0$). Um exemplo de conversão é apresentada na Figura 31.

Figura 31 – Conversão de uma imagem de entrada para imagem integral



Fonte: O autor

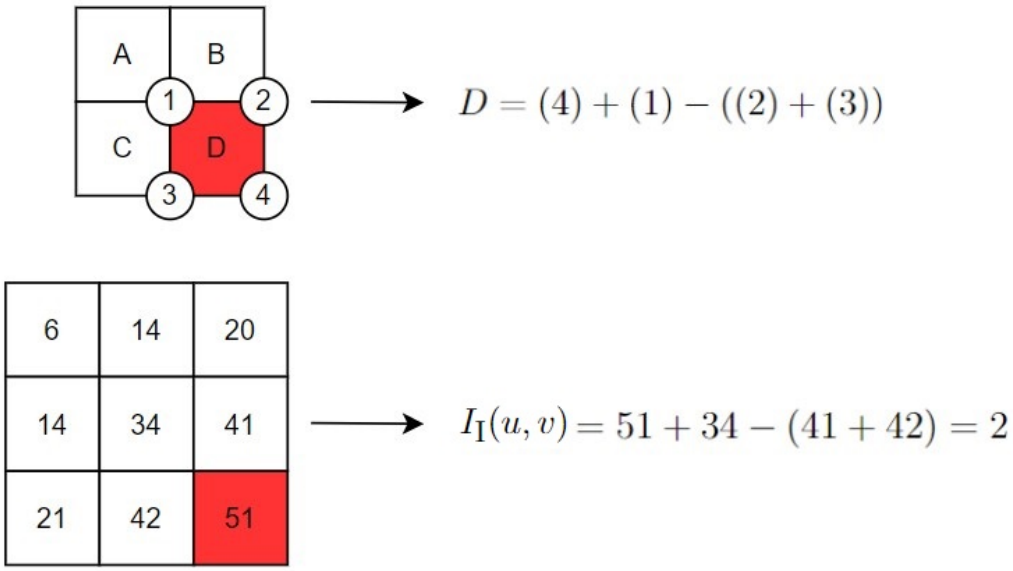
A imagem integral permite que um conjunto retangular de *pixels* seja representado como um valor inteiro. Uma coordenada (u, v) em uma imagem integral I_{I}, portanto, pode ser descrita matematicamente por

$$I_{\mathrm{I}}(u, v) = \sum_{\substack{u' \leq u \\ v' \leq v}} I(u', v'). \tag{25}$$

como ilustrado pela Figura 32.

A imagem integral é uma das razões pelo qual o algoritmo possui um tempo de processamento reduzido, viabilizando a sua execução em tempo real. Como as características de Haar são encontradas computando-se a diferença entre dois retângulos em uma imagem, a imagem integral permite que essa operação seja realizada com uma simples operação de subtração de quadrados, sem a necessidade de computar todos os *pixels* individualmente na imagem original.

Figura 32 – Identificação de um *pixel* na imagem de entrada por meio da imagem integral.



Fonte: O autor

2.2.9.3 Algoritmo AdaBoost e Classificador em Cascata

De maneira simplificada, o AdaBoost (FREUND; SCHAPIRE, 1997) é um algoritmo de aprendizado de máquina, utilizado no algoritmo de Viola-Jones com o proposito de reduzir a quantidade de falsos negativos durante a detecção de faces.

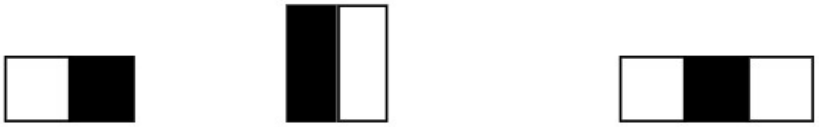
O algoritmo gera um classificador considerado forte, composto pela combinação linear de múltiplos classificadores fracos, contendo um peso para cada um deles com base no quão relevante é uma característica de Haar na identificação de um rosto. Matematicamente, pode-se expressar essa relação como

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + ... + \alpha_n f_n(x) \tag{26}$$

onde $F(x)$ é o classificador forte e α_n é o peso dado à uma determinada característica de Haar f_n. Observe pela Figura 33 uma representação visual deste equacionamento.

A etapa final para detecção de faces consiste em um método de classificação de múltiplas etapas, onde uma janela da imagem é fornecida como entrada de uma sequência de classificadores fortes treinados por meio do algoritmo AdaBoost. A cada etapa, o classificador forte possui um classificador fraco adicional. Caso, em um dado classificador forte, não seja identificada a característica de Haar correspondente a uma face na janela de entrada, essa janela é imediatamente descartada. Um diagrama ilustrando esse processo pode ser observado na Figura 34.

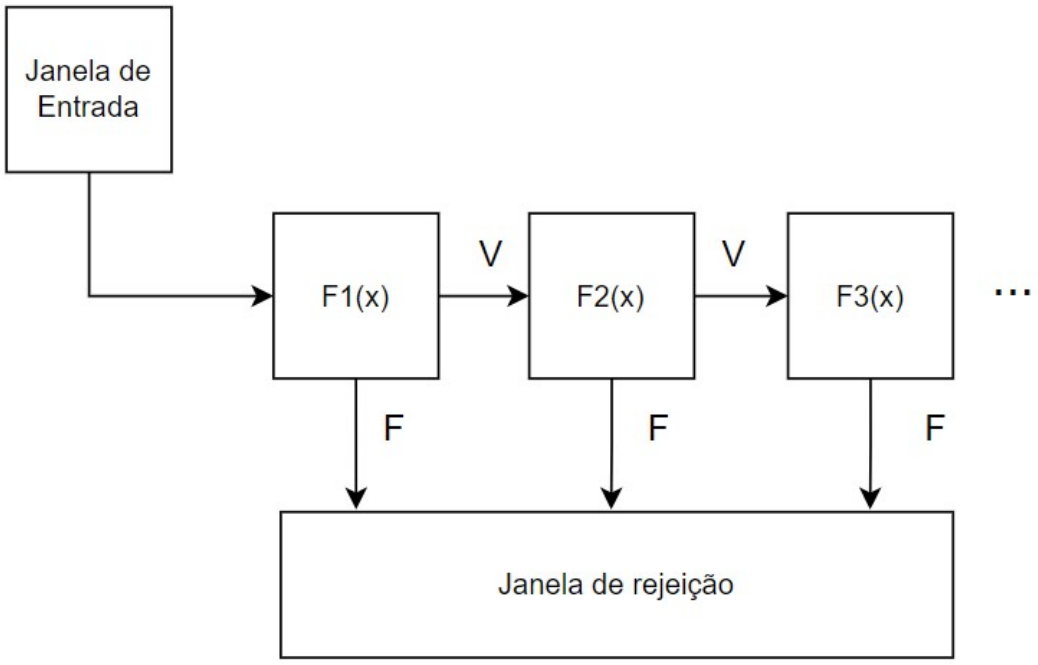
Figura 33 – Equacionamento de um classificador forte por meio de Adaboost



$$F(x) = \alpha_1 f_1(x) + \alpha_1 f_1(x) + ... + \alpha_n f_n(x)$$

Fonte: O autor, inspirado em (GUPTA, 2019)

Figura 34 – Processo de classificação em cascata de uma janela de entrada no algoritmo de Viola-Jones.



Fonte: O autor, inspirado em (VIOLA; JONES, 2001) e (LEE, 2020)

Vale ressaltar que com o passar das etapas o tempo de processamento torna-se maior, porém como muitas janelas são descartadas logo nos estágios iniciais este tempo é reduzido significantemente. A Figura 35 contém imagens de rostos identificados por meio do algoritmo de Viola-Jones.

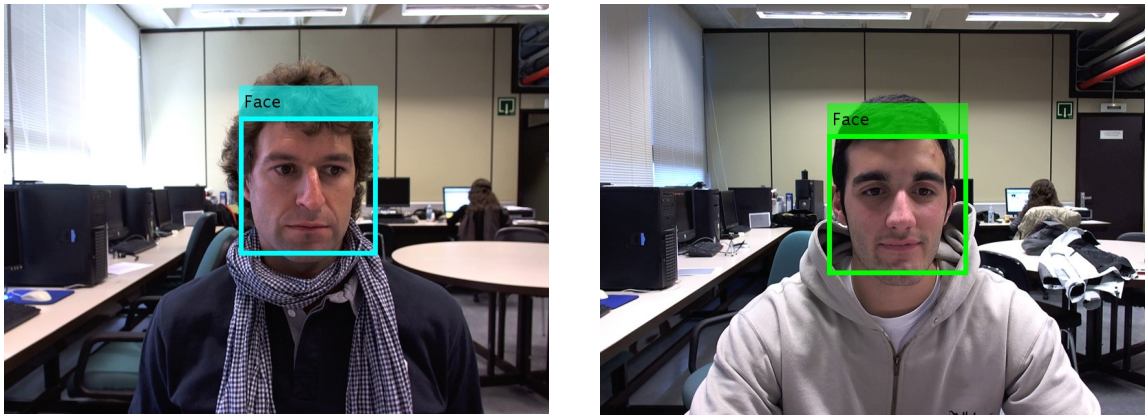
2.3 TÓPICOS INTRODUTÓRIOS DE APRENDIZADO DE MÁQUINA

O aprendizado de máquina, ou *Machine Learning*, constitui um campo de estudo cujo principal objetivo é a criação de métodos e algoritmos que permitem que computadores "aprendam"e sejam capazes, por meio de identificação de padrões a partir de dados

Figura 35 – Rostos identificados por Viola-Jones, utilizando *Matlab*

(a) Rosto destacado em ciano

(b) Rosto destacado em verde



Fonte: Adaptado de GI4E Database - Universidad Pública de Navarra

fornecidos, desenvolver soluções e respostas para problemas do mundo real (MITCHELL, 1997).

Ao avaliar a escolha de um algoritmo de aprendizado de máquina, é de grande importância definir com clareza o tipo de problema que deve ser solucionado e também o qual o método de aprendizado utilizado por este algoritmo. Nas subseções seguintes, são apresentadas as principais formas de aprendizado de máquina, os principais tipos de problemas que podem ser encontrados e uma visão geral do algoritmo utilizado (*SVM - Support Vector Machine*) para a realização deste trabalho.

2.3.1 Aprendizado Indutivo

O aprendizado por indução consiste na obtenção de conclusões genéricas a partir de um conjunto de dados fornecido (LORENA; CARVALHO, 2007). Este aprendizado pode ser subdividido em aprendizado supervisionado e não-supervisionado.

2.3.1.1 Aprendizado Supervisionado

O aprendizado supervisionado é aquele em que uma máquina indutora recebe um conjunto de exemplos rotulados, definindo o que são os conjuntos de entrada e o que são os conjuntos de saída (SOUTO *et al.*, 2003). Com base nisso, o algoritmo deve ser capaz de extrair a relação existente entre os dados de entrada e saída e ser capaz rotular corretamente as saídas para um conjunto de entradas não apresentadas previamente (LORENA; CARVALHO, 2007).

2.3.1.2 Aprendizado Não-Supervisionado

No aprendizado não-supervisionado, a máquina indutora recebe dados de entrada que não são rotulados, e deve ser capaz de agrupar esses dados seguindo uma medida de qualidade previamente estabelecida. A partir do momento em que essa máquina se adéqua às regularidades e padrões de dados de entrada, ela se torna capaz de gerar suas próprias representações internas e segregar esses dados em classes específicas (HAYKIN, 2009).

2.3.2 Problemas de Aprendizado de Máquina

2.3.2.1 Classificação e Regressão

Quando um conjunto de dados rotulados possui valores discretos e deve ser segregado em diferentes classes por intermédio de um algoritmo de aprendizado, tem-se um problema de classificação. Por outro lado, quando um conjunto de dados rotulados possui valores reais e a relação quantitativa entre esses dados deve ser interpretada por intermédio de um algoritmo de aprendizado, tem-se um problema de regressão (KUMAR; BATUT, 1970).

Para ambos os problemas, o algoritmo deve ser previamente treinado utilizando um conjunto de exemplos rotulados, que fornecem informação para que esse algoritmo seja capaz de avaliar novos dados de entrada que não foram previamente apresentados.

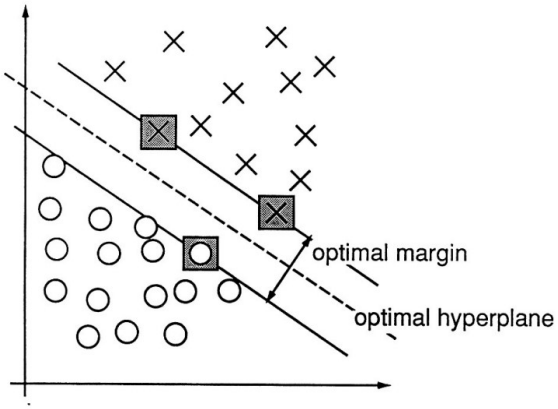
2.3.3 Técnica SVM - *Support Vector Machine*

A Máquina de Vetor de Suporte, ou SVM, consiste de um algoritmo de aprendizado supervisionado que pode ser utilizado para casos de classificação ou regressão. De maneira geral, dado um conjunto de dados de treinamento, uma SVM é capaz de construir um plano divisor (denominado hiperplano) de forma a criar uma margem nítida e divisora das diferentes classes existentes nesse conjunto de entrada (HAYKIN, 2009).

O principal objetivo de uma SVM é traçar o hiperplano de tal forma que a distância entre todas as diferentes classes presentes nos dados de treinamento seja maximizada. Observe na Figura 36 duas classes de dados separadas por um hiperplano gerado por uma SVM, em que esses dados estão distanciados desse plano por dois intervalos denominados margens ideais, que constituem as distâncias mínimas entre o hiperplano separador e os dados amostrais (LORENA; CARVALHO, 2007).

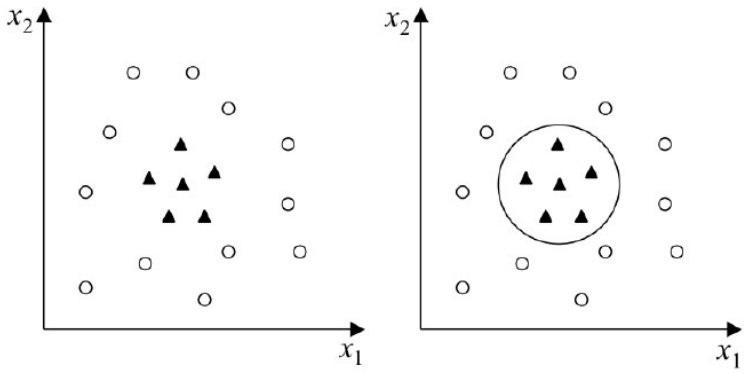
É válido ressaltar que o Hiperplano pode nem sempre ser uma linha reta (ou um plano). Hiperplanos que distinguem duas classes linearmente separáveis são gerados por SVMs do tipo lineares. Entretanto, para casos mais complexos de conjuntos de dados não-lineares, é possível utilizar uma SVM não-linear, gerando uma fronteira curva que pode dividir as diferentes classes de uma forma mais precisa (LORENA; CARVALHO, 2007), ilustrado na Figura 37.

Figura 36 – Hiperplano gerado por uma SVM linear, separando dados de treinamento em duas classes.



Fonte: (CORTES; VAPNIK, 1995)

Figura 37 – Hiperplano de uma SVM não-linear, com fronteira de divisão entre classes curva.



Fonte: (LORENA; CARVALHO, 2007)

3 DESENVOLVIMENTO

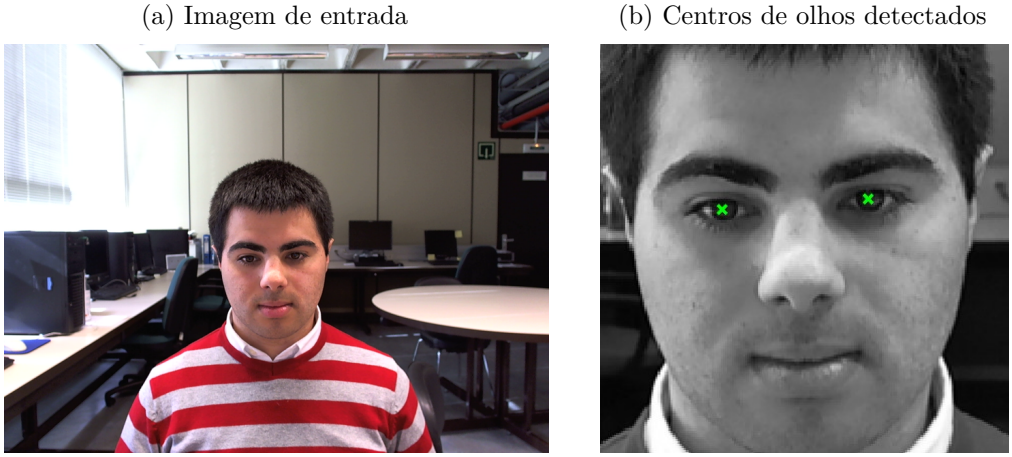
3.1 ALGORITMO IMPLEMENTADO

Este trabalho descreve o desenvolvimento de um algoritmo para localização dos centros de olhos de uma dada imagem contendo um rosto. Para implementação foi utilizado o software *MatLab*. Especificamente, o algoritmo recebe uma imagem de entrada, que pode ser colorida ou em escala de cinza, a qual deve respeitar as seguintes condições de entrada:

- A imagem deve conter apenas um rosto humano.
- O rosto presente na imagem deve estar em posição vertical, com mínima angulação ou rotação.
- O rosto deve conter um par de olhos. Imagens contendo um rosto pela metade ou com um dos olhos obstruído não são computadas corretamente.

Essas condições são necessárias pois o algoritmo não processa corretamente imagens com múltiplos rostos da forma como foi implementado e só é capaz de retornar as coordenadas de olhos se for capaz de identificar a existência de dois olhos na imagem. Após os processamentos necessários, o algoritmo retorna as coordenadas dos *pixels* dos centros de olhos no rosto detectado, como ilustra a Figura 38.

Figura 38 – Detecção dos centros de olhos de uma imagem de entrada, utilizando o algoritmo proposto.



Fonte: Modificado de GI4E Database - Universidad Pública de Navarra

O algoritmo desenvolvido neste trabalho é baseado no algoritmo proposto em (AHMED; LASKAR, 2019). Enquanto algumas etapas de processamento são idênticas as apresentadas em (AHMED; LASKAR, 2019), há outras etapas onde buscou-se uma simplificação e/ou melhoria de implementação, as quais são melhor abordadas na Seção 3.9.

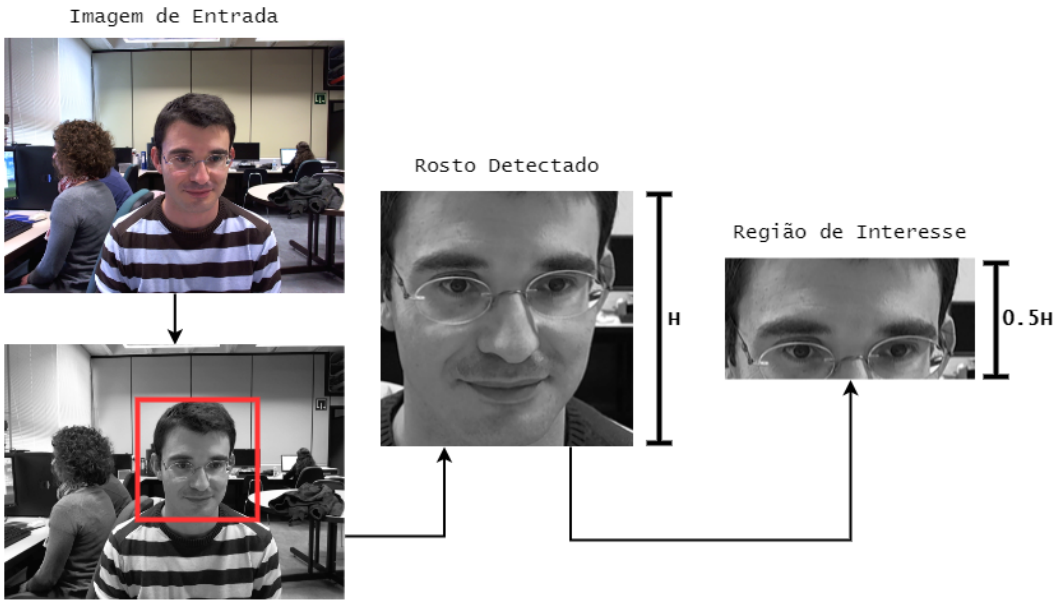
Nas próximas seções, são descritas as etapas de processamento do algoritmo implementado neste trabalho.

3.2 IDENTIFICAÇÃO DO ROSTO E PROCESSAMENTO DA REGIÃO DE INTERESSE

Como primeira etapa de processamento, a partir da imagem de entrada **I**, localiza-se o rosto que está presente utilizando o algoritmo de Viola-Jones (Seção 2.2.9). Os processamentos são realizados com a imagem em escala de cinza, portanto se a imagem de entrada for colorida é necessário convertê-la previamente.

Após a localização do rosto, seleciona-se a região de interesse por meio de uma operação de recorte (onde são realizados os processamentos seguintes), que consiste na metade superior do rosto identificado contendo ambos os olhos (veja diagrama da Figura 39).

Figura 39 – Extração da região de interesse à partir da imagem de entrada



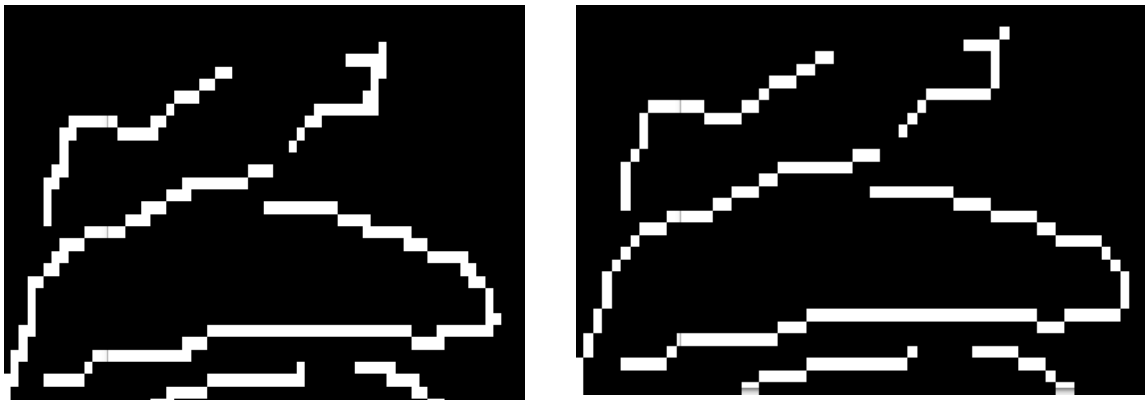
Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

Na sequência, aplica-se o algoritmo de Canny (Seção 2.2.7), é possível para gerar uma nova imagem contendo apenas os *pixels* de borda desta região de interesse. Adicionalmente, é necessário esqueletonizar essa imagem. A esqueletonização de uma imagem consiste na redução de *pixels* de uma imagem binária, preservando apenas o formato esquelético que compõe os principais traços das figuras presentes (veja Figura 41). A Figura 40 mostra um comparativo entre uma região ampliada da imagem de bordas e uma região ampliada dessa imagem após a esqueletonização.

Figura 40 – Comparativo de regiões ampliadas na imagem de bordas e na imagem após processo de esqueletonização

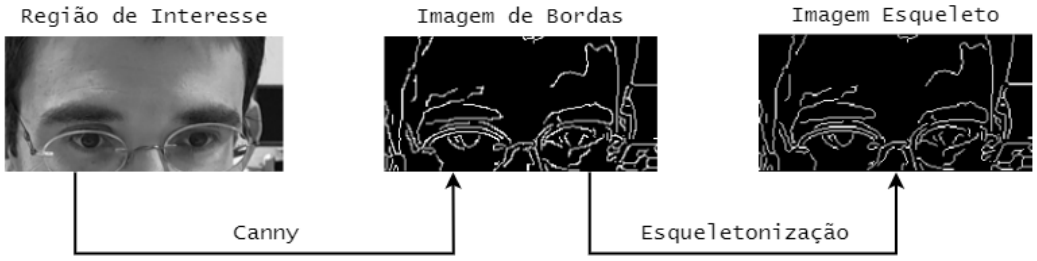
(a) Recorte da imagem de bordas

(b) Recorte da imagem de bordas após esqueletonização



Fonte: O Autor

Figura 41 – Processo de esqueletonização da região de interesse



Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

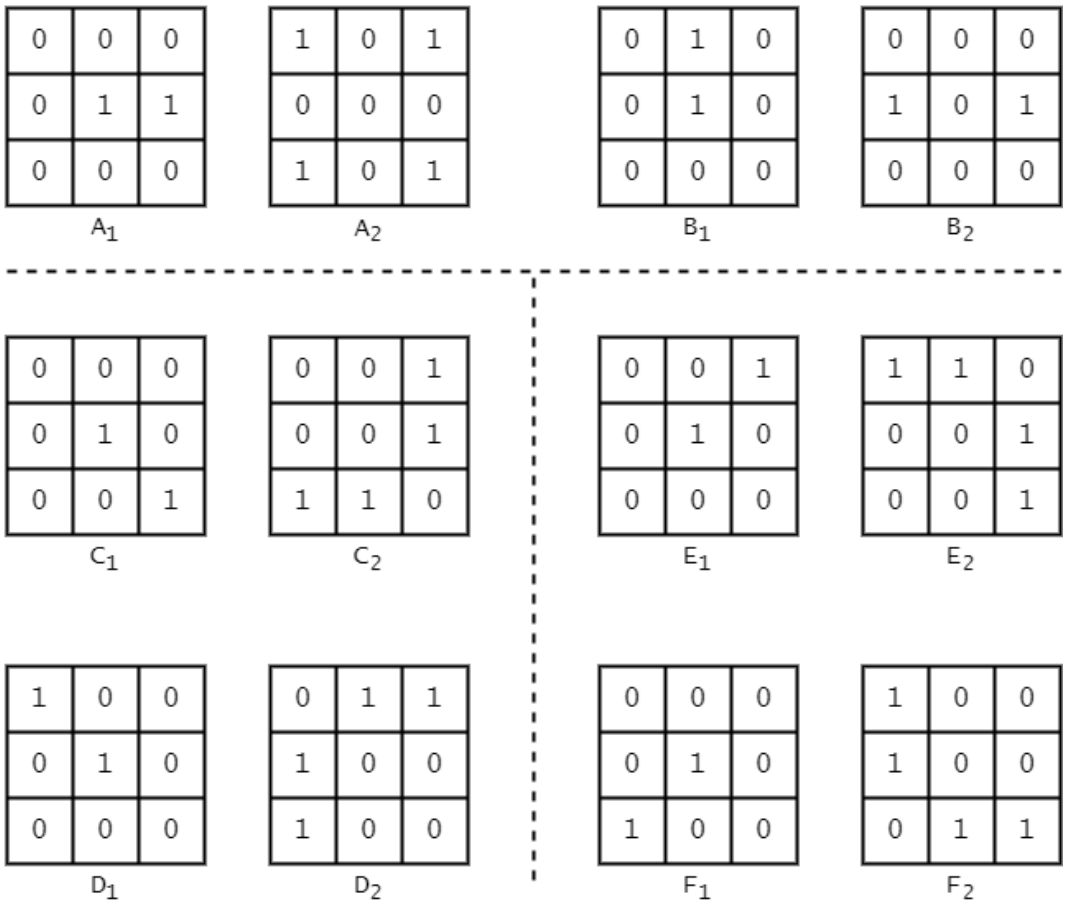
3.3 EXTRAÇÃO DAS BORDAS COM INCLINAÇÕES POSITIVAS E NEGATIVAS

Nesta etapa, são identificados todos os segmentos de borda que possuem inclinação positiva (PSE - *Positive Slope Edges*) e negativa (NSE - *Negative Slope Edges*) na imagem esqueleto. Especificamente, duas imagens distintas são geradas após essa segregação, uma contendo apenas as características PSE e outra contendo apenas as características NSE.

Para isso, utiliza-se a Transformada Hit-or-Miss com seis pares de elementos estruturantes, com cada par possuindo a finalidade de identificar a presença de uma característica específica nos segmentos de borda. Especificamente, cada par de elementos estruturantes consiste de um elemento de primeiro plano, responsável por localizar a forma do segmento de borda desejado, e um elemento de segundo plano, responsável por identificar os *pixels* de contorno externo da borda desejada.

A Figura 42 ilustra os pares de elementos estruturantes utilizados, onde (A_1, A_2),

Figura 42 – Pares de elementos estruturantes utilizados para identificação de inclinações das características de borda



Fonte: O autor

(B_1, B_2), (C_1, C_2), (D_1, D_2), (E_1, E_2) e (F_1, F_2) constituem pares distintos, em que a primeira matriz corresponde ao elemento de primeiro plano e a segunda matriz corresponde ao elemento de segundo plano.

Utilizando os pares de elementos estruturantes, pode-se processar a imagem de bordas por meio de três etapas:

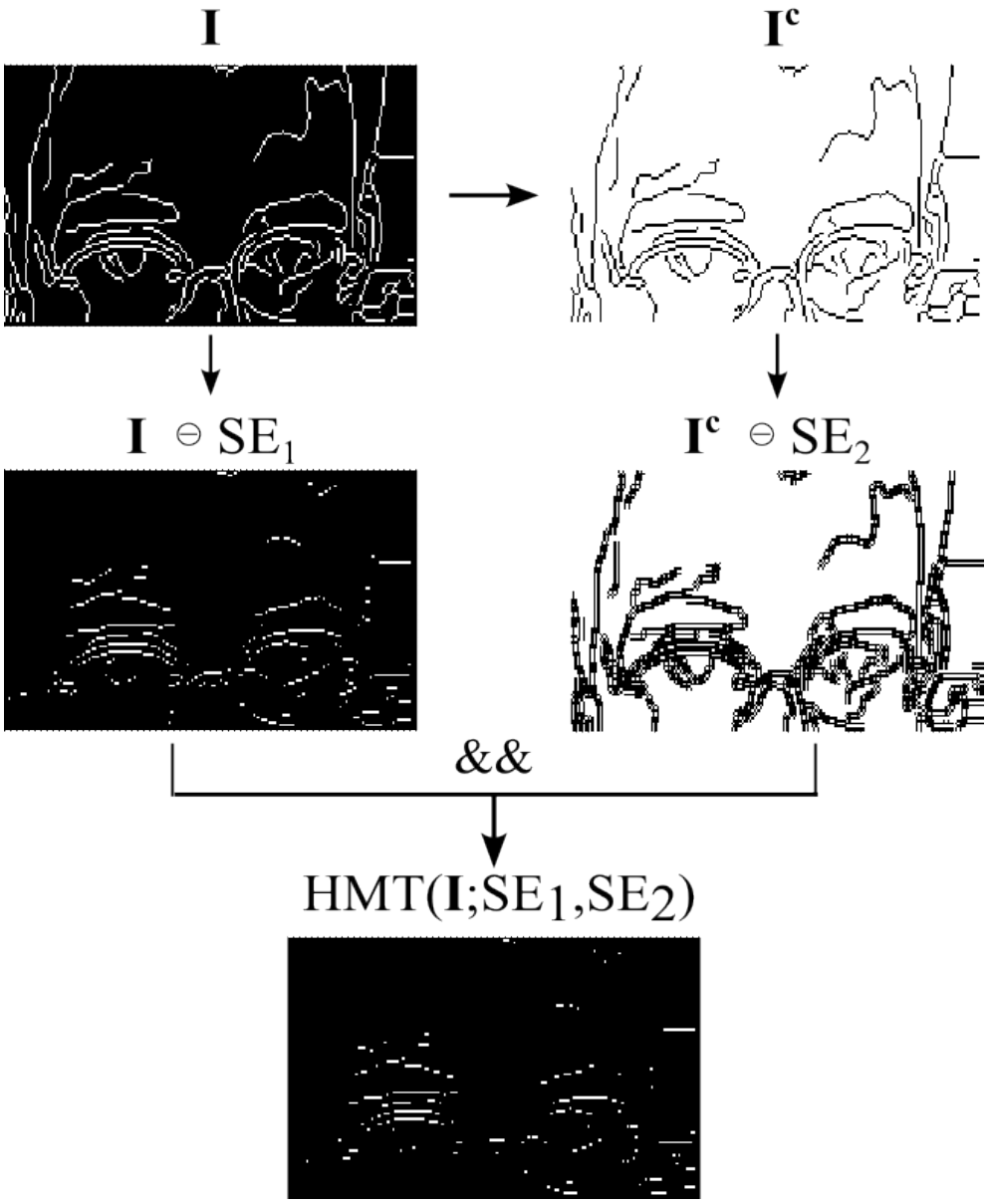
- Aplicar operação de erosão na imagem utilizando o elemento de primeiro plano;
- Aplicar operação de erosão no complemento da imagem (isto é, sua imagem negativa) utilizando o elemento de segundo plano;
- Realizar a interseção das regiões em branco das imagens binárias obtidas nas etapas anteriores.

Matematicamente, esse conjunto de operações pode ser definido por

$$\begin{aligned} HMT(\mathbf{I}; SE_1, SE_2) &= (\mathbf{I} \ominus SE_1) \cap (\mathbf{I}^c \ominus SE_2) \\ &= HMT(\mathbf{I}; SE) \end{aligned} \tag{27}$$

onde SE_1 e SE_2 são os elementos de primeiro plano e segundo plano, respectivamente, e $\mathbf{I}^c$ é a imagem complemento de $\mathbf{I}$. A Figura 43 ilustra esses procedimentos.

Figura 43 – Diagrama ilustrativo do procedimento de identificação de inclinações por HMT

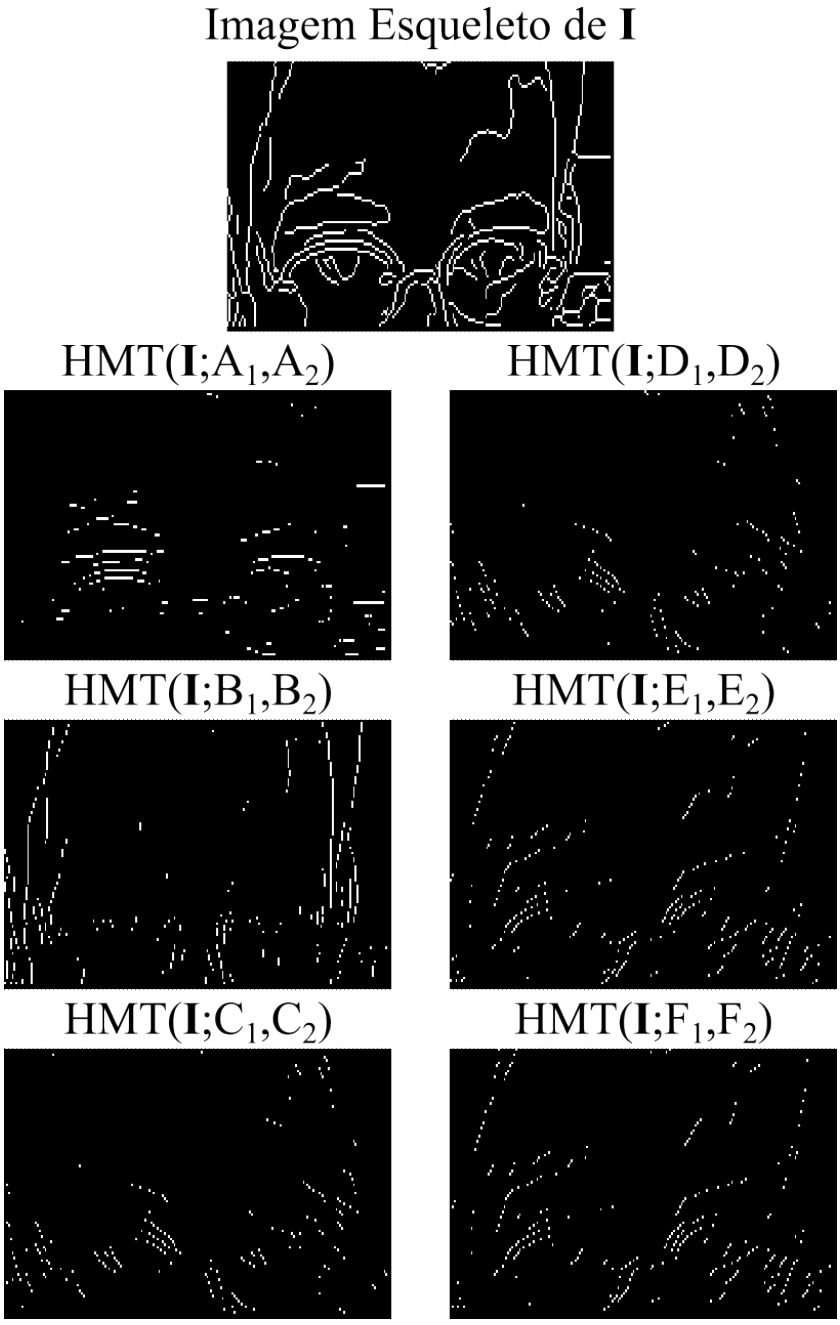


Fonte: O autor

Dependendo dos pares de elementos estruturantes utilizados, a operação de HMT retorna características de inclinação diferentes. Por exemplo, as operações $HMT(\mathbf{I}; A_1, A_2)$ e $HMT(\mathbf{I}; B_1, B_2)$ retornam bordas com inclinações horizontais e verticais, respectivamente. As operações $HMT(\mathbf{I}; C_1, C_2)$ e $HMT(\mathbf{I}; D_1, D_2)$ retornam bordas que possuem inclinação diagonal positiva enquanto $HMT(\mathbf{I}; E_1, E_2)$ e $HMT(\mathbf{I}; F_1, F_2)$ retornam bordas que possuem inclinação diagonal negativa. A Figura 44 ilustra a resposta HMT de

todos os pares de elementos estruturantes apresentados na Figura 42.

Figura 44 – Resposta HMT correspondente a todos os pares de elementos estruturantes.



Fonte: O autor

As imagens contendo as características *PSE* e *NSE* (ilustradas na Figura 45) consistem de uma combinação das respostas HMT da imagem **I** previamente encontradas e podem ser matematicamente representadas como:

$$PSE(\mathbf{I}) = HMT(\mathbf{I}; A) + HMT(\mathbf{I}; B) + HMT(\mathbf{I}; C) + HMT(\mathbf{I}; D) \quad (28)$$

$$NSE(\mathbf{I}) = HMT(\mathbf{I}; A) + HMT(\mathbf{I}; B) + HMT(\mathbf{I}; E) + HMT(\mathbf{I}; F) \quad (29)$$

Figura 45 – Imagens contendo características de borda com inclinação positivas e negativas

(a) Imagem PSE

(b) Imagem NSE

Fonte: O autor

3.4 SELEÇÃO DE BORDAS COM CARACTERÍSTICAS SEMI-ELÍPTICAS E SEMI-CIRCULARES

Até o momento, nenhum procedimento realizado nos processamentos anteriores levou em consideração as características e proporções físicas inerentes ao rosto humano. As imagens PSE e NSE obtidas identificam todas as bordas com inclinações positivas e negativas, porém é necessário avaliar quais informações presentes nestas imagens são relevantes para a identificação de um olho.

Há duas características principais dos olhos humanos que são exploradas no algoritmo: a borda semi-elíptica das pálpebras (característica sEES - *semi-Eliptical Edge Slope*) e a borda semi-circular da íris (característica sCES - *semi-Circular Edge Slope*), como ilustrado na Figura 46.

Em particular, o sistema deve ser capaz de identificar bordas nas imagens PSE e NSE relacionadas com as características sEES e sCES. Especificamente, realizam-se todas as combinações possíveis entre as bordas da imagem PSE com bordas da imagem NSE, verificando se o par gerado forma uma característica sEES ou sCES. Porém, antes disso, são eliminadas algumas bordas nas imagens PSE e NSE visando eliminar ruídos e reduzir a complexidade computacional das etapas posteriores.

Cada conjunto de *pixels* que costitui um componente conectado (considerando conectividade $\mathcal{N}^8$) pode ser rotulado como uma característica de borda única. A largura

Figura 46 – Rostos com características sEES e sCES realçadas em vermelho e verde, respectivamente

(a) (b)

Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

e altura de um olho humano correspondem à aproximadamente 1/5 e 1/6 da altura da face humana aproximadamente (FARKAS *et al.*, 2005). Com base nessas informações, é possível traçar alguns limiares com relação ao tamanho esperado das bordas nas imagens PSE e NSE com potencial de estarem relacionadas com as características sEES e sCES.

Pode-se considerar um ruído nessa imagem toda e qualquer borda que seja totalmente vertical ou horizontal. Outra condição que impede uma borda de estar relacionada com um olho é seu tamanho: se uma borda possuir menos de dois *pixels* ou se for maior do que 1/3 do tamanho relativo à altura ou à largura do rosto, ela deve ser descartada (a face extraída por Viola-Jones está contida em uma imagem quadrada, ou seja, suas medidas de largura e altura são iguais). Aplicando as condições mencionadas nas imagens PSE e NSE, obtém-se as suas versões filtradas que podem ser observadas na Figura 47.

Figura 47 – Imagens PSE e NSE após extração de características indesejadas

(a) Imagem PSE filtrada (b) Imagem NSE filtrada

Fonte: O autor

As características de bordas restantes nessas imagens sofrem mais processamentos, visando verificar quais os pares de bordas nas imagens PSE e NSE formam características

sEES e sCES. Nesta etapa, selecionam-se pares de PSE e NSE que formam as características sCES e sEES. Para tal são utilizados certos critérios dimensionais e de distância relativa, descritos a seguir:

$$sCES_{(u,v)} = \begin{cases} 1 & \begin{cases} 1 < (PSE_h, PSE_l, NSE_h, NSE_l, Dist_{(pse,nse)}) \leq 0.1 \times Face_l \\ Avg_{intensidade} < Face_{intensidade} \end{cases} \\ 0 & \text{caso contrário} \end{cases} \tag{30}$$

$$sEES_{(u,v)} = \begin{cases} 1 & \begin{cases} 1 < (PSE_h, NSE_h) \leq 0.1 \times Face_l \\ 1 < (PSE_l, NSE_l) \leq 0.25 \times Face_l \\ 0.1 \times Face_l < Dist_{(pse,nse)} \leq 0.25 \times Face_l \end{cases} \\ 0 & \text{caso contrário} \end{cases} \tag{31}$$

onde PSE_h e NSE_h correspondem às alturas em *pixels* das bordas analisadas e PSE_l e NSE_l correspondem às larguras em *pixels*. O termo $Dist_{(pse,nse)}$ denota a distância em *pixels* entre os centroides das analisadas. Por fim, $Avg_{intensidade}$ caracteriza a intensidade média dos *pixels* presentes na face na região local de onde as características PSE e NSE foram extraídas, enquanto $Face_{intensidade}$ representa a intensidade média dos *pixels* da região de interesse (metade superior do rosto).

O cálculo dos centroides, que correspondem às coordenadas do *pixel* central de cada uma das bordas, (veja Figura 48), pode ser realizado através das equações:

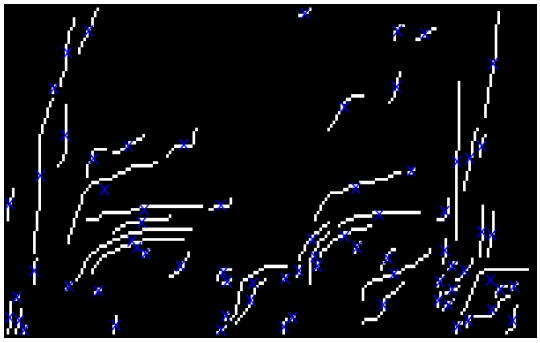
$$C(u,v) = [C_u, C_v] \tag{32}$$

$$C_u = \frac{\sum\limits_{(u,v)\in\mathbf{I}} \mathbf{I}(u,v)}{\sum\limits_{(u,v)\in\mathbf{I}} u\mathbf{I}(u,v)} \tag{33}$$

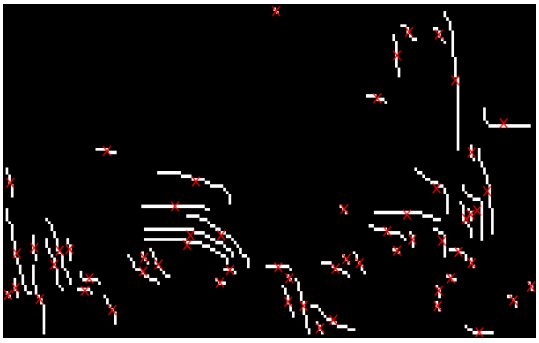
$$C_v = \frac{\sum\limits_{(u,v)\in\mathbf{I}} \mathbf{I}(u,v)}{\sum\limits_{(u,v)\in\mathbf{I}} v\mathbf{I}(u,v)} \tag{34}$$

Figura 48 – Imagens PSE e NSE filtradas e com centroides das características demarcados

(a) Imagem PSE filtrada com centroides



(b) Imagem NSE filtrada com centroides



Fonte: O autor

Note que, de maneira geral, as características de borda devem estar dentro da conformidade com respeito ao seu tamanho em relação ao tamanho do rosto. Uma borda muito longa, por exemplo, não poderia compor o contorno de uma íris ou uma curvatura de pálpebra e portanto já pode ser descartada. O limiar de intensidade durante a seleção de *sCES* é utilizado para averiguar se a região do rosto em que uma PSE ou NSE se encontra corresponde à um olho humano. Como um olho é uma região escura, a sua intensidade de *pixels* local ($Avg_{intensidade}$) deve ser menor do que a intensidade de *pixels* média do rosto ($Face_{intensidade}$), e se isso não for verdade para alguma das características de borda, ela pode ser descartada. Já o limiar de distância ($Dist_{(pse,nse)}$) entre PSE e NSE é utilizado para descartar características que estejam muito afastadas entre si. Caso a distância entre elas seja muito acentuada, elas não podem ser candidatas à formar uma *sCES* ou uma *sEES*.

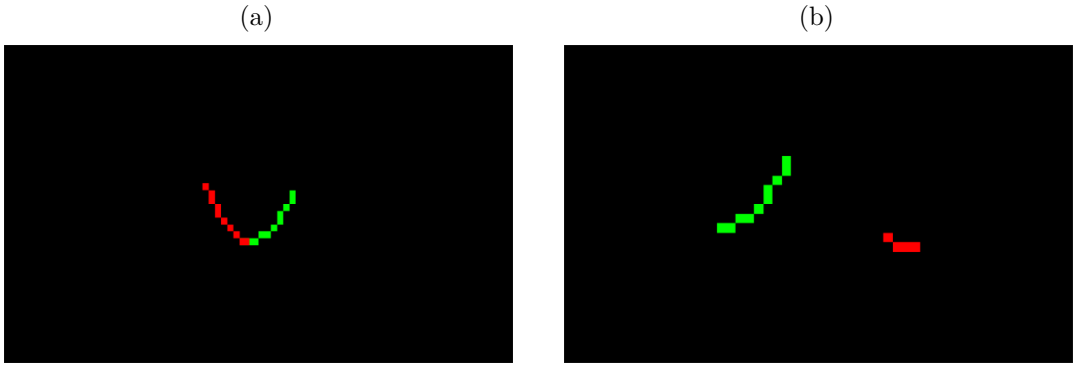
Com esses limites definidos, todas as características PSE são comparadas com todas as características NSE. Caso essa comparação resulte em um valor positivo, ambas as bordas são armazenadas em uma célula, referente às utilizadas para formação de uma característica *sCES* ou *sEES*.

3.5 AVALIAÇÃO DE CURVATURA E ESTIMATIVA DE CENTRO DAS *SCES* E *SEES*

As características de borda separadas na etapa anterior são selecionadas/filtradas uma última vez, agora para avaliar o formato côncavo ou convexo da PSE em relação ao formato da NSE. Observe na Figura 49 exemplos de duplas formando possíveis características *sCES* e na Figura 50 exemplos de duplas formando possíveis características *sEES*.

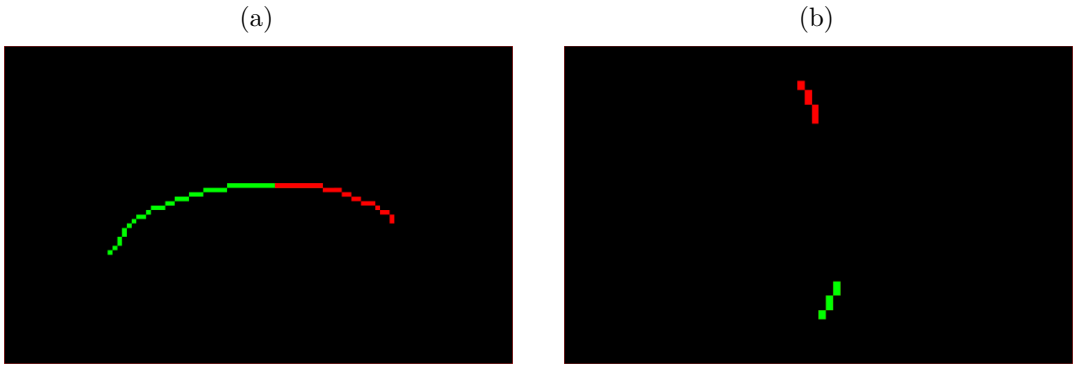
Observe que bordas que formam uma característica *sCES* ou *sEES* semelhante

Figura 49 – Característica PSE (verde) e NSE (vermelho) formando duplas candidatas à *sCES*



Fonte: O autor

Figura 50 – Característica PSE (verde) e NSE (vermelho) formando duplas candidatas à *sEES*



Fonte: O autor

àquelas encontradas em um olho humano possuem a mesma concavidade. Dessa forma, deve-se analisar individualmente cada dupla de candidatos PSE e NSE e identificar suas respectivas concavidades, seguindo as condições da Tabela 1. Na avaliação realizada, também é considerada a concavidade definida como neutra, que corresponde a uma reta (com inclinação positiva ou negativa).

Note portanto, de acordo com as informações apresentadas na Tabela 1, os pares de borda ilustradas nas Figuras 49(a), 50(a) e 49(b) são duplas aprovadas para formação de uma característica, enquanto o par de bordas da Figura 50(b) forma um par rejeitado.

Todas as duplas aprovadas nesta etapa de processamento correspondem à candidatos à olhos, seja por meio de características *sCES* ou *sEES*. Logo, é necessário extrair o centro relativo à PSE e NSE presentes, de maneira a estimar a coordenada do centro de olho que essas características carregam. Essa estimativa é dada pela média aritmética das

Tabela 1 – Tabela condicional para aprovação ou rejeição da dupla de características de acordo com suas concavidades

PSE	**NSE**	**Aprovação / Rejeição**
Concavidade Positiva	Concavidade Positiva	Aprovado
Concavidade Positiva	Concavidade Neutra	Aprovado
Concavidade Positiva	Concavidade Negativa	Rejeitado
Concavidade Negativa	Concavidade Neutra	Aprovado
Concavidade Negativa	Concavidade Negativa	Aprovado
Concavidade Negativa	Concavidade Positiva	Rejeitado
Concavidade Neutra	Concavidade Negativa	Aprovado
Concavidade Neutra	Concavidade Positiva	Aprovado
Concavidade Neutra	Concavidade Neutra	Rejeitado

Fonte: O autor

coordenadas dos centroides de ambas as características. Matematicamente, tem-se

$$C_{olho} = [C_{(u,olho)}, C_{(v,olho)}] \tag{35}$$

$$C_{(u,olho)} = \frac{C_{(u,pse)} + C_{(u,nse)}}{2} \tag{36}$$

$$C_{(v,olho)} = \frac{C_{(v,pse)} + C_{(v,nse)}}{2} \tag{37}$$

onde C_{olho} denota o centro relativo à PSE e NSE, $C_{(u,olho)}$ e $C_{(v,olho)}$ são as coordenadas horizontais e verticais do centro relativo, respectivamente, e $C_{(u,pse)}$, $C_{(v,pse)}$, $C_{(u,nse)}$ e $C_{(v,nse)}$ são as coordenadas horizontais e verticais dos centroides das bordas PSE e NSE, calculados por meio das equações (33) e (34).

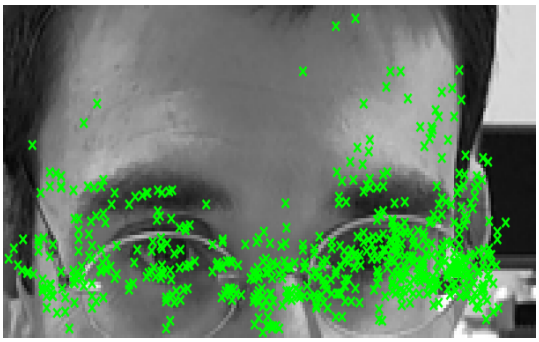
Após a realização do procedimento descrito, tem-se agora um vetor contendo todas as estimativas candidatas de centros de olhos. A Figura 51 mostra todos os possíveis centros de olhos encontrados referente às *sCES* e *sEES* na região de interesse da Figura 39.

Mesmo com todos os processamentos realizados, nota-se ainda uma grande quantidade de falsos positivos na identificação de centros de olhos. Nas etapas seguintes, esse número é reduzido com o auxílio de técnicas de aprendizado de máquina.

Figura 51 – Região de interesse com estimativa de centro baseadas nas características *sCES* e *sEES* identificadas

(a) Estimativa dos centros das características *sCES*

(b) Estimativa dos centros das características *sEES*



Fonte: O autor

3.6 TREINAMENTO DE UMA MÁQUINA DE VETOR DE SUPORTE (*SVM*) PARA IDENTIFICAÇÃO DE OLHOS

Com todos os candidatos a centros de olhos obtidos, é necessário criar um classificador capaz de distinguir uma imagem que contenha um olho de uma imagem que não contenha um olho. Para tal, uma *SVM* foi treinada utilizando imagens rotuladas, separadas em duas classes:

- Um conjunto de imagens rotuladas contendo olhos humanos.
- Um conjunto de imagens rotuladas contendo regiões da face que não contenham olhos humanos.

Todas as imagens utilizadas no treinamento do classificador foram geradas a partir do *dataset GI4E Database*, que contém um total de 1236 imagens de rostos humanos. Nesse *dataset*, as coordenadas dos centros olhos são fornecidas, o que permite a geração de imagens contendo somente olhos.

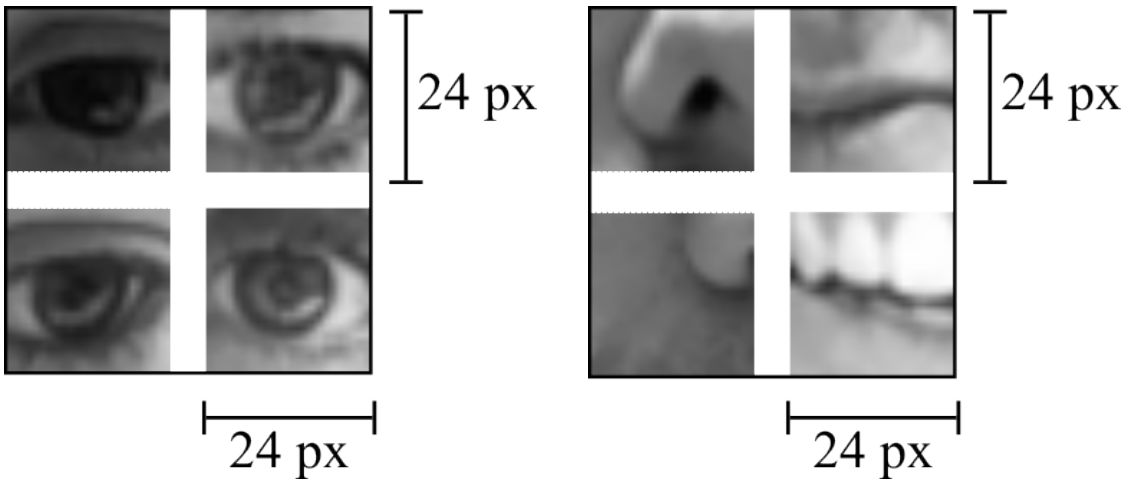
Para fins de padronização, as imagens dos conjuntos de treinamento foram definidas todas do mesmo tamanho: 24 *pixels* de altura por 24 *pixels* de largura e em escala de cinza. Para cada imagem, gerou-se um vetor de características, consistindo na concatenação de três outros vetores de características, a saber: os vetores *HOG*, *LBP* e *CMI* (veja Seção 2.2.8). Além disso, cada vetor ainda possui como último dígito o rótulo 0, caso a imagem não seja um olho, ou o rótulo 1, caso a imagem seja um olho. Este vetor pode ser matematicamente descrito por:

$$V_{caracteristicas} = [V_{HOG}, V_{LBP}, V_{CMI}, R] \tag{38}$$

onde R corresponde ao rótulo 0 ou 1. A Figura 52 ilustra exemplos de imagens de olhos e de não-olhos.

Figura 52 – Exemplos de imagens de olhos e não-olhos geradas a partir do *GI4E Database*

(a) Conjunto de imagens de olhos (b) Conjunto de imagens de não-olhos



Fonte: O autor

Para a criação do conjunto de treinamento rotulado como "olhos", inicialmente considerou-se todos os rostos presentes no *Dataset*. Os 1236 rostos contém um total de 2472 imagens de olhos. Como as coordenadas dos centros dos olhos são conhecidas, gerou-se uma janela de 24×24 *pixels* ao redor destes centros. Das 2472 possíveis imagens de olhos, foram utilizadas 1500 para a geração de 1500 vetores de características distintos, com rótulo $R = 1$. Os olhos foram escolhidos aleatoriamente, incluindo olhos esquerdos ou direitos de qualquer uma das imagens. É importante ressaltar que esse conjunto de vetores foi obtido apenas de imagens de olhos abertos, não contemplando olhos fechados, semi-cerrados ou parcialmente obstruídos.

Para a geração do conjunto de treinamento de "não-olhos", também foram utilizadas imagens dos rostos do *Dataset*. Para cada imagem de rosto, foram definidos oito pontos aleatórios na face, fora da região dos olhos: seis deles na face inferior aos olhos, contemplando as características faciais como boca, nariz e bochechas, e dois pontos na região da testa. Para cada um desses pontos, uma janela de 24×24 *pixels* foi gerada ao redor do local, de onde os vetores de características com rótulo $R = 0$ são extraídos. No total, foram obtidos 9888 diferentes vetores de características, dos quais 1500 foram posteriormente fornecidos para o treinamento da *SVM*. Veja na Figura 53 um exemplo de imagem de rosto com pontos demarcados para a extração de imagens de não-olhos.

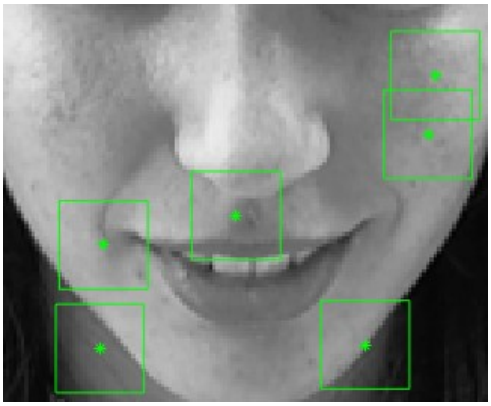
Após a extração de todas as janelas de imagens contendo olhos ou não-olhos e seus respectivos vetores de características, treinou-se um classificador *SVM*, utilizando a aplicação *Classification Learner*, disponível no *Matlab*.

Cada um dos vetores de características possui 268 posições, onde a última posição é o rótulo classificador R. Para o treinamento do classificador SVM na aplicação *Classifi-*

Figura 53 – Exemplo de rosto com extração de imagens de não-olhos aleatórias

(a) Imagem original

(b) Regiões aleatórias na face para criação de imagens de não-olho



(c) Regiões aleatórias na testa para criação de imagens de não-olho

Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

cation Learner, gerou-se uma tabela contendo todos os 3000 vetores selecionados na etapa anterior. Especificamente, foi treinado um classificador SVM para cada função de kernel, sendo selecinado aquele que retornou os melhores resultados de acurácia (no caso,SVM com função de kernel não-linear cúbica). Observe na Figura 54(a) os resultados obtidos após o treinamento dos classificadores SVM e na Figura 54(b), a matriz de confusão com os resultados obtidos no treinamento. Note que, dos 3000 vetores fornecidos, houve apenas 1 resultado falso negativo.

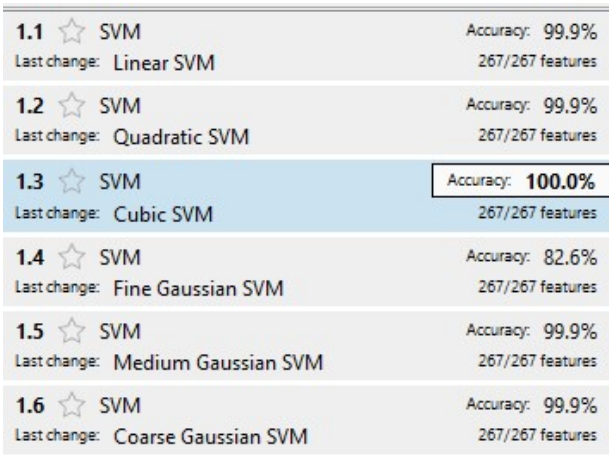
3.7 AVALIAÇÃO DOS CENTROS DE OLHOS ESTIMADOS UTILIZANDO O CLASSIFICADOR TREINADO

Com o classificador treinado, é possível utilizá-lo para distinguir quais candidatos a centros de olhos encontrados na Seção 3.5 representam de fato regiões com olhos e quais são falsos positivos.

De maneira similar à criação do conjunto de treinamento, para cada um dos candidatos é extraída uma janela de 24×24 *pixels*. Então, o vetor de características $V_{caracteristicas}$ dessa região é extraído, e essas informações são avaliadas pelo classificador que irá apontar a presença ou não de um olho nesta janela.

Figura 54 – Informações extraídas do aplicativo *Classification Learner*

(a) Acurácia das *SVM* testadas



(b) Matriz de confusão da *SVM* não-linear cúbica escolhida

Fonte: *Matlab - Classification Learner App*

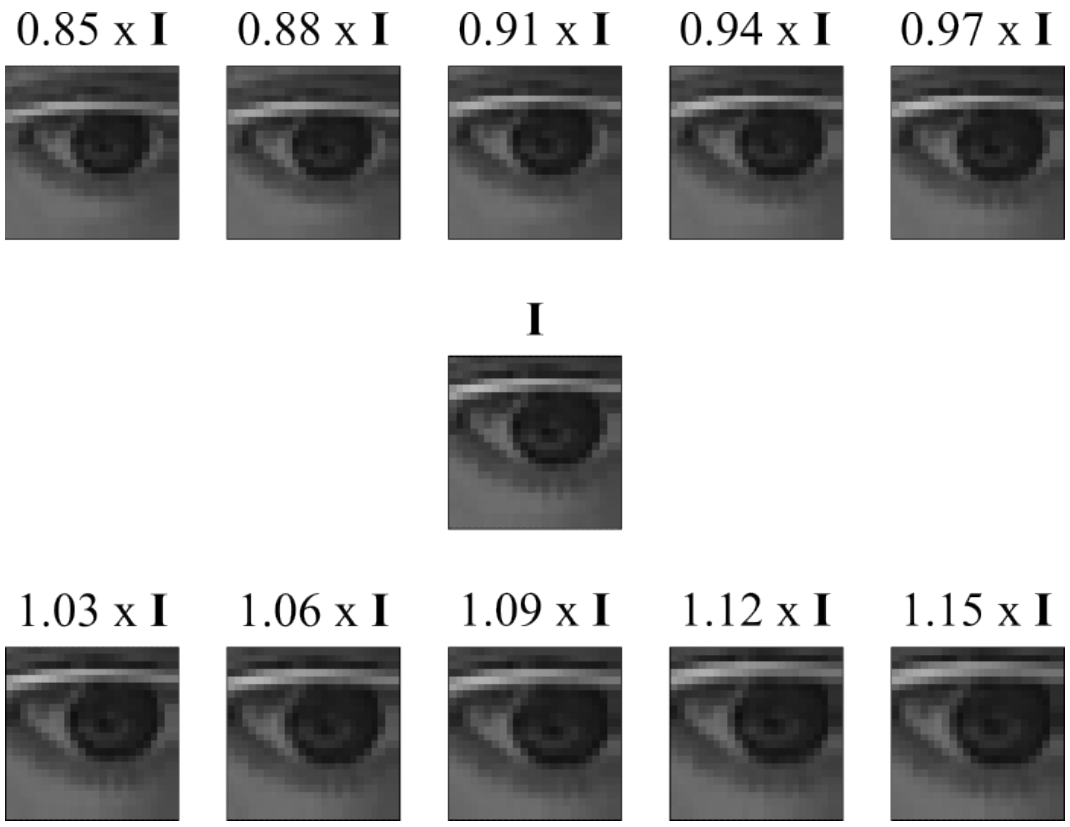
Entretanto, apenas essa avaliação pode não ser suficiente para classificar a região analisada corretamente. As imagens contendo rostos são suscetíveis a muitas variações de escala, que podem ser oriundas do formato do rosto, da proximidade com a câmera que capturou a foto ou até mesmo da resolução da câmera com a qual a imagem de rosto foi obtida. Para contornar esse problema, a imagem ao redor do candidato a centro de olho é redimensionada em diferentes escalas, a fim de fornecer ao classificador múltiplas janelas de uma mesma região com escalas de tamanho levemente distintas. Especificamente, a

região analisada é tanto aumentada quanto diminuída, seguindo um intervalo de escala de 0.85 a 1.15 com relação às proporções da imagem na janela original (veja Figura 55).

Ao final, onze diferentes vetores de características são processados pelo classificador:

- O vetor correspondente à janela original;
- Os vetores correspondentes à cinco janelas com uma imagem maior;
- Os vetores correspondentes à cinco janelas com uma imagem menor.

Figura 55 – Janelas de um candidato à olho com diferentes proporções de tamanho



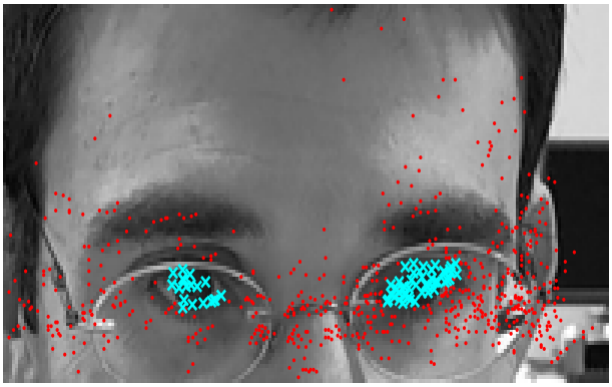
Fonte: O autor

Caso qualquer uma das onze janelas seja rotulada pelo classificador treinado como sendo um olho, as coordenadas do candidato a centro de olho analisado são armazenadas e este centro recebe uma nota, que pode variar de 1 a 11 de acordo com o número de janelas classificadas como olho. Por outro lado, caso o centro de olho analisado tenha todas as onze janelas classificadas como não-olho, este candidato a centro de olho é eliminado.

Na sequência, avaliam-se todos os candidatos a centros de olhos identificados pelo classificador SVM, tanto aqueles encontrados por meio de características *sCES* como de características *sEES*. A Figura 56 mostra os centros identificados pelo classificador como olho, assim como os centros rejeitados.

É necessário, entretanto, verificar quais duplas de candidatos a centros formam um par de olhos. Para isto, utilizam-se dois parâmetros: a distância entre os centros, que

Figura 56 – Centros identificados pelo classificador como regiões de olho (marcadores em azul) e centros rejeitados (marcadores em vermelho)



Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

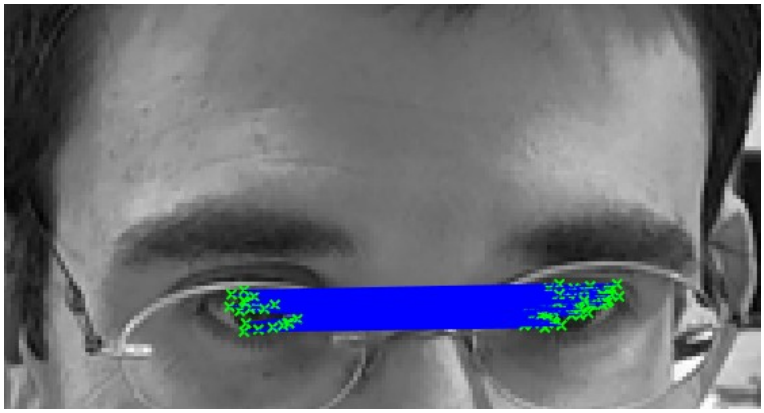
deve ser grande o suficiente para que os centros pertençam a olhos diferentes, e também o ângulo entre esses centros com respeito à uma linha horizontal, visto que se o ângulo entre os centros for elevado, um dos centros é um falso positivo ou então ambos os centros estão no mesmo olho.

Portanto, as condições para a formação de um par de olhos podem ser descritas por

$$Par_{(C_1,C_2)} = \begin{cases} 1, & \text{se} \begin{cases} 0.20 \times Face_l \leq Dist_{(C_1,C_2)} \leq 0.80 \times Face_l \\ \theta_{(C_1,C_2)} < 30^\circ \end{cases} \\ 0, & \text{caso contrário} \end{cases} \tag{39}$$

onde $Dist_{(C_1,C_2)}$ é a distância entre ambos os centros e $\theta_{(C_1,C_2)}$ é o ângulo entre os centros. A Figura 57 ilustra todos os pares de centros interligados.

Figura 57 – Pares de centros representados em verde com ligações representadas em azul



Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

Vale ressaltar que qualquer centro que não for capaz de gerar um par com nenhum outro centro é descartado.

3.8 AVALIAÇÃO DOS PARES DE CENTRO E DEFINIÇÃO DAS COORDENADAS DE CENTRO DE OLHO FINAIS

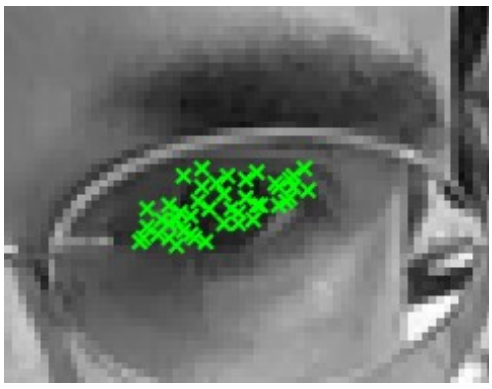
Conforme descrito na Seção 3.7, cada centro de olho possui uma nota de 1 a 11, a depender da quantidade de janelas identificadas como olho durante a etapa de classificação das janelas com diferentes escalas de tamanho. Baseado nesta nota, define-se que um par de olhos perfeito é aquele em que ambos os centros possuem nota máxima, cuja nota do par portanto resulta em 22. Caso contrário, considera-se que o par é imperfeito.

Especificamente, apenas esses pares perfeitos (pode existir mais de um par perfeito) são processados na etapa seguinte, com os pares imperfeitos sendo descartados. Caso não exista nenhum par considerado perfeito, então todos os pares imperfeitos encontrados são processados.

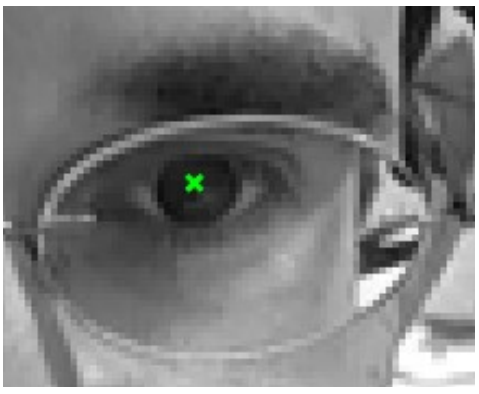
Após a aplicação dessas condições, os candidatos a centros de olhos são agrupados em dois grupos (isto é, candidatos a centros dos olhos esquerdo e direito, respectivamente). Tal agrupamento é realizado visando estimar um único centro para cada um dos olhos. Para este agrupamento, utiliza-se o algoritmo de *k-means*. Essencialmente, esse algoritmo busca separar as amostras de um conjunto por meio da determinação do centroide que melhor representa os dois grupos existentes (MACQUEEN, 1967). A Figura 58 ilustra a região de olho antes e depois do agrupamento realizado pelo algoritmo de k-means.

Figura 58 – Centros de um olho antes e depois do agrupamento por *k-means*

(a) Centros de olhos identificados antes do agrupamento por *k-means*



(b) Centro identificado após agrupamento por *k-means*



Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

As coordenadas encontradas de ambos os olhos após esse agrupamento são armazenadas em um vetor, e o processamento é encerrado. A Figura 59 mostra ambos os olhos detectados no rosto identificado.

Figura 59 – Olhos identificados na imagem de entrada

(a) Olhos marcados na imagem de rosto identificada

(b) Olhos marcados na imagem de entrada em escala de cinza

Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

3.9 DIFERENÇAS ENTRE O ALGORITMO PROPOSTO E O ALGORITMO DE AHMED E LASKAR (AHMED; LASKAR, 2019)

Conforme mencionado na Seção 3.1, o algoritmo desenvolvido neste trabalho é baseado no algoritmo apresentado em (AHMED; LASKAR, 2019), contendo algumas diferenças que são discutidas nas seções seguintes. As diferenças de implementação visaram em parte simplificar o algoritmo, enquanto outras diferenças são soluções próprias para realização de etapas de processamento que não são muito bem descritas no trabalho original (AHMED; LASKAR, 2019).

3.9.1 Identificação dos centroides das características de borda

Na Seção 3.4 utiliza-se a métrica de $Dist_{(pse,nse)}$ para avaliar quais características de borda PSE e NSE estão próximas o suficiente para compor uma característica *sCES* ou *sEES*, utilizando as equações (30) e (31).

No trabalho original, os autores não descrevem como essa distância é calculada. Dessa forma, neste trabalho optou-se por utilizar o centroide (Equações (33) e (34)) das características de borda PSE e NSE como ponto de referência para o cálculo dessas distâncias, visto que representa de fato o *pixel* central dessas características de borda. Além disso, esses centroides são utilizados posteriormente para o cálculo do centro relativo entre uma PSE e uma NSE, por meio das Equações (36) e (37).

3.9.2 *SVM* utilizada e conjunto de treinamento

No artigo (AHMED; LASKAR, 2019), foi utilizado um classificador *SVM* com kernel de base radial (Kernel RBF). Além disso, os vetores de características utilizados para treinar esse classificador foram retirados de um total de 27052 imagens de olho e 74857 imagens de não-olho.

No algoritmo implementado neste trabalho, foi utilizado uma *SVM* cúbica, visto que apresentou os melhores resultados por meio do aplicativo *Classification Learner* do *Matlab*. Além disso, utilizou-se uma quantidade menor de imagens no treinamento (1500 imagens de olhos e 1500 imagens de não-olhos), o que se mostrou suficiente para obtenção de um classificador com alta precisão.

3.9.3 Método para identificação do par de olhos final

Após a identificação dos pares de centros de olhos, etapa ilustrada pela Figura 57, Ahmed e Laskar utilizam um método baseado na extração de gradientes locais para identificar o par de olhos final, modificando a abordagem proposta por Timm e Barth (TIMM; BARTH, 2011). Ao utilizar esse método, um dos pares é então escolhido com base na maior nota atribuída.

No algoritmo proposto, visando simplificar a implementação desta etapa, todos os pares de centro de olho definidos como par perfeito (ou alternativamente todos os pares imperfeitos caso não existam pares perfeitos) são agrupados por meio do algoritmo de *k-means* (veja Seção 3.8).

4 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos com o algoritmo de localização de centros de olhos implementado, assim como é realizada uma comparação com com os resultados apresentados em (AHMED; LASKAR, 2019).

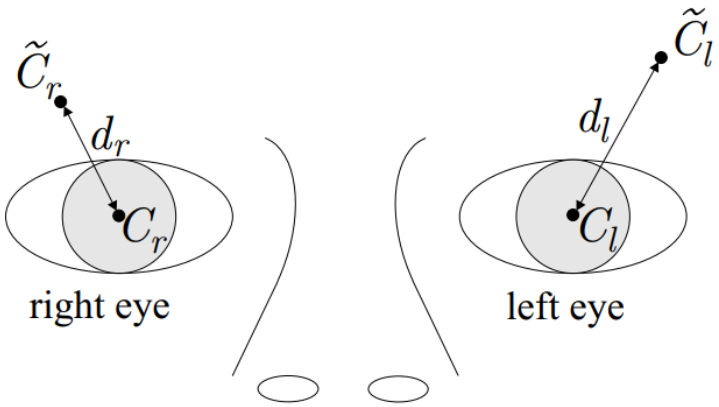
4.1 MÉTRICA PARA AVALIAÇÃO DE ACURÁCIA DO ALGORITMO

Para avaliar os resultados encontrados utilizou-se uma métrica de erro relativo baseada nas distâncias entre as coordenadas reais dos olhos na imagem de entrada e as coordenadas identificadas pelo algoritmo. Matematicamente, este erro relativo pode ser descrito por (JESORSKY; KIRCHBERG; FRISCHHOLZ, 2001):

$$N_{err} = \frac{max(d_l, d_r)}{||C_l - C_r||} \tag{40}$$

onde d_l corresponde à distância entre o centro real direito (C_r) e o centro direito identificado ($\tilde{C}_r$) e d_r corresponde à distância entre o centro real esquerdo (C_l) e o centro esquerdo identificado ($\tilde{C}_l$), como ilustra a Figura 60.

Figura 60 – Relações entre os centros reais e os centros estimados das posições dos olhos



Fonte: (JESORSKY; KIRCHBERG; FRISCHHOLZ, 2001)

Quanto menor for o valor de N_{err}, menor é a distância entre os centros reais e os centros identificados e mais precisa é a estimativa da posição dos centros dos olhos obtidas pelo algoritmo. Para definir a taxa de acerto do algoritmo, considera-se que os centros dos olhos são localizados corretamente quando $N_{err} < \alpha$. Aqui, para fins de avaliação são computados as taxas de acerto obtidas para $\alpha = \{0.05, 0.10, 0.25\}$ (JESORSKY; KIRCHBERG; FRISCHHOLZ, 2001).

Tabela 2 – Tabela comparativa dos resultados obtidos à partir do processamento do *Dataset GI4E Database*

GI4E Database	$N_{err} < 0.05$	$N_{err} < 0.1$	$N_{err} < 0.25$
Algoritmo (C/ erros de Viola-Jones)	72.77%	87.03%	88.89%
Algoritmo (S/ erros de Viola-Jones)	79.25%	94.79%	96.82%
(AHMED; LASKAR, 2019)	89.46%	97.86%	98.36%

Fonte: O autor

Vale também ressaltar que algoritmo de Viola-Jones (utilizado na primeira etapa de processamento para localização da face) pode apresentar erros (ou seja, não localizando uma face quando ela existe na imagem). Assim, levando isso em consideração, aqui são computadas duas taxas de acerto, a primeira desconsiderando as imagens onde não são identificados rostos (devido a uma falha do algoritmo de Viola-Jones), e a segunda taxa de acerto considerando no cômputo geral da métrica todas as imagens (isto é, incluindo aquelas onde o algoritmo de Viola-Jones não é capaz de localizar o rosto).

4.2 *DATASETS* DE IMAGENS UTILIZADOS PARA AVALIAÇÃO DOS RESULTADOS DO ALGORITMO

Especificamente, o algoritmo implementado neste trabalho foi testado em três diferentes *datasets*, a saber: *Gaze Interaction for Everybody - GI4E Database* (UPNA, 2022) e *bioID Face Database* (THE. . . , 2001), ambos utilizados por Ahmed e Laskar para testar os resultados encontrados em seu artigo (AHMED; LASKAR, 2019), e o *Georgia Tech Face Database - GTdb* (GEORGIA. . . , 2022). Os resultados obtidos em cada um dos *datasets* são apresentados e discutidos nas seções seguintes.

4.2.1 Gaze Interaction for Everybody - GI4E Database

O *GI4E Database* é composto por 1236 imagens, contendo o rosto de 103 indivíduos. Todas as imagens são coloridas e possuem a mesma resolução de 800×600 *pixels*, e na grande maioria de suas imagens os olhos estão bem visíveis e não obstruídos por nenhum objeto. Além disso, as coordenadas dos olhos são fornecidas para todas as imagens deste *dataset*, permitindo computar a taxa de acerto (do algoritmo implementado) para cada imagem. A Figura 61 mostra alguns exemplos de imagens desse *dataset*.

É válido ressaltar que uma parte deste *dataset* foi utilizada para treinar o classificador *SVM* do algoritmo, portanto há uma chance maior dos olhos serem melhor reconhecidos no processamento dessas imagens. Especificamente, o algoritmo de localização de centros de olhos implementado processou todas as imagens do *dataset* com os resultados sendo apresentados na Tabela 2. Nesta tabela também são mostrados os resultados obtidos pelo algoritmo proposto por Ahmed e Laskar.

Figura 61 – Imagens do *dataset GI4E Database*

Fonte: GI4E Database - Universidad Pública de Navarra

A Figura 62 mostra exemplos de rostos no *Dataset* em que os olhos foram identificados corretamente, com $N_{err} < 0.25$, enquanto a Figura 63 mostra exemplos de olhos não identificados ou identificados com $N_{err} > 0.25$. Observe que na Figura 62(b), os olhos foram identificados corretamente mesmo com o indivíduo utilizando um óculos. Caso adereços e adornos estejam próximos dos olhos mas não afetem as principais características desses olhos, ele não será um empecilho na identificação correta dos centros. Em contrapartida, a Figura 63(a) e 63(b) possuem inconsistências na identificação de centro de olho pois características faciais muito próximas dos olhos apresentaram formas semelhantes à aquelas julgadas importantes pelo algoritmo como sendo características de olhos. Esses exemplos servem para ilustrar que as próprias características faciais, como sobrancelhas, cílios ou olheiras muito expressivas podem causar interferência no processamento.

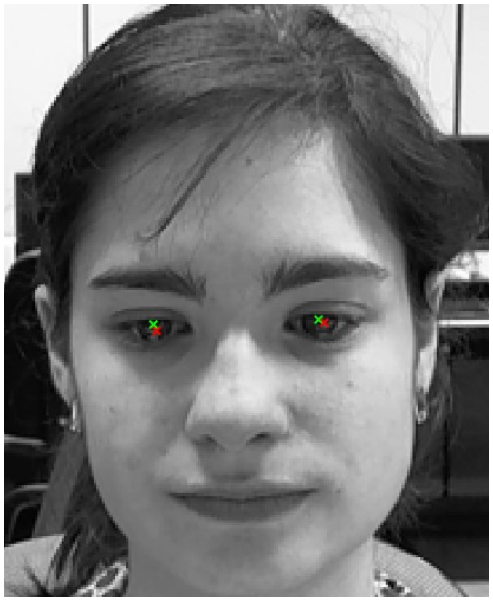
4.2.2 BioID Face Database

O *BioID Face Database* é um *dataset* de 1521 imagens de 384×286 *pixels* em escala de cinza, contendo rostos de 23 pessoas diferentes realizando diversas expressões faciais. A Figura 64 mostra alguns exemplos de imagem desse *dataset*.

O processamento deste conjunto de imagens representa um desafio maior ao algoritmo, visto que em muitos rostos os olhos estão obstruídos, fechados ou a pessoa está usando adereços que podem dificultar a identificação de um olho, assim como menor resolu-

Figura 62 – Olhos do *dataset GI4E Database* corretamente identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} = 0.043$



(b) Imagem com $N_{err} = 0.024$

(c) Imagem com $N_{err} = 0.041$

Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

ção das imagens quando comparado ao *dataset GI4E Database*, que pode também dificultar o processamento. A Tabela 3 mostra os resultados obtidos pelo algoritmo implementado e pelo algoritmo proposto por Ahmed e Laskar.

A Figura 65 mostra exemplos de rostos no *Dataset* em que os olhos foram identi-

Figura 63 – Olhos do *dataset GI4E Database* não identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} > 0.25$

(b) Imagem com $N_{err} > 0.25$

(c) Imagem com olhos não identificados

Fonte: O autor, adaptado de GI4E Database - Universidad Pública de Navarra

ficados corretamente, com $N_{err} < 0.25$, enquanto a Figura 66 mostra exemplos de olhos não identificados ou identificados com $N_{err} > 0.25$. Por meio da Figura 66(a) e 66(b), fica evidente o quanto a obstrução dos olhos, normalmente resultante do uso de óculos de cabelos longos, afetam o desempenho do algoritmo. Óculos em particular geram problemas caso existam muitos reflexos nas lentes ou se as bordas da armação são muito espessas. No caso da Figura 66(a), o algoritmo acusou o centro de olho de estar na armação do óculos. Já na Figura 66(b), não foi detectado nenhum par de olhos na imagem. A Figura 67 mostra os candidatos à centros de olho obtidos a partir da análise das características *sCES* e *sEES*.

Figura 64 – Imagens do *dataset BioID Face Database*

Fonte: BioID Face Database

Tabela 3 – Tabela comparativa dos resultados obtidos à partir do processamento do *Dataset BioID Face Database*

BioID Face Database	Nerr <0.05	Nerr <0.1	Nerr <0.25
Algoritmo Proposto (C/ erros de Viola-Jones)	32.87%	54.70%	59.10%
Algoritmo Proposto (S/ erros de Viola-Jones)	39.80%	66.24%	71.57%
(AHMED; LASKAR, 2019)	86.98%	95.20%	99.00%

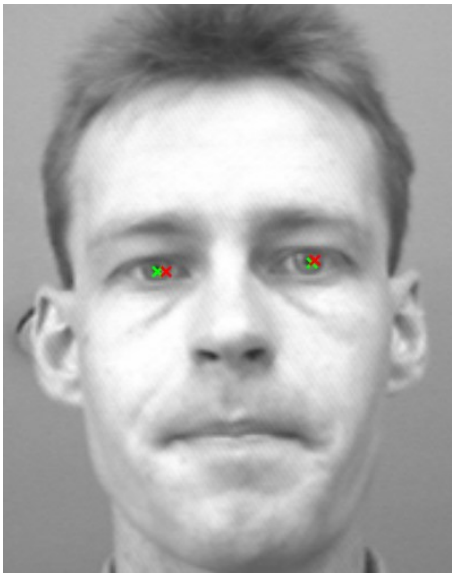
Fonte: O autor

Observe pela Figura 67(a) que as características semi-circulares (*sCES*) que são responsáveis por identificar as características da íris do olho não foram corretamente identificadas. Caso o algoritmo processe apenas os centros de olhos advindos de características *sCES*, ele seria incapaz de identificar um olho. Entretanto, é possível observar na Figura 67(b) que as características *sEES* foram identificadas em regiões próximas a olhos, mas mesmo assim um olho não foi identificado, o que leva ao segundo problema que pode ocorrer durante o processamento: o classificador *SVM* não foi capaz, em alguns casos, de distinguir uma região de olho de uma região de não-olho.

Estes resultados podem ter como causa a quantidade relativamente pequena de imagens utilizadas para treinar o classificador *SVM*. Além disso, no treinamento foi utilizada uma seleção de olhos sem qualquer tipo de obstrução, reflexos ou olhos semi-cerrados. Ao encontrar uma imagem particularmente desafiadora, mesmo que os processamentos

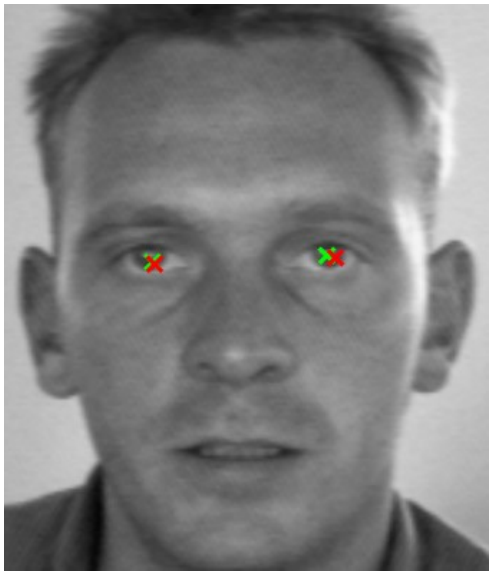
Figura 65 – Olhos do *dataset BioID Face Database* identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} = 0.025$



(b) Imagem com $N_{err} = 0.026$

(c) Imagem com $N_{err} = 0.050$



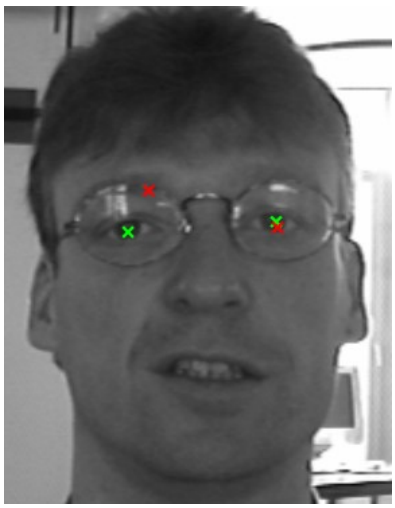
Fonte: O autor, adaptado de BioID Face Database

anteriores sejam bem sucedidos e alguns candidatos de olhos estejam de fato localizados na região do olho na imagem, é possível que ele não seja capaz de identificar que aquela região corresponda à um olho, impossibilitando o retorno de uma coordenada de pares ao final do processamento.

Esse problema se torna mais evidente ao avaliar as características *sCES* e *sEES* na Figura 68, referente à imagem da Figura 67(c). Note pela Figura 68(a) e 68(b) que tanto as características *sCES* como as características *sEES* sinalizam centros ao redor da

Figura 66 – Olhos do *dataset BioID Face Database* não identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} > 0.25$



(b) Imagem com olhos não identificados

(c) Imagem com olhos não identificados

Fonte: O autor, adaptado de BioID Face Database

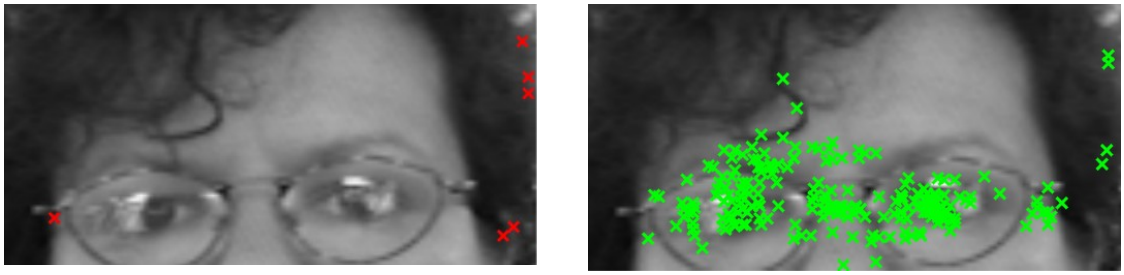
região de olhos, porém ainda assim os olhos não foram identificados, significando falha do classificador SVM.

4.2.3 Georgia Tech Face Database - GTdb

O último *dataset* testado foi o *GTdb* e, dentre todos, é o *dataset* que apresenta o maior desafio ao algoritmo, visto que muitas das imagens possuem rostos inclinados, olhos fechados ou semi-cerrados e condições diversas de luminosidade e escalas de tamanho. Esse *dataset* possui um total de 750 imagens coloridas com 640×480 *pixels* de resolução, retiradas de 50 diferentes indivíduos. A Figura 69 mostra alguns exemplos de imagem

Figura 67 – Centros candidatos à olhos baseados nas Características *sCES* e *sEES* da Figura 66(b)

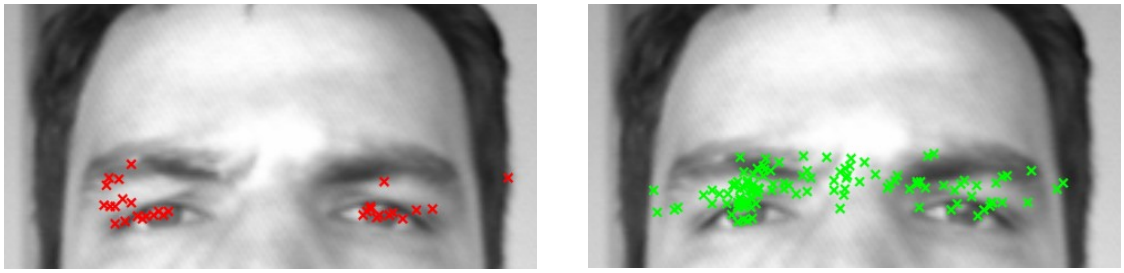
(a) Características *sCES* (b) Características *sEES*



Fonte: O autor, adaptado de BioID Face Database

Figura 68 – Centros candidatos à olhos baseados nas Características *sCES* e *sEES* da Figura 66(c)

(a) Características *sCES* (b) Características *sEES*



Fonte: O autor, adaptado de BioID Face Database

Tabela 4 – Tabela comparativa dos resultados obtidos à partir do processamento do *Dataset Georgia Tech Face Database - GTdb*

GTdb Face Database	Nerr <0.05	Nerr <0.1	Nerr <0.25
Algoritmo Proposto (Com erros de Viola-Jones)	24.00%	43.06%	50.40%
Algoritmo Proposto (Sem erros de Viola-Jones)	29.17%	52.35%	61.26%

Fonte: O autor

desse *dataset*.

Para este caso, as coordenadas de centros de olhos não estavam mapeadas, portanto foi necessário manualmente extrair todas as coordenadas destes centros, para que fosse possível computar a métrica de desempenho obtida pela algoritmo de localização de centro de olho implementado. A Tabela 4 mostra os resultados obtidos pelo algoritmo desenvolvido.

A Figura 70 mostra exemplos de rostos no *dataset* em que os olhos foram identificados corretamente, com $N_{err} < 0.25$, enquanto a Figura 71 mostra exemplos de olhos

Figura 69 – Imagens do *dataset GTdb*

Fonte: Georgia Tech Face Database - GTdb

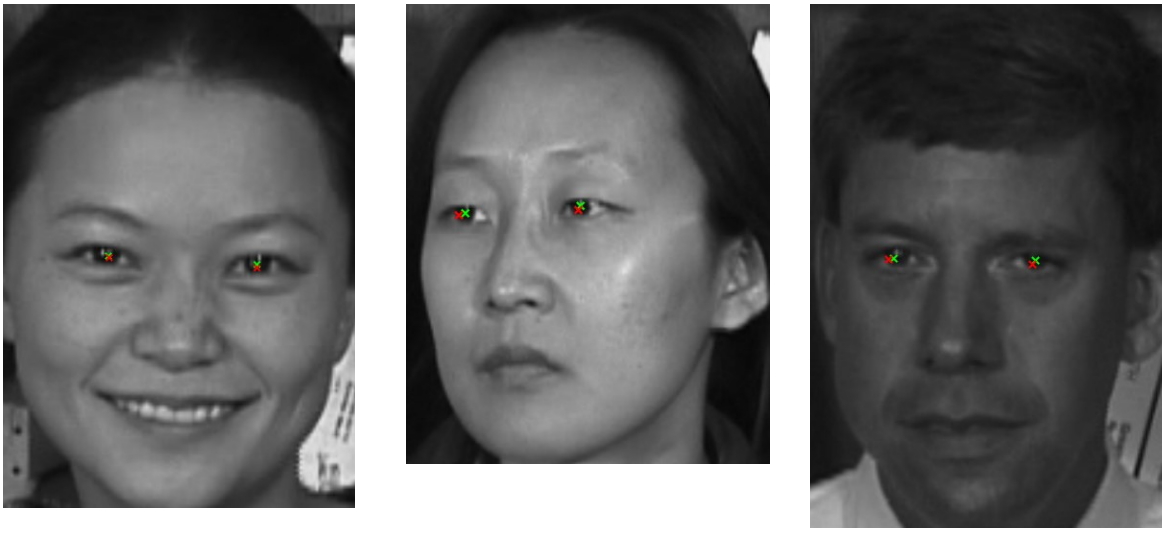
não identificados ou identificados com $N_{err} > 0.25$. Na Figura 71(b) e 71(c), fica bem evidente a dificuldade em identificar olhos semi-cerrados ou completamente fechados. Essas imagens praticamente impossibilitam o algoritmo de detectar características *sCES* no rosto, forçando o algoritmo a identificar a localização dos olhos apenas por meio das características *sEES*. Além disso, mesmo se a localização for encontrada, o classificador SVM pode não identificar os centros candidatos à olho, visto que ele não foi treinado com imagens de olhos fechados.

4.3 DISCUSSÃO GERAL DOS RESULTADOS

De maneira geral, o algoritmo implementado apresenta resultados positivos. Ao considerar imagens que possuam as condições de entrada apresentadas no início do Capítulo 3, com rostos identificados, as coordenadas de centros de olhos são identificadas corretamente. Além disso, o algoritmo detecta com consistência as características *sEES* e

Figura 70 – Olhos do *dataset BioID Face Database* identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} = 0.021$ (b) Imagem com $N_{err} = 0.054$ (c) Imagem com $N_{err} = 0.037$



Fonte: O autor, adaptado de GTdb

sCES em um rosto nas posições corretas da região do olho, o que é essencial para que os candidatos à centros de olhos sejam definidos. Entretanto, quando as imagens de entrada se mostram mais desafiadoras, com rostos inclinados, contendo reflexos de luz na região dos olhos ou imagens em que os olhos estão fechados ou obstruídos, o algoritmo tem seu desempenho reduzido. De acordo com os resultados encontrados, as principais razões que levam à falha de identificação do olho são:

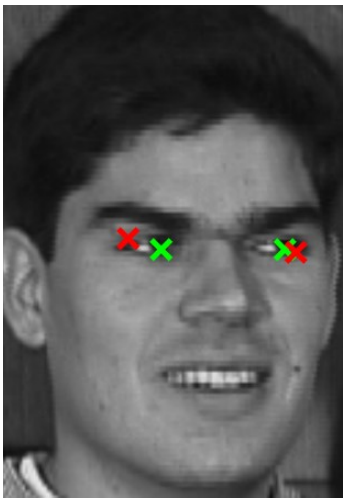
- Alta taxa de erros apresentada pelo algoritmo para detecção facial de Viola-Jones.
- Alta taxa de falsos negativos no processo de avaliação utilizando o classificador SVM.

Uma possível maneira de contornar alguns problemas trazidos pelo algoritmo de Viola-Jones seria utilizar um algoritmo mais robusto para detecção de rostos. Outro problema apresentado pelo algoritmo de Viola-Jones é a detecção de múltiplos rostos em uma imagem que só continha um. Para mitigar esse problema, o algoritmo todo deve ser re-adaptado para processar múltiplos rostos em uma imagem. Da forma como o algoritmo é implementado neste trabalho, na ocasião de múltiplos rostos serem encontrados apenas o maior rosto encontrado é processado, e caso este maior rosto encontrado é um falso positivo de Viola-Jones, muito provavelmente nenhum par de olhos será detectado.

Já para a questão do treino do classificador, a principal razão para a sua alta de taxa de falsos negativos é a quantidade de imagens fornecidas para o treinamento, assim como a sua baixa variedade. Todas as imagens utilizadas para seu treinamento foram

Figura 71 – Olhos do *dataset BioID Face Database* não identificados pelo algoritmo. A marcação verde denota a posição real e a vermelha denota a posição identificada.

(a) Imagem com $N_{err} > 0.25$



(b) Imagem com olhos não identificados

(c) Imagem com olhos não identificados

Fonte: O autor, adaptado de GTdb

olhos abertos e sem nenhum tipo de obstrução ou exemplares de olhos semi-cerrados. Isso dificulta a detecção de olhos em *Datasets* mais desafiadores, como o *bioID* ou o *GTdb*.

5 CONCLUSÕES

O presente trabalho teve como objetivo a implementação de um algoritmo para localização de centros de olhos em imagens digitais. A localização precisa de centro de olhos é necessária em diversas aplicações, como sistemas de segurança ou para a indústria do entretenimento. No Capítulo 1 foi apresentada uma contextualização do trabalho em relação às aplicações e dificuldades existentes na identificação de olhos por meio de sistemas de visão computacional, assim como uma apresentação das abordagens já existentes.

O Capítulo 2 revisitou a fundamentação teórica deste trabalho, apresentando a revisão de alguns conceitos como: definições de imagem digital, imagem colorida e imagem em escala de cinza, a introdução de conceitos essenciais para a manipulação dessas imagens, como as transformações geométricas e morfológicas, e também uma descrição detalhada de algoritmos clássicos de processamento digital de imagem como o algorítmo de Canny e o algorítmo de Viola-Jones.

No Capítulo 3 foi apresentado em detalhes todas as etapas de processamento realizadas pelo algoritmo implementado. De maneira sucinta, o algoritmo consiste em: *i*) encontrar as características semi-circulares e semi-elípticas no entorno dos olhos no rosto detectado da imagem; *ii*) classificar os candidatos à centro de olho (obtidos na etapa anterior) por meio de um classificador SVM; e *iii*) encontrar pares de olhos válidos e reduzi-los a um único par de olhos.

Por fim, o Capítulo 4 apresentou uma análise dos resultados obtidos através do processamento de três diferentes *datasets*, possibilitando tanto ilustrar a eficácia do algoritmo implementado como as maiores dificuldades encontradas na identificação dos olhos em um rosto. Além disso, foi possível comparar os resultados obtidos neste trabalho com os resultados apresentados no artigo de Ahmed e Laskar (AHMED; LASKAR, 2019), no qual este trabalho foi baseado.

5.1 DESENVOLVIMENTOS FUTUROS

Como uma forma de melhorar os resultados obtidos pelo algoritmo desenvolvido neste trabalho, além de poder adaptar o algoritmo para novas funcionalidades, os seguintes melhoramentos podem ser realizados no futuro:

- Adaptar o algoritmo para realizar a leitura de múltiplos rostos em uma mesma imagem, retornando as coordenadas dos pares de olhos de todos os rostos presentes na imagem.
- Treinar o classificador SVM com uma gama amostral maior e mais diversificada de imagens, aumentando sua taxa de acerto em situações mais desafiadores.
- Implementar e adaptar o algoritmo para realizar processamento de imagens em tempo real, viabilizando utilizar o algoritmo de localização de olhos no

processamento de vídeos obtidos de câmeras de monitoramento.

REFERÊNCIAS

AHMED, Manir; LASKAR, Rabul Hussain. Eye center localization in a facial image based on geometric shapes of iris and eyelid under natural variability. **Image and Vision Computing**, v. 88, p. 52–66, 2019. ISSN 0262-8856.

CANNY, John F. A Computational Approach to Edge Detection. **IEEE Trans. Pattern Anal. Mach. Intell.**, v. 8, n. 6, p. 679–698, 1986.

CHANG, Tang-Hsien; CHEN, Yi-Ru. Driver fatigue surveillance via eye detection. *In*: 17TH International IEEE Conference on Intelligent Transportation Systems (ITSC). [*S.l.*: *s.n.*], 2014. P. 366–371.

CHERNOV, Vladimir; ALANDER, Jarmo; BOCHKO, Vladimir. Integer-based accurate conversion between RGB and HSV color spaces. **Computers Electrical Engineering**, v. 46, p. 328–337, 2015. ISSN 0045-7906.

CONGALTON, Russell G. Exploring and evaluating the consequences of vector-to-raster and raster-to-vector conversion. **Photogrammetric Engineering and Remote Sensing**, [Falls Church, Va.] American Society of Photogrammetry., v. 63, n. 4, p. 425–434, 1997.

CORCORAN, Peter M.; NANU, Florin; PETRESCU, Stefan; BIGIOI, Petronel. Real-time eye gaze tracking for gaming design and consumer electronics systems. **IEEE Transactions on Consumer Electronics**, v. 58, n. 2, p. 347–355, 2012.

CORKE, Peter I. **Robotics, Vision & Control: Fundamental Algorithms in MATLAB**. Second. [*S.l.*]: Springer, 2017. ISBN 978-3-319-54413-7.

CORTES, Corinna; VAPNIK, Vladimir. Support-Vector Networks. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 20, n. 3, p. 273–297, set. 1995. ISSN 0885-6125.

DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. **Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on**, v. 1, p. 886–893, 2005.

FARKAS, Leslie *et al.* International Anthropometric Study of Facial Morphology in Various Ethnic Groups/Races. **The Journal of craniofacial surgery**, v. 16, p. 615–46, ago. 2005.

FIUME, Eugene Lucas. **The Mathematical Structure of Raster Graphics**. 1. ed. Toronto: Academic Press, INC, 1989. P. 2.

FREUND, Yoav; SCHAPIRE, Robert E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000.

GEORGIA Tech Face Database. Disponível em: http://www.anefian.com/research/face_reco.htm. Acesso em: 28 ago. 2022.

GHAZALI, Kamarul Hawari Bin; MA, Jie; XIAO, Rui. An Innovative Face Detection based on Skin Color Segmentation. **International Journal of Computer Applications**, Volume 34, n. 02, 2011.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital image processing**. [*S.l.*]: Prentice Hall, 2008. ISBN 9780131687288 013168728X 9780135052679 013505267X.

GUPTA, Rohan. **Breaking Down Facial Recognition: The Viola-Jones Algorithm**. 2019. Disponível em: https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999. Acesso em: 4 jul. 2022.

HAYKIN, Simon S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

JESORSKY, Oliver; KIRCHBERG, Klaus J; FRISCHHOLZ, Robert W. Robust face detection using the hausdorff distance. *In*: SPRINGER. INTERNATIONAL conference on audio-and video-based biometric person authentication. [*S.l.*: *s.n.*], 2001. P. 90–95.

KIM, Hyunjun; JO, Jaeik; TOH, Kar-Ann; KIM, Jaihie. Eye detection in a facial image under pose variation based on multi-scale iris shape feature. **Image and Vision Computing**, v. 57, p. 147–164, 2017. ISSN 0262-8856.

KUMAR, Anup; BATUT, Bérénice. Machine learning: classification and regression. Galaxy Training Network, 1970.

LEE, Socret. **Understanding Face Detection with the Viola-Jones Object Detection Framework**. 2020. Disponível em: https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14. Acesso em: 4 jul. 2022.

LORENA, Ana Carolina; CARVALHO, André C. P. L. F. de. Uma Introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, dez. 2007.

MACQUEEN, J. B. Some Methods for Classification and Analysis of MultiVariate Observations. *In*: CAM, L. M. Le; NEYMAN, J. (Ed.). **Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability**. [*S.l.*]: University of California Press, 1967. v. 1, p. 281–297.

MAIA, Deise; TRINDADE, Roque. Face Detection and Recognition in Color Images under Matlab. **International Journal of Signal Processing, Image Processing and Pattern Recognition**, v. 9, p. 13–24, fev. 2016.

MARTIN, D R; FOWLKES, C C; MALIK, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. **TPAMI**, v. 26, n. 5, p. 530–549, 2004. ISSN 0162-8828.

MITCHELL, Tom M. **Machine learning, International Edition**. [*S.l.*]: McGraw-Hill, 1997. (McGraw-Hill Series in Computer Science). ISBN 978-0-07-042807-2.

OJALA, Timo; PIETIKÄINEN, Matti; HARWOOD, David. A comparative study of texture measures with classification based on featured distributions. **Pattern Recognition**, v. 29, n. 1, p. 51–59, 1996. ISSN 0031-3203.

PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A.; JAWAHAR, C. V. Cats and Dogs. *In*: IEEE Conference on Computer Vision and Pattern Recognition. [*S.l.*: *s.n.*], 2012.

RAHMAN, Muh. Arif; EDY PURNAMA, I Ketut; PURNOMO, Mauridhi Hery. Simple method of human skin detection using HSV and YCbCr color spaces. *In*: 2014 International Conference on Intelligent Autonomous Agents, Networks and Systems. [*S.l.*: *s.n.*], 2014. P. 58–61.

SANTOS, Thiago.; AIRES, Kelson.; SILVA, Romuere. e; LIMA, Kalyf.; VERAS, Rodrigo.; SOARES, André. Histograma de Gradientes Orientados na Detecção de Motocicletas. **JIM 2012 - IV Jornada de Informática do Maranhão**, 2012.

SKODRAS, Evangelos; FAKOTAKIS, Nikos. Precise localization of eye centers in low resolution color images. **Image and Vision Computing**, v. 36, p. 51–60, 2015. ISSN 0262-8856.

SOUTO, Marta Poncet; LORENA, Ana Carolina; DELBEM, Alexandre C. B.; CARVALHO, André C. P. L. F. de. Técnicas de aprendizado de máquina para problemas de biologia molecular. *In*.

SZELISKI, Richard. **Computer Vision - Algorithms and Applications.** [*S.l.*]: Springer, 2011. P. i–xx, 1–812. (Texts in Computer Science). ISBN 978-1-84882-935-0.

THE BioID Face Database. 2001. Disponível em: https://www.bioid.com/facedb/. Acesso em: 28 ago. 2022.

TIMM, Fabian; BARTH, Erhardt. Accurate Eye Centre Localisation by Means of Gradients. *In*: p. 125–130.

TRINDADE, Fernando Henrique Vieira. Estudo de técnicas de visão computacional para auxílio à acessibilidade. Criação Editora, 2018.

UPNA. **GI4E - Gaze Interaction for Everybody**. Disponível em: https://www.unavarra.es/gi4e/databases/gi4e/. Acesso em: 28 ago. 2022.

VILLANUEVA, Arantxa; CABEZA, Rafael; PORTA, Sonia. Eye tracking: Pupil orientation geometrical modeling. **Image and Vision Computing**, v. 24, n. 7, p. 663–679, 2006. ISSN 0262-8856.

VILLANUEVA, Arantxa; PONZ, Victoria; SESMA-SANCHEZ, Laura; ARIZ, Mikel; PORTA, Sonia; CABEZA, Rafael. Hybrid Method Based on Topography for Robust Detection of Iris Center and Eye Corners. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)**, v. 9, ago. 2013.

VIOLA, Paul A.; JONES, Michael J. Rapid Object Detection using a Boosted Cascade of Simple Features. *In*: CVPR (1). [*S.l.*]: IEEE Computer Society, 2001. P. 511–518.