UNIVERSIDADE FEDERAL DO PARANÁ

DENILSON LOPES

JHONATHAN DE SOUZA LIMA

DESENVOLVIMENTO DE SISTEMA ANTIFURTO PARA VEÍCULOS, COM USO DE RECONHECIMENTO BIOMÉTRICO DA ÍRIS

CURITIBA

2019

DENILSON LOPES
JHONATHAN DE SOUZA LIMA

DESENVOLVIMENTO DE SISTEMA ANTIFURTO PARA VEÍCULOS, COM USO DE RECONHECIMENTO BIOMÉTRICO DA ÍRIS

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia Elétrica do Setor de Tecnologia da Universidade Federal do Paraná como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientadora: Profª. Dra. Giselle Lopes F. Ronque

CURITIBA

2019

TERMO DE APROVAÇÃO

DENILSON LOPES
JHONATHAN DE SOUZA LIMA

DESENVOLVIMENTO DE SISTEMA ANTIFURTO PARA VEÍCULOS, COM USO DE RECONHECIMENTO BIOMÉTRICO DA ÍRIS

Trabalho de Conclusão de Curso aprovado como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica, Setor de Tecnologia da Universidade Federal do Paraná, pela seguinte banca examinadora:

_____
Orientadora: Profª. Dra. Giselle Lopes F. Ronque
Departamento de Engenharia Elétrica, UFPR

_____
Profª. Dra Viviana Cocco Mariani
Departamento de Engenharia Elétrica, UFPR

_____
Prof M.Sc. Douglas Pelacini
Departamento de Engenharia Elétrica, UFPR

Curitiba, 22 de novembro de 2019

AGRADECIMENTOS

Agradecemos a Deus, pela saúde e disposição que nos permitiu a realização deste trabalho.

À Professora Giselle Ferrari, nossa orientadora, que acreditou e nos incentivou desde o início do trabalho, aconselhando, revisando, sugerindo novas abordagens e apresentando ideias durante todo o desenvolvimento deste projeto.

Aos nossos pais, por toda a dedicação, incentivo e amor incondicional depositados em nós durante toda essa caminhada.

Aos nossos amigos, que ao longo dos anos em que frequentamos a universidade, nos deram todo o apoio necessário para que essa etapa fosse cumprida da melhor forma, sempre nos motivando nos momentos de dificuldade.

RESUMO

Os índices de furtos de veículos no Brasil mantêm-se altos, desta forma é necessário a utilização de recursos tecnológicos para a inibição da prática de furtar, de forma a desencorajar o infrator. Os sistemas de reconhecimento biométricos são amplamente difundidos nos âmbitos acadêmicos e da tecnologia e neste trabalho é desenvolvido um conceito de reconhecimento de íris para exemplificar o estado da arte Visão Computacional e Técnicas de Programação. A finalidade deste trabalho é apresentar um modelo de reconhecimento da íris para usuários devidamente cadastrado pelo sistema e inibir a ação de infratores em caso de furto interrompendo os mecanismos de ignição do veículo, através do uso de eletrônica embarcada e recursos computacionais. Os materiais (físicos e digitais) usados para a construção deste modelo são acessíveis a todos os cientistas e engenheiros e possuem baixo custo para serem implementados. Os recursos digitais utilizados compreendem bibliotecas e ambientes da linguagem de programação, assim como dispositivos encontrado no varejo de produtos eletrônicos. O resultado a ser apresentado é uma forma de validar o sistema e é medido através de métricas presentes na literatura. Ao final deste trabalho é possível inferir que os sistemas biométricos são adequados no combate a criminalidade.

Palavras-Chave: íris, reconhecimento, furto, biométricos, tecnologia.

ABSTRACT

The vehicle theft rates in Brazil remain high, so it's necessary to use technological resources to inhibit theft in order to discourage a violation. Biometric recognition systems are widely disseminated in the academic and technological fields and in this work an iris recognition concept is developed to exemplify the state of art regarding Computer Vision and Programming Techniques. The purpose of this paper is to present a iris recognition model for users properly registered by the system and to inhibit violators by interrupting the ignition mechanisms of a vehicle, through the use of embedded electronics and computational resources. The materials (physical and digital) used to construct this model are accessible to all scientists and engineers and are inexpensive to implement. The resources used in this works are comprised by libraries and programming language environments, as well as devices found in electronics retail. The result to be presented is a way to validate the system and it is measured by current metrics in the literature. At the end of this work it is possible to understand that biometric systems are suitable tool in criminality combat which is present in the country and that science and technology are fundamental to society.

Keywords: iris, recognition, theft, biometrics, technology.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EER	*Equal Error Rate* (Taxa de Igual Erro)
FA	*False Accept* (Falsa Aceitação)
FAR	*False Accept Rate* (Taxa de Falsa Aceitação)
FR	*False Reject* (Falsa Rejeição)
FRR	*False Reject Rate* (Taxa de Falsa Rejeição)
IBM	*International Business Machines Corporation*
Ipea	Instituto de Pesquisa Econômica Aplicada
IPP	*Intel's Integrated Performance*
PIL	*Python Imaging Library*
SVM	*Support Vector Machine*
T_E	Taxa de erro

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO E PROBLEMA

Os índices de criminalidade vêm mantendo-se em níveis elevados no Brasil, ao longo dos últimos anos, conforme noticiados todos os dias através dos diversos meios de comunicação. Segundo o Atlas da Violência 2019, desenvolvido pelo Instituto de Pesquisa Econômica Aplicada (Ipea), mostra que o Brasil está na lista dos países mais violentos do planeta.

As causas dessa elevada taxa de violência, no Brasil, têm origem complexa e são frequentemente atribuídas a fatores socioeconômicos, desigualdades sociais, infraestrutura precária em quase todos os campos (educação, saúde, moradia, transporte e segurança pública), desemprego, impunidade, além de um expressivo aumento populacional ocorrido nas últimas décadas.

Para Cerqueira (2016, p.36), acerca da efetividade para diminuir crimes, evidências empíricas internacionais demonstram-se favoráveis à teoria de se abrir portas às crianças e jovens em situação de vulnerabilidade socioeconômica que, sem o adequado processo de estímulo, supervisão e educação infantil, possam ser os criminosos de amanhã. Com base na literatura especializada da psicologia social, criminologia, psiquiatria e neurociência, verifica-se que o processo de desenvolvimento infanto-juvenil e os determinantes dos distúrbios comportamentais, iniciados na primeira infância, podem aumentar as possibilidades de uma trajetória no sentido da delinquência e do crime (CERQUEIRA, 2016).

Nesse incremento de delitos estão os crimes contra o patrimônio em que, segundo levantamento do Anuário Brasileiro de Segurança Pública do ano de 2019, ocorreram 247.148 roubos e 243.808 furtos de veículos no ano de 2018, no Brasil. No Estado do Paraná foram 7.874 veículos roubados e 17.620 furtados, conforme tabela constante do Anexo A, deste trabalho.

Em se tratando de uma situação de roubo, o fato deixa de representar apenas um risco ao patrimônio, mas, também, ao(s) ocupante(s) do veículo que, em alguns casos, pode(m) permanecer em poder dos criminosos durante a ação.

A possibilidade de alguém ficar na condição de refém do assaltante, no momento ou após a abordagem, deve ser considerada no projeto de um sistema de

segurança veicular. Um dos requisitos, deste projeto, é que o seu funcionamento não contribua para potencializar a ação violenta do(s) criminoso(s), contra a(s) vítima(s), ou cause acidente de trânsito que exponha a(s) vítima(s), os pedestres e/ou outros motoristas a riscos adjacentes.

Destarte, o público-alvo deste trabalho é representado por motoristas e ocupantes de veículos automotores, frequentemente vulneráveis à ação de criminosos.

1.2 JUSTIFICATIVA

Considerando o aumento da criminalidade no tocante à incidência de furtos e roubos de veículos, e suas consequências quanto aos danos materiais e à integridade física de seus usuários, este trabalho visa apresentar uma alternativa viável para mitigar esses danos e reduzir o nível de vulnerabilidade do condutor e/ou ocupantes do veículo diante de tais situações.

Os obstáculos existentes numa possível ação contra um veículo dotado com o sistema antifurto, viabilizado neste projeto, o transformarão num alvo não compensatório aos criminosos, os quais necessitam, o mais rápido possível, deixar o local do crime.

Nesse sentido, a razão desta implementação é desenvolver não somente um dispositivo antifurto, mas sim uma alternativa que desestimule e desencoraje o criminoso para tal prática, dado as dificuldades que o sistema estabelece.

Tais dificuldades possibilitam a interrupção do *iter criminis,* que é o caminho percorrido desde a ideação até a consumação do crime (VIANA, 2008, p. 27).

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um sistema antifurto de veículo com uso de eletrônica embarcada, traduzida em um dispositivo de reconhecimento biométrico de íris cuja saída possa permitir, ou não, a ignição e/ou dirigibilidade do veículo.

1.3.2 Objetivos Específicos

Os objetivos específicos do trabalho são:

- consolidar a fundamentação teórica referente às interfaces de autenticação biométrica por reconhecimento da íris.
- utilizar *hardware* comercial, adquirido no mercado, para a captura das imagens e extração dos dados biométricos;
- implementar o algoritmo para o processamento dos dados biométricos adquiridos;
- efetuar os arranjos lógicos, mecânicos e eletroeletrônicos necessários para a integração dos dispositivos de *hardware* e *software* do sistema;
- desenvolver o protótipo que execute a aquisição das características, o processamento biométrico e apresente o resultado do reconhecimento; e
- apresentar o funcionamento do protótipo, o qual deverá executar e exibir os resultados do reconhecimento e autenticação de usuários.

1.4 ESTRUTURA DO TRABALHO

Este documento está dividido em 6 capítulos, estruturados da seguinte forma:

O capítulo 1 apresenta a introdução do assunto com seu contexto e exposição do problema, a justificativa e os objetivos do trabalho, bem como, a apresentação de como se encontra estruturado.

No capítulo 2 é exposta uma revisão bibliográfica de todo o assunto contemplado neste projeto, considerados os aspectos biométricos e de processamento de imagem, assim como, o estado da arte com algumas soluções já implementadas, relacionadas ao objeto deste trabalho.

O capítulo 3 é voltado para a apresentação dos métodos aplicados na realização do trabalho e no capítulo 4 é apresentado o desenvolvimento do dispositivo antifurto e descritas a prototipagem e os algoritmos desenvolvidos.

Os resultados alcançados na elaboração deste projeto são apresentados no capítulo 5 e, por fim, no capítulo 6 são demonstradas as conclusões e sugestões para trabalhos futuros, acerca do tema.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os fundamentos teóricos necessários ao desenvolvimento deste trabalho. Após uma breve apresentação das soluções existentes, são apresentados conceitos e definições da biometria, suas aplicações e características. Em seguida, são descritos alguns princípios basilares da multibiometria, noções de sistemas modais e a comparação entre sistemas biométricos.

Os aspectos mais relevantes do reconhecimento de íris são referenciados, dos principais conceitos, definições e os métodos de processamento dessa medida biométrica.

O capítulo encerra-se com o estudo da aplicabilidade prática dos recursos eletrônicos com uso da biometria, palavra chave da configuração do sistema embarcado ora projetado.

2.1 ESTADO DA ARTE

O uso de sistemas biométricos embarcados tem sido aprimorado pela indústria automobilística nos últimos anos, os quais têm focado em desenvolver veículos que podem interagir automaticamente com os ambientes.

Nessa linha, a fabricante General Motors revelou durante o Salão de Xangai 2015, o “Chervrolet FNR”, um conceito veicular futurista autônomo, movido por eletricidade. De acordo com a empresa, o carro conta com sistema de ignição com base no reconhecimento da íris do proprietário (AUTO ESPORTE, 2015).

Em 2017, no *Consumer Eletronics Showcase(CES)* em Las Vegas (EUA), a montadora alemã Continental estreou o seu Sistema de Acesso Biométrico onde os motoristas devem ser autenticados através da impressão digital para o acionamento do motor do veículo, auxiliando contra furtos e roubos. Uma câmera interior faz reconhecimento facial para ajustar as configurações do veículo com base no usuário, garantindo e personalizando seu conforto (BRASIL ECONÔMICO, 2017).

Nesse mesmo evento, a Companhia de Eletrônicos *Gentex Corporation* apresentou um sistema biométrico com digitalização da íris incorporada no espelho retrovisor do veículo, com precisão de 99,9% na autenticação do motorista,

conforme visualizado na figura 1. Se a pessoa não é reconhecida, o sistema pode enviar um texto ao dono do carro ou limitar a funcionalidade. Se o motorista é reconhecido, tudo, desde o assento até o rádio do carro será ajustado automaticamente de acordo com as preferências do usuário.

FIGURA 1 – Sistema de Reconhecimento de Íris embarcada no espelho retrovisor do veículo

FONTE: Mobile ID World (2019).

A Chrysler, fabricante de veículos, utiliza um conceito de minivan elétrica com reconhecimento facial, através de uma câmera atrás do volante que identifica o rosto do motorista. Após a identificação inicial, o veículo manteria o perfil que inclui estações de rádio favoritas do motorista, a posição do assento, o livro de endereços, o calendário, etc (AUTOS NOVOS, 2017).

2.2 DEFINIÇÕES DE BIOMETRIA

A palavra biometria (do grego: *bios* = vida e *metron* = medida) designa o uso de **características biológicas** como a impressão digital, a voz e a retina, o formato do rosto, a geometria da mão, etc (PINOCHET, 2014).

Muitas referências bibliográficas indicam a utilização de dados biométricos a cerca de quatro mil anos na Babilônia, onde impressões digitais eram inseridas em placas de cerâmica para uso em transações comerciais, visando tanto legalizar o contrato quanto evitar falsificações (SAEED e NAGASHIMA, 2012).

A biometria é uma ciência que possibilita o reconhecimento e autenticação segura de determinada pessoa, através de suas características físicas e comportamentais (SANTOS, 2007).

2.3 REQUISITOS DE UMA CARACTERÍSTICA PARA A BIOMETRIA

As características biológicas (impressão digital, voz, íris, etc.) devem cumprir determinados requisitos específicos para serem utilizadas em medições biométricas.

Para que estes traços biológicos, físicos ou comportamentais, possam adequar-se à autenticação e, assim, validar-se numa medida biométrica “útil”, são elencados sete fatores a serem considerados nessa adequação (JAIN et al, 2004):

- universalidade: todo indivíduo possui essa determinada característica biológica;
- unicidade: a característica é exclusiva e suficientemente distinta dos demais indivíduos;
- invariância: a característica é imutável em função do tempo;
- mensurabilidade: o mensurando é passível de captura e digitalização;
- desempenho: necessidade de recursos computacionais para atingir o processamento/precisão desejado;
- aceitabilidade: o fornecimento dessa medida biométrica depende da disponibilidade do indivíduo em cedê-la; e
- resistência à falsificação, evasão: capacidade da característica em “dificultar” imitação, falsificação ou que possa facilitar a utilização de artifícios que permita indivíduos burlarem o sistema de identificação.

2.4 O QUE É UM SISTEMA BIOMÉTRICO

De uma forma geral, conforme (NEWMAN, 2010), um sistema biométrico é descrito como um sistema automatizado capaz de:

- capturar a amostra biométrica do usuário;
- extrair os dados da amostra;
- comparar os dados biométricos com modelos armazenados em um banco de dados;
- indicar se uma autenticação de identidade foi ou não alcançada.

Com referência ao modelo de banco de dados, faz-se necessário definir uma característica, o qual se traduz no modelo ou padrão utilizado como referência, e que servirá de base para as comparações dos dados biométricos lidos pelos sensores.

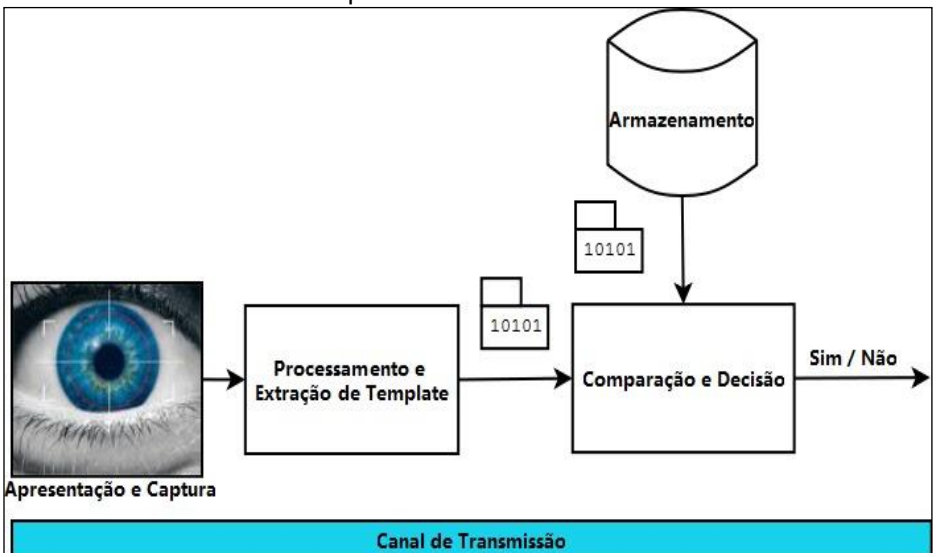
Assim sendo, um Sistema Biométrico típico tem pelo menos cinco componentes principais (CANEDO, 2010):

- Componente de apresentação e captura de dados biométricos;
- Componente de processamento e extração de características;
- Componente de armazenamento do característica;
- Componente de comparação de decisão; e
- Canal de transmissão.

2.4.1 Componentes de um Sistema Biométrico Típico

Aqui são definidos os componentes principais de um Sistema Biométrico típico, como mostrado na figura 2.

FIGURA 2 - Componentes de um Sistema Biométrico



FONTE: www.forumbiometria.com (2019).

2.4.1.1 Apresentação e captura dos dados biométricos

Apresentação do traço biométrico a um sensor (câmeras, scanners, microfones, tablets, teclados etc.) que transformará a informação em uma representação digital (foto, vídeo, áudio, etc). Etapa crítica do sistema que introduz uma componente comportamental (e psicológica) que podendo variar amplamente entre usuários, aplicações, testes laboratoriais e ambiente operacional.

A saída do sensor é uma combinação do traço biométrico, da forma como foi apresentado ao sensor e das características técnicas do sensor.

Mudanças em quaisquer desses fatores pode afetar a capacidade de reconhecer um indivíduo e até mesmo de diferenciá-lo da população.

2.4.1.2 Processamento do dado biométrico e extração de característica

A autenticação de uma impressão digital, por exemplo, significa que ela foi comparada ao modelo e apresentou “não conformidades” dentro dos padrões aceitáveis para aquela certificação.

O processamento do dado biométrico consiste na transformação da representação digital do traço biométrico nesse modelo. Normalmente isso é feito através da segmentação da representação digital, avaliação (e melhoria) da qualidade e extração de características únicas.

A extração ocorre através de diversas técnicas de processamento de imagens, reconhecimento de padrões, estatística, inteligência artificial e visão computacional.

O conjunto das características extraídas é representado matematicamente e armazenado de forma compacta e protegida, de modo a garantir uma autenticação segura, sem que esse modelo, ao longo do tempo, sofra alterações no próprio sistema de armazenamento e/ou corra o risco de ser adulterado ou roubado por ações criminosas.

2.4.1.3 Armazenamento de característica

As formas e locais de armazenamento desses modelos padrão apresentam crescente desafio e possui muitas opções de implementação.

Com a popularização da biometria, base de dados com milhões de informações se tornaram realidade e os recursos para garantir a segurança, qualidade, manutenção e gerenciamento são enormes e o risco considerável, em caso de sua perda.

Os modelos padrões podem ser armazenados, dentre outros, no próprio sensor, em um repositório central, em um cartão ou, ainda, em um computador pessoal cujo local é o mais simples, porém, o mais vulnerável. O armazenamento em cartão é atrativo por não requerer armazenamento local ou central e, ainda, o usuário poder carregar a sua própria característica (mais privacidade), podendo usá-lo em qualquer leitor autorizado.

2.4.1.4 Comparação de características e decisão

Na comparação de características, os valores matemáticos das características extraídas são comparados para determinar o grau de correlação ou similaridade, num processo chamado de *matching* (comparação). O processo de comparar duas características biométricas resulta em uma pontuação (*score)*, o que na maioria dos sistemas é relacionado através de um limiar (*threshold*) e, então, a decisão é tomada de acordo com essa relação pontuação/limiar.

O processo de “casamento” envolve a comparação de dois modelos, sendo um de cadastro (varredura do banco de dados) e um de verificação (feita somente entre amostras da mesma pessoa). O *score* dos sistemas de casamento aponta apenas o quanto as características são similares, já que não há sistema de biometria que garanta a exatidão entre cadastro e verificação.

Gerada a pontuação, o sistema biométrico irá compará-la com o limiar determinado, pelo administrador, e informar qual o valor necessário para que seja considerada um casamento de características (embora nunca sejam iguais). O limiar determina, na prática, o quão seguro um sistema biométrico vai ser.

2.4.1.5 Canal de transmissão

Comum a todos os outros sistemas de informação, é importante para a definição da arquitetura dos sistemas, política de segurança e privacidade. Pode ocorrer através de cabeamento ou redes e, cada canal tem suas particularidades que devem ser consideradas em um projeto de sistema.

2.5 MODOS DE AUTENTICAÇÃO

Em sistemas biométricos há dois modos de autenticação: a verificação e a identificação (BOLLE, 2004):

A **verificação** fundamenta-se em demonstrar a correta identidade do usuário.

A característica biométrica é apresentada pelo usuário juntamente com uma identidade alegada, usualmente por meio da digitação de um código de identificação.

Esta abordagem de autenticação é dita busca 1:1 (“um para um”), ou busca fechada, comparando a amostra com um modelo específico.

Já a **identificação** busca responder quem é o usuário. Esse usuário a ser identificado fornece apenas sua característica biométrica, competindo ao sistema “identificá-lo”.

Esta abordagem de autenticação é dita busca 1:N (“um para muitos”), ou busca aberta, em um banco de dados de perfis biométricos. O sistema busca todos os registros do banco de dados e retorna uma lista com características suficientemente similares à apresentada.

A lista retornada pode ser refinada posteriormente por comparação adicional, biometria adicional ou intervenção humana.

2.6 ERROS DE “FALSA ACEITAÇÃO” E “FALSA REJEIÇÃO”

De uma maneira geral, no ambiente técnico dos profissionais que trabalham com processos biométricos, há uma diferenciação dos vários tipos de erros, organizados conforme a localização “lógica” de sua ocorrência.

Há dois tipos principais de erros em que o comparador pode incorrer (WAYMAN, 2005):

- *False Accept (FA)*: quando uma identidade é alegada como “legítima”, quando na realidade ela é falsa.
- *False Reject (FR):* quando uma identidade é alegada como “falsa”, quando na realidade ela é legítima.

Estes dois tipos de erros devem ser considerados, frente às suas detecções, quando da escolha de um sistema biométrico, o qual pode ser categorizado por duas medidas (ROSS, 2008):

• a frequência de ocorrências de *False Accept*, ou seja, o percentual de usuários “não-autorizados” identificados, incorretamente, como usuários válidos, taxa conhecida como Taxa de Falsa Aceitação (*FAR - False Accept Rate)*;

• a frequência de ocorrências de *False Reject* , ou seja, o percentual de usuários “autorizados” que são incorretamente rejeitados, taxa essa conhecida como Taxa de Falsa Rejeição (*FRR - False Reject Rate)*.

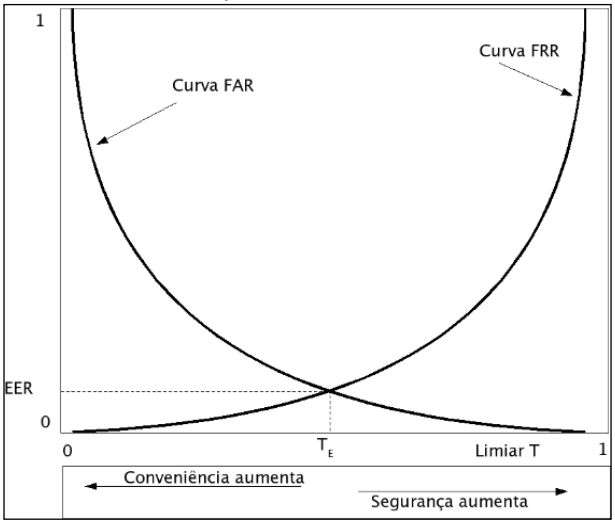
Na figura 3 verifica-se o encontro das curvas de FRR e FAR, no ponto notável de operação denominado *Equal Error Rate* (EER), ou seja, o ponto em que as duas curvas apresentam a mesma taxa de erro (T_E).

Isso significa que o sistema pode operar nas faixas de “conveniência”, abaixo de “T_E” ou de “segurança”, acima de “T_E”, conforme a calibração do limiar.

Diante disso pode-se inferir que: uma alta taxa de usuários “não autorizados” aceitos, pelo sistema, como válidos, significa comprometimento da segurança.

Já uma alta taxa de usuários “válidos” que são “impedidos” pelo sistema, significa uma inconveniência do serviço, aos olhos do usuário, causando-lhe frustração.

FIGURA 3 - Curvas típicas das taxas de erro FAR e FRR



FONTE: www.advancedsourcecode.com/minicurso_biometria.pdf (2016)

2.7 PRINCIPAIS APLICAÇÕES DA BIOMETRIA

De uma forma geral, o desenvolvimento e aplicação da biometria foram possibilitados por tecnologias de *hardware* bem como, de *softwares* e algoritmos que tornam o reconhecimento mais seguro e mais rápido (SANTOS, 2007).

Na medida em que novas tecnologias surgem e o acesso à informação fica cada vez mais facilitado, a capacidade intelectual criminosa também “evolui” e, consequentemente, crimes mais aprimorados são passíveis de realização. Embora a utilização de senhas seja uma das formas de identificação mais usada para evitar a falsificação, o uso da identificação biométrica tem se intensificado nos seus mais diversos campos de aplicação.

A biometria proporciona rapidez, praticidade e segurança, atualmente é utilizada em: controles de acesso em portas, catracas e cancelas de órgãos públicos, instalações comerciais e residenciais, substituição aos antigos relógios-ponto nas empresas, identificação de eleitores, transações bancárias, acesso a redes de computadores e dispositivos móveis, identificação criminal, etc.

Um exemplo dessa utilização são os sistemas biométricos utilizados em redes bancárias, conforme exibido na figura 4. Somente no Brasil, em 2015 já havia 90 mil caixas eletrônicos equipados com sensores biométricos e outros 70 mil passaram pela transformação (ROMANI, 2015).

FIGURA 4 - Leitor biométrico usadp em caixa eletrônico

FONTE: Folha de São Paulo (2015).

2.8 PRINCIPAIS VANTAGENS E DESVANTAGENS DA BIOMETRIA

Cada técnica biométrica possui vantagens e desvantagens em relação ao grau de certeza ou probabilidade de erro, à facilidade de aplicação, aos custos, à rapidez de resposta entre outros parâmetros (JUNIOR, 2001).

De uma maneira ampla, a biometria tem muitas vantagens sobre outros métodos de autenticação como, por exemplo: facilidade de identificação; redução das chances de roubo de identidade; dificuldade em ser duplicada; alta percepção de segurança entre os usuários; eliminação da necessidade em lembrar-se de senhas (HUGOO, 2016).

As desvantagens do uso dessa tecnologia são ligadas ao custo de implementação; à impossibilidade de substituição de dados biométricos comprometidos de alguma forma e, em alguns casos, às tecnologias biométricas que podem ser consideradas invasivas:

"*Ter de colocar o queixo em alguma engenhoca para leitura dos olhos e depois ter os dados armazenados em um banco de dados sem poder opinar sobre o assunto pode ofender algumas pessoas pela invasão da privacidade pessoal*" (HUGOO, 2016).

2.8.1 Fusão biométrica

Como condição indispensável da Multibiometria, a fusão biométrica é um termo genérico que abrange qualquer combinação das diferentes amostras obtidas, a partir de múltiplos traços, instâncias ou sensores (Relatório Técnico UnB, 2015).

A fusão pode ser executada de diferentes maneiras e associa-se a diferentes níveis da operação, desde a aquisição das amostras até o nível decisório, podendo ser em nível de amostra, de primitiva, de pontuação ou de decisão.

2.8.2 Comparação entre sistemas

Alguns fatores dificultam uma comparação direta entre os diferentes sistemas multibiométricos (Relatório Técnico UnB, 2015), dentre os quais se destacam:

• a inexistência de uma base única de dados, de larga escala, composta por diferentes amostras de traços biométricos de indivíduos;

• a falta de padronização de critérios para uma efetiva avaliação comparativa;

• a grande variedade de aplicações e formas combinacionais, as quais exigem diferentes requisitos de amostragem e desempenho.

A dimensão das bases de dados, a forma de distribuição da amostra (idade, sexo, etnia) e as condições de aquisição (ambiente, iluminação, sensor utilizado) variam em cada estudo, dificultando uma comparação mais efetiva, ainda que apenas no nível de taxas de erros (Relatório Técnico UnB, 2015).

Consideradas as limitações inerentes às características biométricas, a categoria multimodal apresenta vantagens sobre os outros tipos (Relatório Técnico UnB, 2015).

O grau de confiança na avaliação de desempenho de um sistema biométrico está diretamente relacionado ao tamanho do banco de dados usado para testes.

Sistemas unibiométricos baseados em reconhecimento da face e impressão digital, por exemplo, já foram testados em bases contendo milhões de usuários (WILSON et al, 2003).

Embora haja um ganho significativo, principalmente relacionado à maior resistência a ruídos, ao se optar pela multibiometria, o custo é uma das principais desvantagens desse sistema, uma vez que se emprega um maior número de meios (sensores, algoritmos, banco de dados) e processamentos.

Outra dificuldade na implementação multibiométrica é a aceitabilidade pelos usuários, já que a combinação de mais de uma característica biométrica tende a conferir maior resistência, tanto pela inconveniência na aquisição quanto por aspectos ligados à privacidade.

Uma ideia que pode ser passada é que quanto mais dados se tem desse usuário, mais fácil rastreá-lo (Relatório Técnico UnB, 2015).

2.9 A ÍRIS COMO SISTEMA BIOMÉTRICO

A leitura da íris humana é uma das melhores e mais caras opções de biometria e, segundo estudos do Dr. John Gustav Daugman, pesquisador da Universidade da Universidade de Cambridge, é uma tecnologia seis vezes mais segura que a utilizada na impressão digital (SANTOS, 2007).

É possível verificar, através da tabela 1, o resultado de estudo comparativo entre as medidas biométricas mais comuns, relativas aos fatores que validam as características físicas numa medida biométrica.

Os níveis alto, médio e baixo são denotados respectivamente por “3”, “2” e “1”, respectivamente (KRUEGER, 2003).

Depreende-se da tabela 1, que o reconhecimento da íris é a medida biométrica que apresenta os mais altos níveis entre as medidas comparadas, seguida pela impressão digital e retina, logo na sequência.

Avanços em estudos sobre técnicas, processos, métodos, meios, instrumentos de captura de imagens e visão computacional contribuíram bastante para o aumento da utilização desta medida biométrica.

TABELA 1 - COMPARAÇÃO ENTRE AS MEDIDAS BIOMÉTRICAS MAIS COMUNS

Medidas Biométricas	**UNIVERSALIDADE**	**UNICIDADE**	**INVARIÂNCIA**	**MENSURABILIDADE**	**DESEMPENHO**	**ACEITABILIDADE**	**EVASÃO**	**Somatório**
Assinatura	1	1	1	3	3	1	1	11
Dinâmica do caminhar	2	1	1	3	3	1	2	13
Face	3	1	2	3	3	1	1	14
Geometria da mão	2	2	2	3	2	2	2	15
Impressão digital	2	3	3	2	2	3	2	17
Íris	**3**	**3**	**3**	**2**	**1**	**3**	**3**	**18**
Retina	3	3	2	1	1	3	3	16
Voz	2	1	1	2	3	1	1	11

FONTE: Adaptado de KRUEGER, M.L. et al (2013).

2.9.1 A íris na anatomia e fisiologia ocular

Os olhos humanos, ilustrado na figura 5, assim como na maioria dos animais predadores, estão localizados na parte frontal da cabeça e fornecem ao cérebro informações essenciais, através da percepção de imagens em três dimensões. O órgão formado principalmente pelas estruturas dentre as quais se destacam (COURROL; PRETO, 2011):

Córnea: compõe a parte externa do olho, equivalente ao vidro de um relógio, e tem como função a proteção da parte anterior do globo ocular e transmitir e refratar a luz, funcionando como uma lente.

Esclerótica: parte branca do olho que tem como função a manutenção da forma e proteção das estruturas oculares.

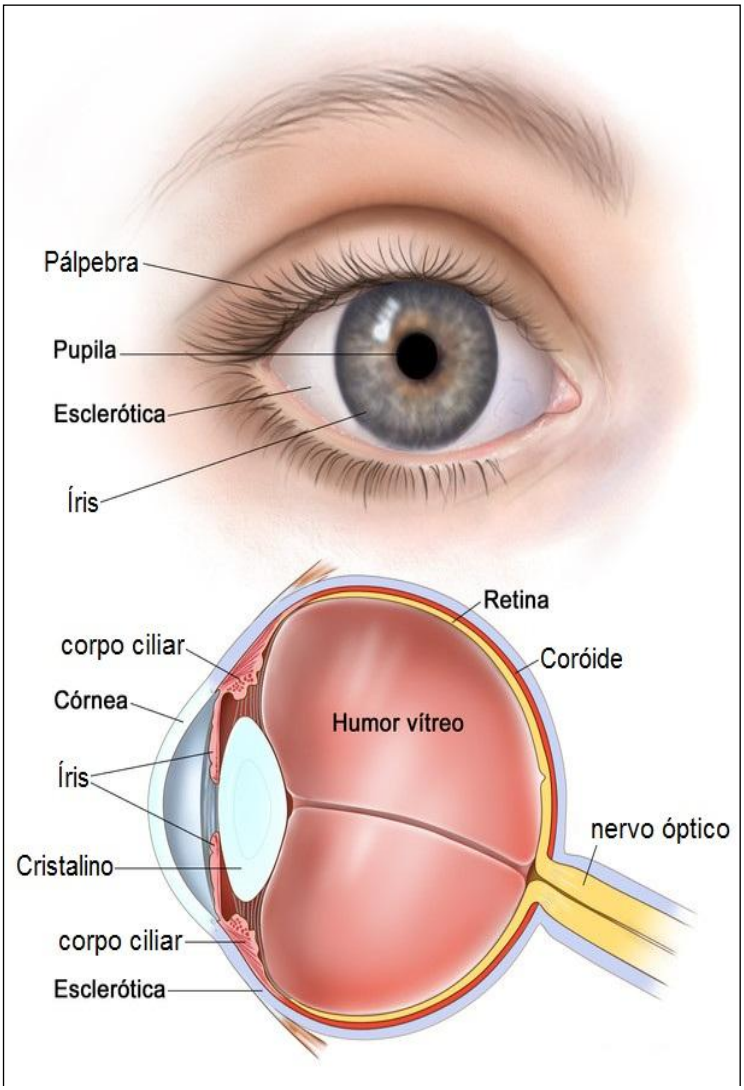
Cristalino: responsável pelo ajuste do foco dos objetos de acordo com a distância em que estes se encontram. Esse ajuste da imagem sobre a retina é denominado acomodação visual. A perda de sua transparência é chamada de catarata e pode levar à cegueira.

Retina: é um tecido que funciona como o “filme” numa máquina fotográfica, recebendo as imagens, formando-as e enviando-as para o cérebro.

Nervo óptico: é a continuação das células nervosas da retina e sua função é transmitir as imagens da retina para o cérebro para formar a visão.

Filme lacrimal: também conhecido como “lágrima”, é o mecanismo de proteção local contra infecções, através da limpeza, umidificação e lubrificação da superfície ocular.

FIGURA 5 – Anatomia do olho humano

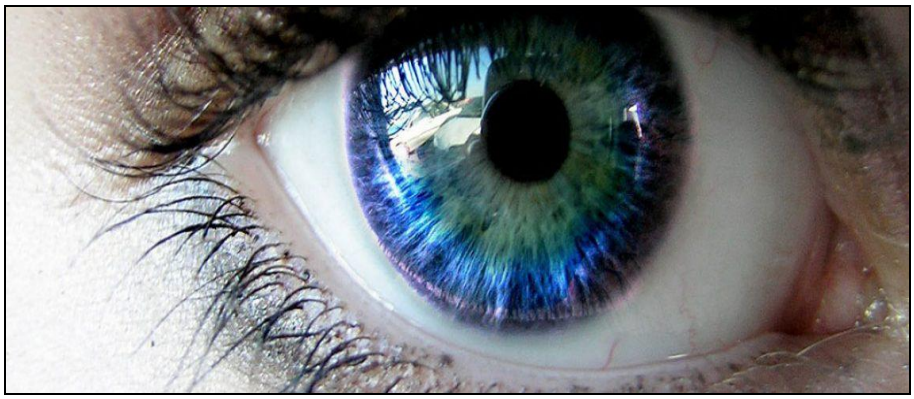


FONTE: https://hob.med.br/como-funciona-o-olho-humano (2019).

Outra estrutura ocular é a íris, conforme ilustrada na figura 6. Ela é a “parte colorida” do olho e essa cor difere de pessoa para pessoa, conforme a quantidade de melanina (VANPUTTE et al, 2016).

Muitos genes afetam a coloração, o que explica a complexidade das cores e os padrões de herança genética. É uma estrutura contrátil que controla a pupila, abertura por onde entra a luz no olho, através de dois grupos de músculos lisos: um grupo circular chamado esfíncter da pupila e radial, chamado dilatador da pupila (VANPUTTE et al, 2016).

FIGURA 6 – Detalhe da Íris humana



FONTE: www.oftalmologistabh.com.br (2019).

Cada pessoa possui uma íris única, ainda que se trate de gêmeos univitelinos. Essa diferença se observa, inclusive, entre, a íris direita e à esquerda de uma mesma pessoa (WILDES, 1997).

2.9.2 Breve histórico do reconhecimento de íris

Em 1885, o físico francês Alphonse Bertillon propôs a ideia de usar a íris como um identificador humano, descrevendo a cor e o tipo de padrão (JAIN et al, 2008).

Em 1936 o oftalmologista inglês Frank Burch propôs a ideia de usar os padrões da íris como método de reconhecimento individual (SANTHI, 2016).

Em 1949, James Doggart, oftalmologista inglês, discursou sobre a complexidade dos padrões da íris e da possibilidade desta ser suficientemente única, como ocorre com as impressões digitais. Em 1987, os oftalmologistas Leonard Flom e Aran Safir patentearam o “Conceito de Doggart”, embora não tivessem algoritmo ou método específico para tornar o reconhecimento possível (JAIN et al, 2008).

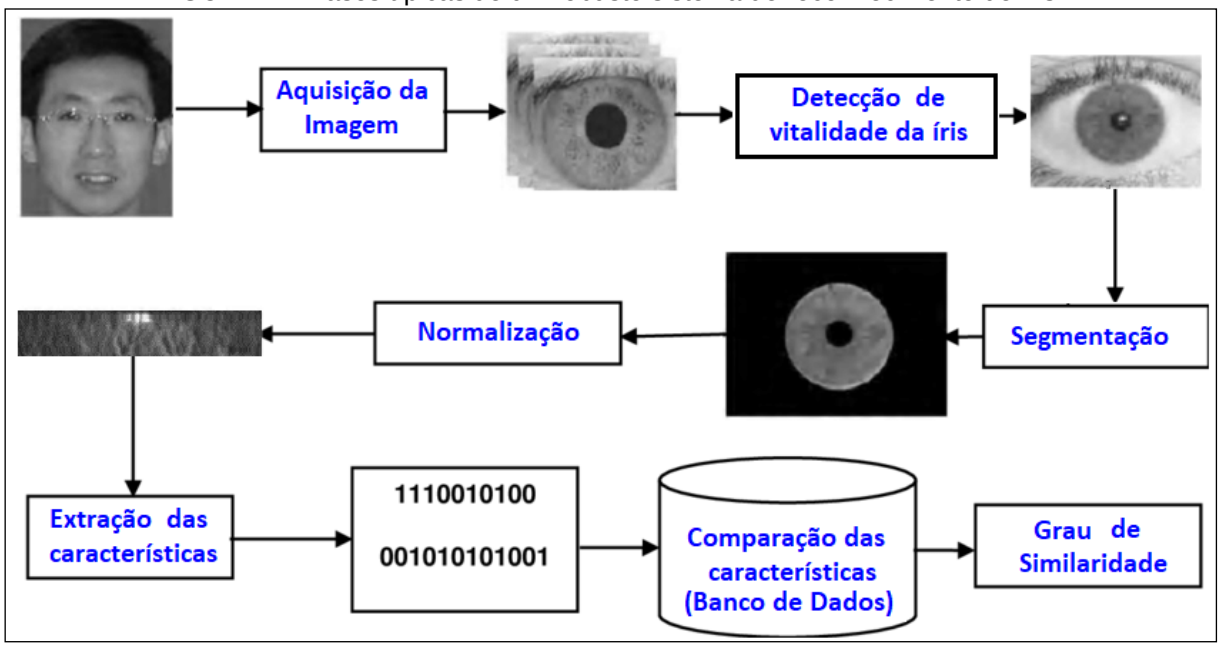
Em 1989, Flom e Safir procuraram o matemático Dr. John Gustav Daugman, para auxiliar na criação de algoritmos que possibilitassem digitalizar imagens da íris e, assim, permitir sua comparação com imagens em tempo real.

Em 1993, Daugman (2004) patenteou um algoritmo computacional para a realização do reconhecimento de um indivíduo através das características da textura da íris, método é o mais conceituado e, também, o mais utilizado nos sistemas atuais de reconhecimento de íris.

2.9.3 Fases de um sistema de reconhecimento da íris

Demonstra-se na figura 7, o diagrama de blocos de um sistema de reconhecimento de íris.

FIGURA 7 – Fases típicas de um robusto sistema de reconhecimento de íris



FONTE: Adaptado de Bodade e Talbar (2014, p. 8).

Bodade e Talbar (2014), esse sistema de reconhecimento possui cinco módulos, conforme a seguinte descrição:

- **Aquisição da imagem**: etapa correspondente à captura da imagem da íris, através de um sensor específico. A qualidade para aplicação prática é um desafio, pois a íris é pequena quando comparada ao tamanho do rosto, é cercada por pálpebras e tem sua textura alterada sob iluminação artificial.
- **Detecção de vitalidade da íris:** ação que visa assegurar que a imagem advém de uma pessoa viva e não de uma fotografia, uma reprodução de vídeo ou uma prótese de vidro, etc.
- **Segmentação da íris:** essa etapa corresponde à localização e segmentação da íris, pupila e pálpebras. O anel da íris (região de interesse) deve ser extraído da imagem capturada.

- **Normalização da íris:** o anel da íris extraído contém inconsistências, como por exemplo, as variações de distância da imagem, dilatações da pupila causadas por variações de luminosidade, inclinação da cabeça, rotação dos olhos etc. Para compensar essas inconsistências o sistema é transformado de Cartesiano para um sistema de coordenadas polares.
- **Extração de características:** é necessário extrair um subconjunto de pixels da imagem, onde características morfológicas são traduzidas para um padrão matemático, que possa ser utilizado por um sistema de comparação.

No processo de segmentação, na localização das bordas da pupila e íris, Daugman propôs a utilização de um operador que assume a aproximação das bordas sejam aproximadas por dois círculos, não necessariamente concêntricos, tendo assim, o comportamento de um detector de bordas circulares (DAUGMAN, 1993).

O Operador Integro-diferencial de Daugman, como é conhecido, é indicado pela equação 1:

$$G(x, y, f, \theta, \sigma) = e^{[-\frac{1}{2}(\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2})]}.e^{[2.\pi.j.f(x_\theta)]} \quad (1)$$

Pelo Método de Daugman a normalização proposta transforma o anel correspondente à íris da imagem de entrada, em um retângulo de dimensões fixas. A imagem original $I_{(x,y)}$ em coordenadas cartesianas é representada agora em um sistema de coordenadas polares na forma $I_{(r,\theta)}$, cuja origem está no centro da íris. Esta transformação geométrica é conhecida por *Rubber Sheet Model*, e é descrita pelas seguintes equações:

$$I(x(r,\theta), y(r,\theta) \rightarrow I(r,\theta) \quad (2)$$

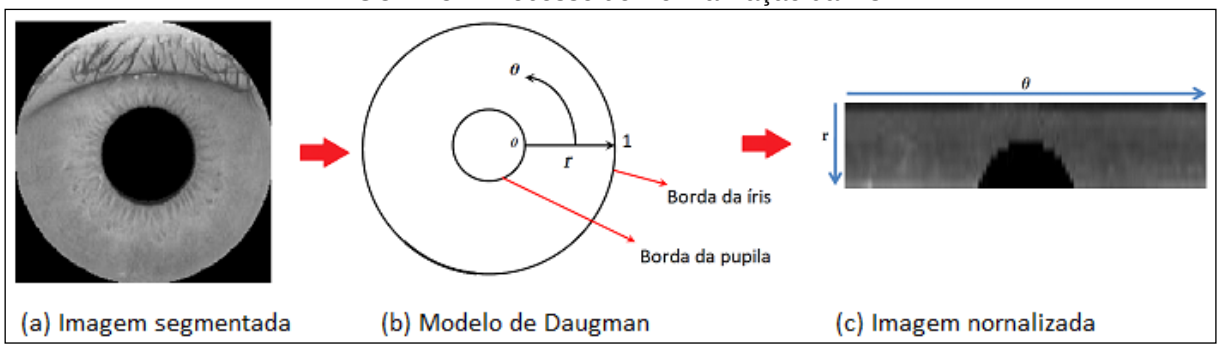
$$x(r,\theta) = (1-r)x_p(\theta) + rx_s(\theta) \quad (3)$$

$$I(x(r,\theta), y(r,\theta) \rightarrow I(r,\theta) \quad (4)$$

Onde $x_p(\theta)$, $y_p(\theta)$, $x_s(\theta)$ e $y_s(\theta)$ são, respectivamente, as coordenadas cartesianas do contorno da pupila e da esclera correspondentes ao ângulo θ. Nestas

equações “r” está contido no intervalo [0,1] e “θ” é um ângulo entre [0,2π]. Um exemplo de normalização de íris é mostrada na figura 8.

FIGURA 8 – Processo de Normalização da íris



FONTE: Os Autores (2019).

No intuito de revelar detalhes que não podem ser vistos apenas com luz visível, Daugman (2004) sugeriu uma captura de imagem com iluminação infravermelha, com um comprimento de onda na faixa de 700 a 900 nm, valor este considerado seguro pela Academia Americana de Oftalmologia.

2.9.4 Banco de dados de imagens de íris

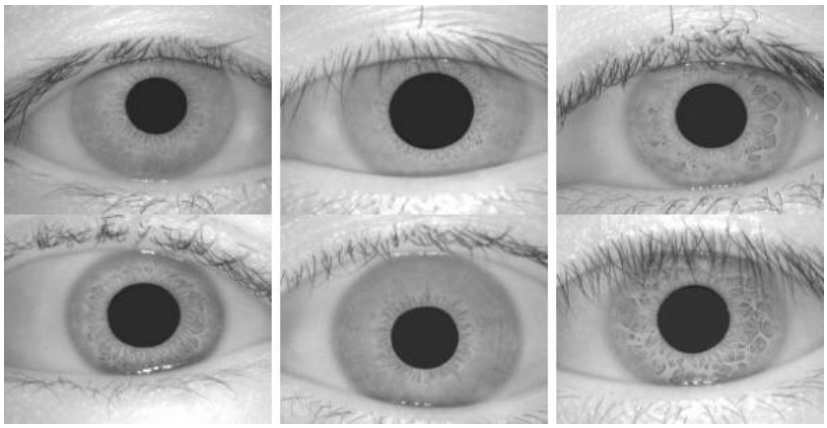
No início das pesquisas de reconhecimento de íris, antes dos anos 2000, os dispositivos de aquisição dessas imagens eram desenvolvidos para uso comercial e não permitiam seu armazenamento (PETROVSKA-DELACRÉTAZ et al, 2009).

Nesse sentido, conforme Petrovska-Delacrétaz, Chollet e Dorizzi (2009, p.30), não havia um banco de imagens de íris disponível, traduzindo-se no principal óbice para a pesquisa e desenvolvimento sobre o tema. Essa situação evoluiu e alguns bancos de dados de imagens de íris foram disponibilizados ao público, entre os quais se destacam:

- CASIA – *Chinese Academy of Sciences, Institute of Automation*: primeiro instituto a compartilhar imagens de íris gratuitamente com pesquisadores. Possui 756 imagens de íris de 108 olhos, com resolução de 320 x 280 pixels, na sua Versão 1, capturadas por três diferentes sensores e a maioria das pessoas são chinesas. Um exemplo ilustrado na figura 9.

- UPOL – inclui 384 imagens de íris com resolução de 768 x 576 pixels, com boa qualidade e sem oclusões de pálpebras/cílios, de 68 pessoas europeias.
- UBATH – desenvolvido pela Universidade de Bath, com captura em alta resolução possui 2000 imagens de íris de 50 pessoas com 1280 x 960 pixels.
- UBIRIS – foi desenvolvido para teste de algoritmos de reconhecimento de íris com diferentes tipos de variações ou degradação da imagem (iluminação, reflexão, contraste, desfocamento e oclusão). Na sua Versão 1 possui 1877 imagens em tom cinza com 430 x 300 pixels, de 241 pessoas. Exemplo ilustrado na figura 10.
- ICE’2005 – disponibilizado pela National Institute of Standards and Technology (NIST) com 2953 imagens de 132 pessoas.

FIGURA 9 – Banco de dados de íris CASIA



FONTE: https://www.researchgate.net/figure/Figura-1-Amostras-de-imagens-de-iris-CASIA (2019).

Os bancos de dados de imagem de íris podem ser divididos em duas categorias: banco de dados de boa qualidade de imagem (CASIA, IITD, ICE, MMU e UPOL) e banco de dados com imagens de íris “ruidosa”, ou seja, com alguma distorção, como por exemplo, a UBIRIS (ZAIM et al, 2011, p. 705).

2.10 USO DO PROCESSAMENTO DE IMAGEM NO RECONHECIMENTO DA ÍRIS

O reconhecimento da íris tem sua base fundamentada nas técnicas de processamento de imagem.

FIGURA 10 – Banco de dados de íris UBIRIS

FONTE: http://iris.di.ubi.pt/ (2019).

De modo geral o processamento de imagem objetiva facilitar tanto a visualização da imagem ou sua adequação, através de correções de defeitos ou realce das regiões de interesse para análise quantitativa, quanto à extração e tratamento dos dados, realizados com o uso da computação (GOMES, 2001).

2.10.1 Representação de imagens digitais

Uma imagem é uma função bidimensional de intensidade da luz $f(x,y)$, onde “x” e “y” denotam as coordenadas espaciais, em qualquer ponto (x, y), é proporcional ao brilho (nível de cinza) da imagem naquele ponto. Convenciona-se atribuir, proporcionalmente, valores mais altos para áreas de maior brilho (CARVALHO, 2004).

Uma imagem digital pode ser considerada uma matriz, cujos índices de linhas e colunas identificam um ponto na imagem, e seus elementos são chamados de pixels ou pels (*pictures elements*), como exemplo mostrado na figura 11.

Sendo o pixel o elemento básico em uma imagem, sua forma mais comum é retangular ou quadrada. Ele possui dimensões finitas na representação de uma imagem digital (ESQUEF et al, 2003).

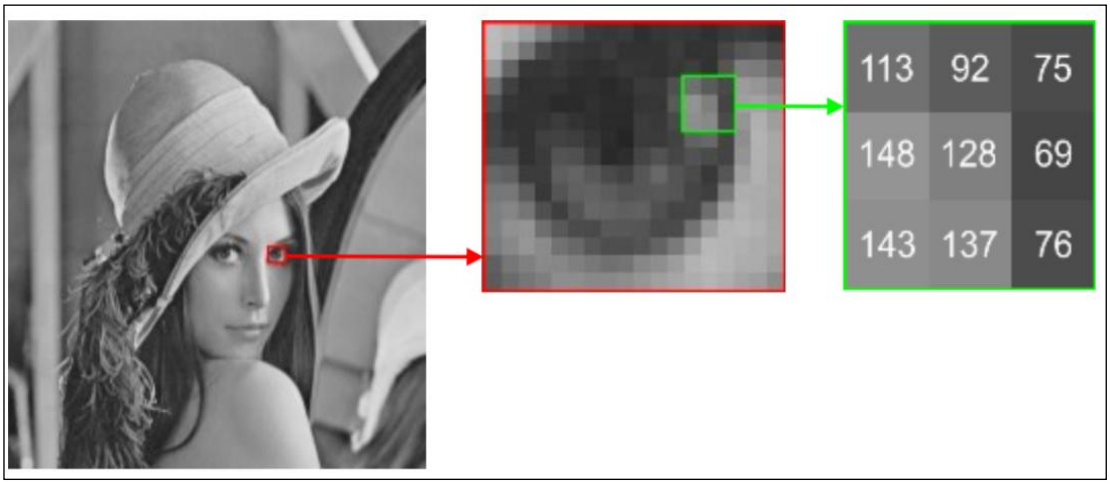
FIGURA 11 – Imagem digital de “Lena”

FONTE: Adaptado de http://ndevilla.free.fr/lena/ (2019).

Para gerar uma imagem digital, $f(x,y)$ deve ser digitalizada espacialmente (amostragem) e em amplitude (quantização em níveis de cinza). A amostragem (normalmente uniforme) de $f(x,y)$ nas direções "*x*" e "*y*" resulta em uma matriz de *NxM* amostras, seguidas de uma quantização do valor de $f(x,y)$ em "L" níveis inteiros de cinza $L \in [0;255]$.

De uma forma muito simplificada pode-se dizer que a imagem digital é representada como um "mundo de quadradinhos" (GOMES, 2018), conforme ilustrado na figura 12.

FIGURA 12 – Concepção da imagem digital exemplificada pela Imagem de "Lena"



FONTE: GOMES (2018).

A imagem monocromática de “Goldhill”, representado na figura 13, apresenta uma região em destaque em que se podem observar os pixels e os níveis de cinza, ou níveis de luminância, de cada um deles.

FIGURA 13 – Imagem monocromática “Goldhill” com destaque para uma região de 17x17 pixels

FONTE: http://www.cbpf.br/cat/pdsi/pdf/cap3webfinal.pdf (2019).

Para uma mesma região do espaço, quanto maior o número de pixels (menor será o tamanho do pixel) e, consequentemente, maior será a resolução espacial da imagem. Considerando um mesmo intervalo em *f(x; y; z),* quanto maior for a profundidade b (menor será a distância entre os níveis de quantização), maior será a resolução radiométrica de uma imagem cinza (MARTINS, 2019);

Segundo Martins (2019), se uma imagem de 256x256 pixels e 256 níveis de cinza tiver uma redução espacial, e.g. para 128x128, 64x64..., mantendo a escala de

cinza, os pixels das imagens de mais baixa resolução terão que ser replicados, caso a imagem permaneça com a mesma área original, conforme mostrado na figura 14.

FIGURA 14 – Imagens da “Lena” com 256 níveis de cinza

(a) 256x256 pixels (b) 128x128 pixels (c) 64x64 pixels

FONTE: Adaptado de https://www.ic.unicamp.br (2019).

Agora mantendo a resolução espacial e alterarmos os níveis de cinza, e.g. de 256 para 16, 8..., teremos regiões com cores homogêneas, conforme pode ser verificado na figura 15.

FIGURA 15 – Imagens de “Lena” com resolução de 256x256 pixels

(a) 16 níveis de cinza (b) 8 níveis de cinza (c) 2 níveis (binária)

FONTE: Adaptado de https://www.ic.unicamp.br (2019).

2.10.2 Processamento e análise digital de imagem

Conforme Gomes (2001) o processamento de imagem pode ser entendido como o conjunto de duas técnicas:

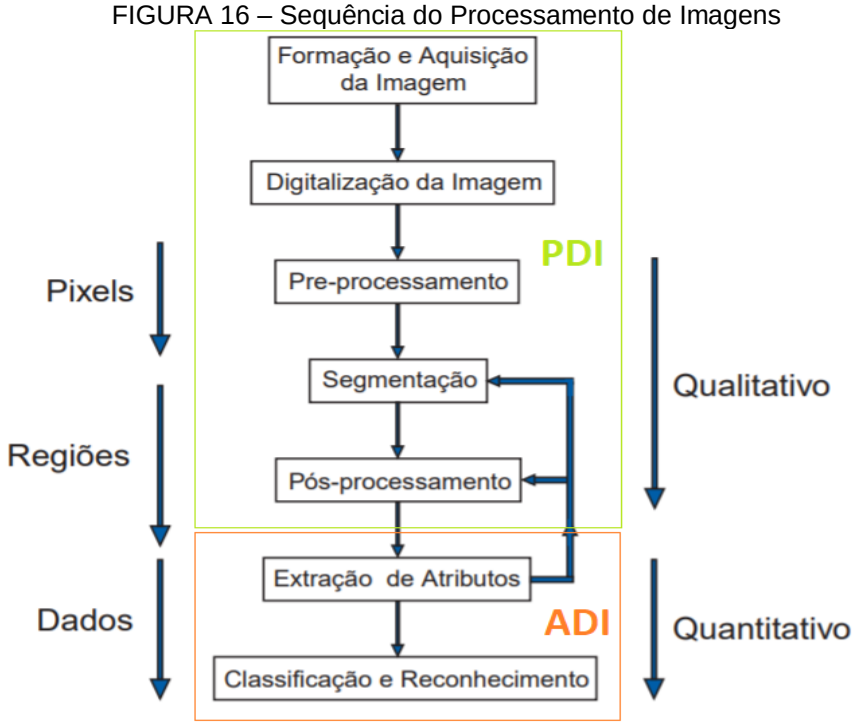
- **Processamento Digital de Imagens (PDI):** consiste no preparo da imagem para posterior análise com utilização de operações matemáticas.
- **Análise Digital de Imagens (ADI):** compreende a análise quantitativa do processo a partir da qual as regiões, partículas e objetos identificados na imagem são medidos.

Nesse dicotomia, o Processamento e Análise Digital de Imagens (PADI) abarca ambas as técnicas e abrange uma ampla escala de *hardware*, *software* e fundamentos teóricos.

O sistema de processamento de imagens é constituído de diversas etapas (ESQUEF et al, 2003), como ilustrado no fluxograma da figura 16:

No fluxograma apresentado na figura 16, as setas indicam os níveis com que se trabalha em cada etapa. Até o processo de segmentação, as operações são feitas no nível dos pixels das imagens, resultando em imagens com objetos ou regiões de pixels contíguos de valores iguais.

FIGURA 16 – Sequência do Processamento de Imagens



FONTE: Adaptado de ESQUEF et al (2003).

As medidas para a geração de dados são feitas na extração de atributos e são utilizadas no fim do processo, na etapa de reconhecimento de padrões e classificação, de forma a obter dados mais complexos. Até o pós-processamento a análise é considerada qualitativa e a partir da extração de atributos, quantitativa (Gomes, 2001).

Destacam-se as seguintes etapas no processamento de imagem:

- Formação e Aquisição da Imagem

Nesta fase o objetivo é a obtenção de uma imagem digital e, para que ela seja adquirida, são necessários:

- um dispositivo físico sensível ao espectro de energia eletromagnético, como, por exemplo, ao espectro de raio-x, luz ultravioleta, visível ou infravermelha. Este dispositivo transdutor deve produzir em sua saída um sinal elétrico proporcional ao nível de energia percebido. Como exemplos destes componentes estão filmadoras, scanners, câmeras fotográficas etc.

As câmeras são usadas para capturar dados de imagem de um objeto ou local examinado para várias aplicações de imagem, como visão de máquina ou automação de fábrica. São componentes de imagem que fazem interface com as lentes de imagem. As câmeras possuem sensores projetados para gerar imagem com foco na luz. Segundo OPTICS (2019), as câmeras são oferecidas em diversos formatos de sensor, CCD ou CMOS, adequados para praticamente qualquer aplicação de imagem de câmera.

Segundo HEDGECOE (2005), uma câmera é composta pelos seguintes componentes:

- fonte de luz: os raios de luz são refletidos a partir do objeto e são transmitidos através da câmera para formar uma imagem latente no sensor de imagem;
- lentes: uma lente consiste em um disco convexo de vidro polido que refrata os raios de luz que incidem para todos os pontos do objeto, para que eles convirjam para formar pontos coerentes. O ponto em que a lente focaliza esses raios – o plano focal – coincide com a posição do sensor de imagem quando a lente está corretamente focada;
- plano de foco: os raios de luz são refratados pela lente que convergem para formar uma imagem de cabeça para baixo. A luz viajando em diferentes distâncias da câmera precisa de vários graus de refração para se concentrar no plano focal, então um foco mecanismo move a lente em direção ou distanciando parte traseira da câmera. A posição do sensor de imagem e plano focal coincidem se a lente estiver corretamente focada;
- abertura: o diâmetro do diafragma da lente pode ser mudado girando o anel de abertura. Isso determina o brilho da imagem atingindo o filme. Mudar o *f-number* altera o tamanho da abertura. Um exemplo de diafragma é mostrado na figura 17.

- obturador: o obturador pode ser ajustado em velocidades diferentes, o que determina o comprimento de tempo que o filme é exposto. Movendo a velocidade do obturador irá aumentar ou reduzir o tempo de exposição (HEDGECOE, 2005). A figura 18 mostra um exemplo de obturador.

FIGURA 17 – Abertura do diafragma da lente

(a) Abertura total (b) Abertura Média (c) Abertura mínima

FONTE: Adaptado de HEDGECOE (2019).

FIGURA 18 – Exemplo de obturador

FONTE: HEDGECOE (2019).

Quando se captura uma foto com uma câmera digital, um obturador é aberto para permitir que a luz do objeto atinja o sensor de imagem.

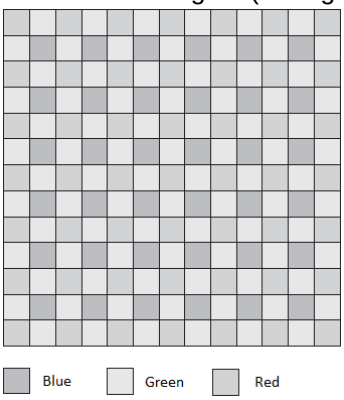
Como quase todas as câmeras digitais usam cores em fotografias, o dispositivo, comumente, utiliza filtros de cores sobre cada um dos fotodiodos, como mostra a figura 20. Cada um dos pixels mede a intensidade da luz que o atinge. A intensidade, medida na quantidade de eletricidade que a luz gera, é convertida em valores digitais (WHITE, 2003).

A figura 19 mostra os componentes que formam um dispositivo para captura de fotos.

FIGURA 19 – Componentes de um dispositivo de captura

FONTE: Adaptado de HEDGECOE (2019).

FIGURA 20 – Sensor de imagem (retângulo de pixels)



FONTE: Adaptado de WHITE (2019).

- Digitalização da Imagem

A conversão do sinal elétrico, produzido na saída do dispositivo transdutor, em um sinal digital, é realizada através de um conversor Analógico-Digital (AD).

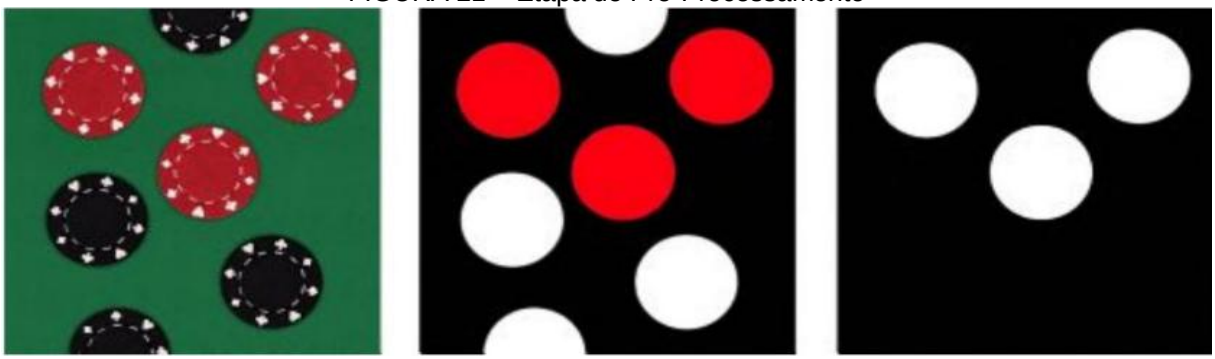
- Pré-processamento

Uma vez definidos os objetos de interesse da imagem digitalizada, as técnicas de pré-processamento envolvem a detecção de bordas e formas geométricas e tratamento de ruídos para realçá-la, facilitando a obtenção de suas informações por um sistema de Visão Computacional (BARELLI, 2018).

- Segmentação

Os objetos de interesse são segmentados, ou seja, são destacados da imagem original, formando uma imagem independente, facilitando a extração de características. Um exemplo de segmentação pode ser visto na figura 21.

FIGURA 21 – Etapa de Pré-Processamento



(a) Imagem original (b) Após Pré-processamento (c) Após segmentação

FONTE: Adaptado de BARELLI (2018).

- Extração de atributos

Nesta etapa são extraídas as características desejadas e codificadas na forma de um vetor de características (vetor de atributos ou assinatura). Este passo deve representar/identificar a imagem da qual a característica foi extraída, da forma mais fiel possível (BACKES, 2016).

- Classificação e reconhecimento

Nesta fase as imagens são classificadas/agrupadas conforme seus conjuntos de características extraídas.

2.10.2.1 Filtragem de sinal no PDI

O uso de filtros no processamento de imagens, de forma geral, traduz-se na aplicação de técnicas de transformação com o intuito de melhorar a qualidade do que é observado com, por exemplo, a redução de ruídos e o realce, suavização ou correção.

A filtragem pode estar no **Domínio Espacial**, em que matrizes (máscaras) percorrem a imagem original alterando o valor dos pixels ou no **Domínio da Frequência**, onde os procedimentos são operados sobre a Transformada de Fourier (TF) da imagem original.

Conforme Barelli (2018), os filtros no Domínio Espacial podem ser classificados de dois tipos, de acordo com o tipo de operação entre a máscara e o pixel-alvo da imagem:

- Filtros lineares

São mais comuns e usam máscaras que fazem somas ponderadas da intensidade dos pixels ao longo de toda a imagem. Eles suavizam ou realçam detalhes da imagem e minimizam efeitos de ruído, sem alterar o nível médio de cinza da imagem.

- Filtros não lineares

Realizam somas não ponderadas, ou seja, na prática realizam as transformações que podem alterar o nível médio de cinza da imagem original, destacando linhas e bordas, por exemplo.

Conforme a faixa de frequência filtrada, alguns filtros classificam-se em:

Passa-Baixa (PB): atenua altas frequências suavizando regiões que representam bordas ou contornos;

Passa-Alta (PA): atenua as baixas frequências, realçando as regiões de bordas e contornos; e

Passa-Faixa (PF): permitirá apenas a passagem da banda selecionada.

Nesse sentido, tendo em vista o melhoramento dos sinais no processamento da imagem, listam-se a seguir alguns detectores e métodos de filtragem julgados importantes no desenvolvimento deste projeto:

- Filtro de Gabor

Desenvolvido por Dennis Gabor em 1946, o método descreve matematicamente a análise espectral e temporal do sinal, indo de encontro às formulações de Fourier, vez que estas são realizadas simultaneamente. O tempo e a frequência formam eixos ortogonais e os sinais são representados em um diagrama bidimensional.

É um filtro linear bidimensional e não variante ao deslocamento podendo ser entendido como o produto de uma função gaussiana, simétrica à origem e uma função cossenoidal de frequência "*f* ", conforme a Equação (5).

$$G(x, y, f, \theta, \sigma) = e^{[-\frac{1}{2}(\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2})]}.e^{[2.\pi.j.f(x_\theta)]} \quad (5)$$

Onde:

$$x_\theta = x.\cos\theta + y.sen\theta \quad (6)$$

$$y_\theta = -x.sen\theta + y.\cos\theta \quad 7)$$

Onde x, y são as coordenadas espaciais da imagem e j=número complexo

Com os seguintes parâmetros:

1) *f*: frequência da onda no plano senoidal;

2) θ_k: é a orientação do filtro;

3) σ_x e σ_y: o desvio padrão da função gaussiana ao longo dos eixos x e y, respectivamente.

A G_{real} pode ser representada no domínio da frequência e, sua TF pode ser obtida pela convolução da TF dessas duas funções.

O resultado é um filtro Passa Faixa, que realça as senóides com frequências em torno de "*f*", reduzindo seus ruídos.

Quando nos domínios do espaço (x, y) e da freqüência, o Filtro Gabor 2D permite análises em diferentes frequências, orientações e escalas e as funções são superfícies gaussianas moduladas por exponenciais complexas.

A técnica tem uma importante aplicação na segmentação de imagens e no reconhecimento facial devido às suas características, com destaque nas representações de frequência e orientação, similares ao sistema humano de visão (LEE et WANG, 1999).

- Filtro de Sobel

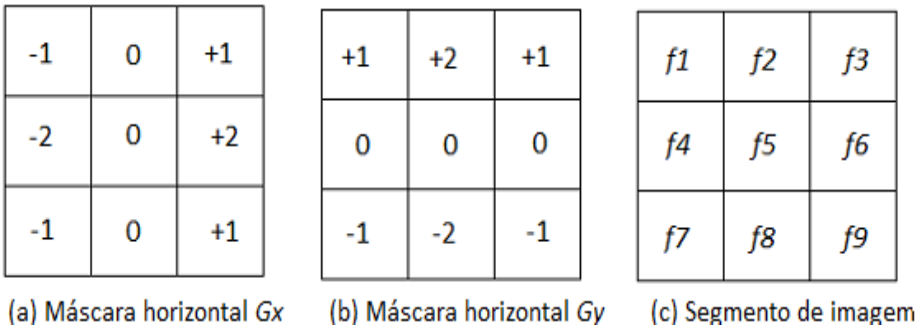
Filtro espacial não linear que realça as linhas verticais e horizontais mais escuras que o fundo, resultando em bordas mais espessas quando comparadas a

outros métodos, mas é a detecção de bordas, em imagens, a principal utilização deste filtro, também conhecido como Operador de Sobel (BARELLI, 2018).

Os objetos presentes em uma imagem são caracterizados por propriedades como cor, textura e forma. As variações entre suas regiões adjacentes definem a percepção sobre o formato de um objeto e determinam sua fronteira com outros elementos da própria imagem, bem como, com outros objetos e o fundo.

Sobel é utilizado tipicamente para encontrar a magnitude aproximada do gradiente de cada ponto numa imagem em tons de cinza. O Filtro de Sobel consiste em um par de matrizes de convolução 3x3 (*kernel*) como mostrada na figura 22. Um *kernel* é simplesmente o outro rotacionado em 90º (QIDWAI et CHEN, 2009).

FIGURA 22 – *Kernels* de Convolução Sobel



FONTE: Adaptado de QIDWAI et CHEN(2009).

Considerando um segmento de imagem do mesmo tamanho que o *Kernel* de Sobel, conforme mostrado na figura 23, o resultado da convolução será conforme a Equação (8):

$$|G| = |(f_1 + 2f_2 + f_3) - (f_7 + 2f_8 + f_9| + |f_1 + 2f_4 + f_7) - (f_3 + 2f_6 + f_9| \quad (8)$$

A magnitude do gradiente é dada pela Equação (9):

$$|G| = \sqrt{(G_x)^2 + (Gy)^2} \quad (9)$$

A magnitude pode ser aproximada através da Equação (10):

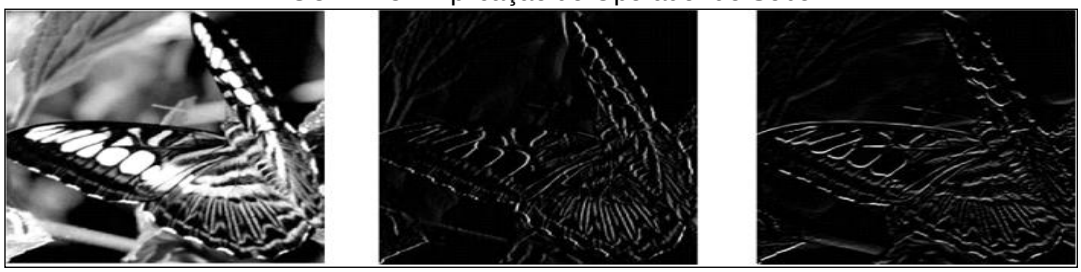
$$|G| = |G_x| + |Gy| \quad (10)$$

O ângulo de orientação da borda, em relação à grade de pixels, que origina o gradiente espacial é indicado pela Equação (11):

$$\theta = \tan^{-1}(\frac{Gy}{G_x}) \quad (11)$$

Um exemplo da aplicação do Filtro de Sobel pode ser visto na figura 23:

FIGURA 23 – Aplicação do Operador de Sobel



(a) Imagem em escala de cinza (b) Máscara horizontal (Gx) (c) Máscara vertical (Gy)

FONTE: Adaptado de QIDWAI et CHEN(2009).

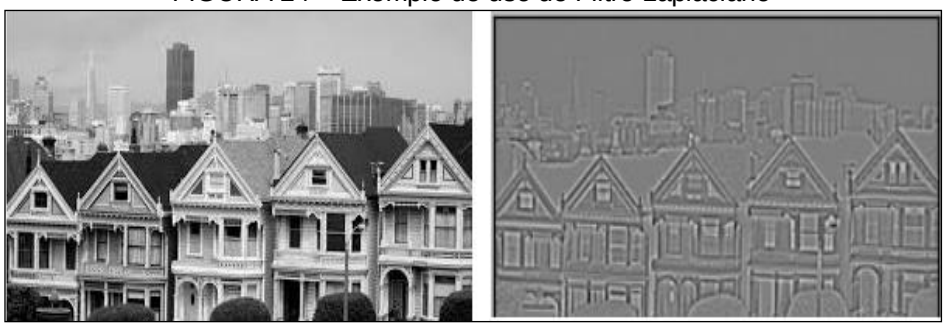
- Filtro Laplaciano

É um filtro espacial dos mais populares e, assim como o Filtro Sobel, é utilizado para realçar bordas. Possui uma máscara de 3ª ordem que percorre a imagem alterando o pixel-alvo através da média ponderada dos vizinhos, elevando o valor obtido ao quadrado. Embora produza bordas mais finas, superior ao desempenho do Filtro Sobel, é muito sensível a ruídos que comprometem a qualidade da imagem (BARELLI, 2018).

A solução para os problemas com ruídos está na instalação de um Filtro Passa-Baixas antes do Filtro Laplaciano como, por exemplo, um Filtro Gaussiano.

Um exemplo da aplicação do Filtro Laplaciano pode ser visto na figura 24:

FIGURA 24 – Exemplo de uso de Filtro Laplaciano



(a) Imagem original (b) Imagem com o Operador Laplaciano

FONTE: Adaptado de www.bogotobogo.com/python/OpenCV_Python/python_opencv3 (2019)

- Filtro de Canny

É um dos mais eficientes filtros para detectar bordas em imagens. Foi desenvolvido por John F. Canny (1986). É um filtro de convolução que usa a primeira derivada, suavizando o ruído e localizando as bordas, combinando um operador diferencial com um Filtro Gaussiano.

A aplicação da função Canny (OPENCV, 2019) para implementar o filtro de Canny segue as seguintes etapas:

- Filtram-se os ruídos usando o Filtro Gaussiano.
- Encontra-se a intensidade do gradiente da imagem, analogamente ao uso do Filtro Sobel, seguindo os procedimentos:
  - um par de máscaras de convolução são aplicadas (Máscara horizontal Gx e Máscara vertical Gy).
  - a magnitude e a direção do gradiente são encontradas através das Equações (10) e (11), respectivamente. A direção é arredondada para um dos quatro ângulos possíveis (0°, 45°, 90° ou 135°).
  - aplica-se um limitador superior a fim de remover os pixels que não são considerados parte de uma aresta, portanto, apenas linhas finas (arestas candidatas) permanecerão.
  - dois limites são usados (superior e inferior): se um gradiente de pixel for maior que o limite superior, o pixel será aceito como uma aresta; se estiver abaixo do limite inferior, ele será rejeitado e; se estiver entre os dois limites, ele será aceito apenas se estiver conectado a um pixel acima do limite superior.

Um exemplo da aplicação do Filtro de Canny é apresentado na figura 25:

FIGURA 25 – Exemplo de uso de Filtro de Canny

(a) Imagem original (b) Imagem com o Filtro de Canny

FONTE: Adaptado de https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html (2019).

2.10.2.2 Transformada de Hough

Foi desenvolvida por *Paul Hough* em 1962 e patenteada pela IBM. A Transformada de *Hough* tem o propósito de detectar linhas, círculos, elipses, entre outros objetos parametrizáveis, em imagens. Muitas figuras geométricas como linhas, círculos e elipses podem ser descritas usando simples equações com poucos parâmetros (BURGUER et BURGE, 2009).

Geralmente esta transformada é aplicada após a imagem sofrer um pré-processamento, como a detecção de bordas.

O princípio da Transformada de *Hough* é mapear um ponto da imagem em um plano de parâmetros, descrito como Espaço de *Hough*. O mapeamento era realizado, pelo método original de Hough, através da parametrização Inclinação-Intersecção (*slope-intercept*), com a equação da reta:

$$y = ax + b \tag{12}$$

Duda e Hart (1972) adaptaram a técnica para imagens digitais utilizando coordenadas polares para definir uma reta e, ao invés de trabalhar com a Inclinação-Intersecção, com o uso dos parâmetros Ângulo-Raio. O método proposto envolve o mapeamento de retas do espaço imagem, para conjuntos de pontos em um espaço de parâmetros Ângulo-Raio.

No plano da imagem, uma reta pode ser representada pela seguinte equação dada por:

$$\rho = x\cos(\theta) + ysen(\theta) \quad (13)$$

Onde:

ρ: distância da normal é a linha da origem da imagem

θ: ângulo da normal com o eixo horizontal

Nesse sentido, a detecção de retas consiste em localizar as células da matriz do Espaço de *Hough*, calculada pela Equação 13, que apresentem os maiores valores, ou seja, pontos que possuem máximos locais.

A HT para círculos usa a equação parametrizada do círculo, conforme a Equação 14:

$$r = \sqrt{(x - a^2) + (y - b^2)} \quad (14)$$

Onde

r: raio do círculo

x e y: são as coordenadas do centro do círculo

a e b: coordenadas do centro do círculo

2.10.2.3 Histograma

O histograma de uma imagem digital com níveis de cinza no intervalo [0, L-1] é uma função discreta $h(r_k) = n_k$, onde r_k é o k-ésimo valor de intensidade e n_k é o número de pixels da imagem com intensidade r_k (GONZALES e WOODS, 2002).

Os histogramas são a base para várias técnicas de processamento de domínio espacial. A manipulação de histogramas pode ser utilizada para o realce de imagens. Além de fornecer estatísticas úteis das imagens, as informações inerentes aos histogramas também são bastante úteis em outras aplicações de processamento de imagens, como compressão de imagem e segmentação (GONZALES e WOODS, 2002).

Como uma introdução ao processamento de histogramas para transformações de intensidade, considera-se a figura 26 mostrada em quatro características básicas em relação à intensidade de imagem: escura, clara, baixo contraste e alto contraste. O lado direito da figura mostra os histogramas

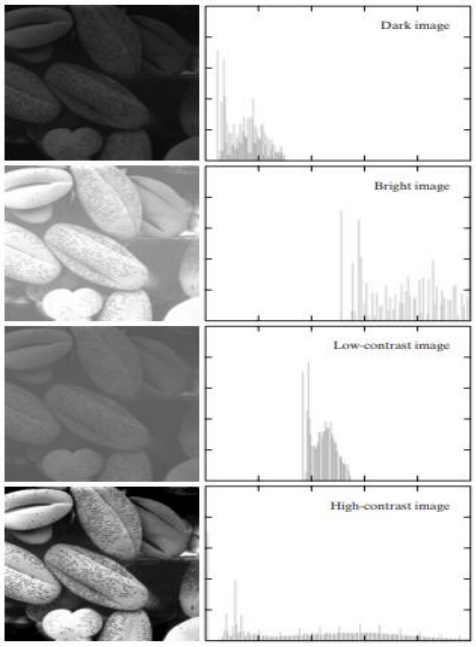
correspondentes a essas imagens. O eixo horizontal de cada gráfico de histograma corresponde a valores de intensidade, r_k. O eixo vertical corresponde a valores de $h(r_k) = n_k$ ou $p(r_k) = n_k / n$ se o valores são normalizados. Desta forma, os histogramas podem ser vistos como gráficos de $h(r_k) = n_k$ *versus* r_k ou $p(r_k) = n_k / MN$ *versus* r_k (GONZALES e WOODS, 2002).

Observe-se na imagem escura que os componentes do histograma estão concentrados no lado inferior (escuro) da escala de intensidades. De forma similar, os componentes do histograma da imagem clara tendem à direção do lado superior da escala. Uma imagem com baixo contraste tem um histograma estreito normalmente localizado no meio da escala de intensidades.

Para uma imagem monocromática, isso implica em uma aparência cinza, destacada e sem brilho. Finalmente, observa-se que os componentes do histograma na imagem de alto contraste cobrem uma faixa bem ampla da escala de intensidades e, também, que a distribuição de pixels não está muito longe de ser uniforme, com poucas linhas verticais sendo muito mais altas que as outras. Intuitivamente, é razoável concluir que uma imagem cujos pixels tendem a ocupar todo o intervalo de níveis possíveis de intensidade e, além disso, tendem a ser distribuídos uniformemente terá uma aparência de alto contraste e exibirá uma grande variedade de tons de cinza.

O resultado final será uma imagem que mostra boa correspondência em relação aos detalhes de nível de cinza e tem uma ampla faixa dinâmica (GONZALES e WOODS, 2002).

FIGURA 26 – Quatro tipos básicos de imagem: escura, clara, baixo contraste, alto contraste e seus histogramas correspondentes.



FONTE: GONZALES e WOODS (2002).

2.11 O APRENDIZADO DE MÁQUINA (MACHINE LEARNING)

Na sua essência, o *Machine Learning* desenvolve programas que aprendam a executar uma tarefa a partir da experiência, sem serem explicitamente programados para isso. O programa precisa ser capaz de entender o desempenho da tarefa para que possa ajustar-se e aprender com a “própria” experiência. Portanto, é crucial definir uma medida que identifique adequadamente o que significa "executar a tarefa" bem ou mal. Esse ciclo - aprender com os dados de treinamento, avaliar dados de validação e refinar, é o cerne de prever adequadamente o mundo, formulando hipóteses e conduzindo uma série de experimentos de validação para decidir como avançar (BRANDEWINDER, 2015).

Dessa forma, antes de aplicar um algoritmo, geralmente dividem-se os dados rotulados em grupos de treino e grupo de teste. Assim, treina-se o algoritmo com um grande volume de dados de treino, valida o resultado do algoritmo com os dados de teste e, a partir de então, coloca o algoritmo em campo para execução de dados reais.

O reconhecimento de imagens, e a escrita humana em particular, é um problema clássico no aprendizado de máquina. Esta é a razão pela qual os CAPTCHAs (*Completely Automated Public Turing test to tell Computers and Human Apart*) se apresentam de maneira simples e eficazes para descobrir se alguém é um

ser humano ou um robô, conforme exemplo da figura 27. O cérebro humano tem essa incrível capacidade de reconhecer letras e dígitos, mesmo quando estão muito distorcidos (BRANDEWINDER, 2015).

FIGURA 27 – Exemplo de CAPCHA

FONTE: PANDA et al (2019).

A técnica de aprendizado de máquina pertence ao subgrupo de inteligência artificial focado em desenvolver indivíduos capazes de extrair informações e se adaptar a mudanças, ou seja, aprender sobre a tarefa e/ou ambiente.

A área de *Machine Learning* é dividida em dois grupos de técnicas:

- Supervisionada: o algoritmo se baseia em um avaliador que indica se as respostas estão corretas;
- Não supervisionado: caso em que não há avaliador e cabe ao programa identificar padrões e verificar quais são as semelhanças entre os dados;

A aplicabilidade dos métodos se resume a tarefas não programáveis e de complexa descrição ou no tratamento de massiva quantidade de dados, como no caso de reconhecimento de padrões, agrupamento de dados e na regressão de valores (SHALEV-SCHWARTZ, 2014).

A figura 28 indica o fluxo de tarefas na aplicação de um algoritmo de *Machine Learning.*

O pré-processamento dos dados busca encontrar dados irregulares e discrepantes no conjunto de dados, também se preocupa em nivelar as diferentes *features* aplicando técnicas como a normalização e padronização, garantindo que as diferentes características não oscilem o seu grau de relevância de acordo com a amplitude dos dados.

Na etapa de treinamento o algoritmo é exposto aos dados de treino, e através do método indutivo, aprenderá a diferenciar os dados de acordo com os exemplos apresentados.

FIGURA 28: Fluxo de desenvolvimento do *Machine Learning*

FONTE: Os autores (2019).

Os conceitos utilizados em sistemas de aprendizado são compartilhados entre as diversas técnicas da área, como a utilização de pesos na ponderação das entradas, funções de custo para avaliação do desempenho do algoritmo e técnicas de aprendizado como o *back-propagation* aplicada em redes neurais ou o *gradient descent* comum em sistemas de regressão.

Diferentes métodos de treinamento podem ser utilizados, os mais comuns é a divisão dos dados em 3 grupos:

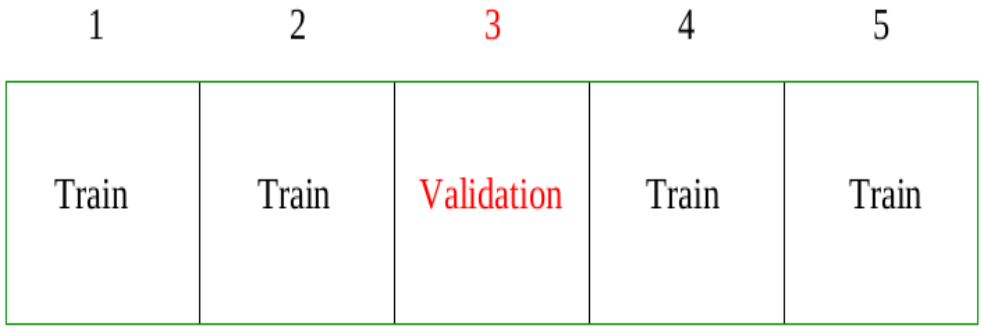
- Treinamento: Conjunto empregado no treinamento do algoritmo, normalmente representa 60% à 70% de todos os dados;
- Validação: Grupo utilizado no ajuste dos chamados hiperparâmetros, como no tipo de *kernel* aplicado em algoritmos do *Support Vector Machine* (SVM), ou o número de camadas da rede neural. Normalmente este grupo é utilizado na etapa de avaliação e comporta de 15% a 20% de todo os dados;
- Teste: Coleção de amostras que resume o desempenho do algoritmo, normalmente utilizado na etapa do fim do desenvolvimento, momento de apresentar os resultados a outros departamentos ou equipes.

O segundo método aplicado é chamado de *k-fold cross-validation*, em que os dados são divididos em n grupos, o treinamento e avaliação acontecem n vezes,

variando os grupos de cada etapa. O resultado para avaliação será a média das métricas utilizadas nos n ciclos do *k-fold.*

A figura 29 ilustra o método *k-fold*.

FIGURA 29 - k-fold *validation*



FONTE: HASTIE et al (2017).

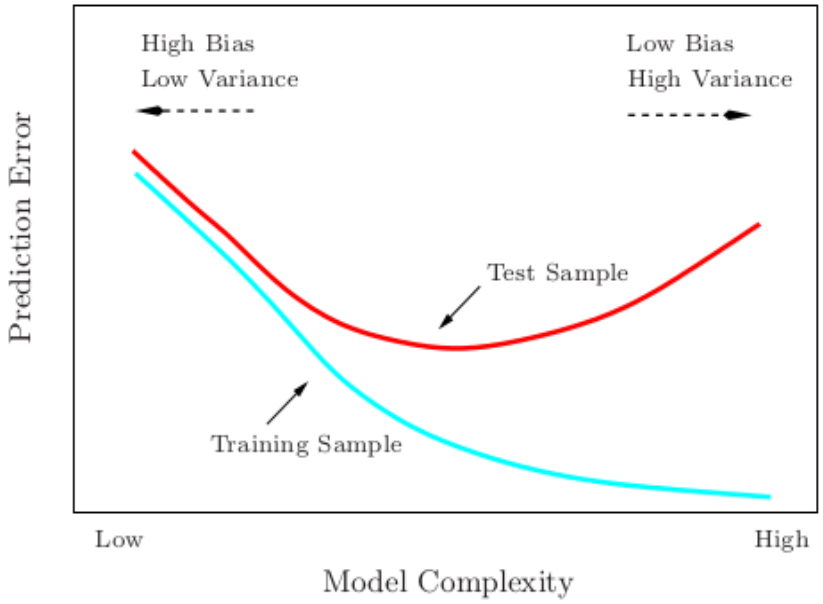
2.11.1 Métricas de avaliação

Além da importante etapa de treinamento do algoritmo, há também a sua avaliação, o que através deste procedimento é possível verificar e adequar o método para aumentar o desempenho na tarefa a qual está executando.

Há diferentes metodologias para avaliação de desempenho para algoritmos de regressão. É comum aplicar a análise das curvas de custo nas etapas de treinamento e validação a fim de evitar o ***overfitting***, momento em que o algoritmo decora o conjunto de treinamento e deixa de generalizar, e ***underfitting***, etapa em que o modelo é muito simples e não consegue aprender as características determinantes para representação dos dados.

As curvas das funções de custo do período de treinamento (tracejado azul) e do período de validação (traço em vermelho) são apresentadas na figura 30. Conforme a complexidade do modelo aumenta o custo da curva de treinamento somente diminui, porém, a partir de determinado ponto, na curva vermelha, passa a aumentar, mesmo com a alta complexidade do modelo. Este comportamento é um indicativo do efeito de *overfitting,* é recomendado a adoção do modelo que tem o menor custo, sem tal característica.

FIGURA 30 - Curva de custo do treinamento e validação



FONTE: HASTIE et al (2017).

Como algoritmos de classificação trabalham com a da determinação de classes e não de valores, as métricas de avaliação são diferentes. O procedimento padrão de análise se baseia na tabela de confusão, ilustrado pela tabela 2.

TABELA 2 - MATRIZ DE CONFUSÃO

		Classes verdadeiras	
		Positivo	Negativo
Classes previstas	Positivo	Verdadeiro positivo (TP)	Falso positivo (FP)
	Negativo	Falso negativo (FN)	Verdadeiro negativo (TN)

FONTE: Os Autores (2019).

As amostras classificadas como verdadeiras são aquelas que foram corretamente previstas pelo agrupadas pelo algoritmo, pertencentes à classe positiva ou negativa.

Os resultados presentes na coluna “Falso” indicam que houve erro na categorização.

Além da tabela de confusão, outras métricas também auxiliam na avaliação do algoritmo, dentre elas estão:

- *Precision*: Proporção de positivos corretamente classificados.
- *Recall:* Taxa de acerto da classificação positiva;

- *F1 score:*_Valor da média harmônica entre *recall* e *precision*, indica o equilíbrio entre os dois quesitos de avaliação.

A fórmula matemática das expressões é indicada pelas equações (15), (16) e (17):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{15}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{16}$$

$$\text{F1 score} = \frac{2.\text{precision.recall}}{\text{precision.recall}} \tag{17}$$

2.11.2 *Support Vector Machine* (SVM)

O SVM é um dos algoritmos de *Machine Learning* mais utilizados para classificação de pequenos conjuntos de dados, cujo objetivo é construir um hiperplano entre as amostras utilizadas para o treinamento da rede, maximizando a margem entre os vetores de suporte, dados relevantes para descobrir o melhor separador, e o hiperplano, encontrando o plano ótimo entre as amostras de diferentes classes (HAYKIN, 2008).

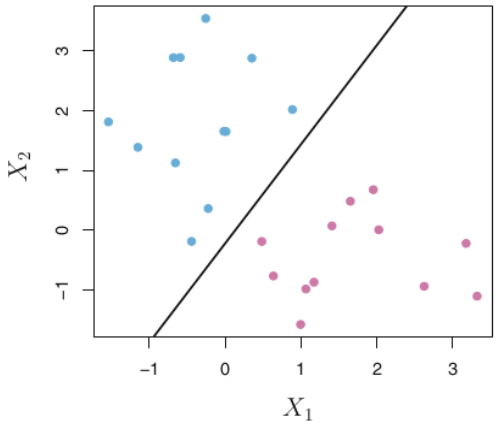
O hiperplano é descrito pela Equação (18):

$$W^T.X + b = 0 \tag{18}$$

O vetor “W” são os coeficientes do plano multidimensional, “X” são os valores de entrada e “b” é o *bias* do algoritmo.

A figura 31 ilustra o plano ótimo sobre um conjunto de dados bidimensional. A diferença na cor dos pontos indica que pertencem a diferentes classes.

FIGURA 31 - Hiperplano ótimo



FONTE: JAMES et al.(2013).

Ao contrário de redes neurais e cadeias de Markov, SVMs não são probabilísticas, ou seja, não classificam os dados de acordo com uma distribuição de probabilidade limitada por [0,1].

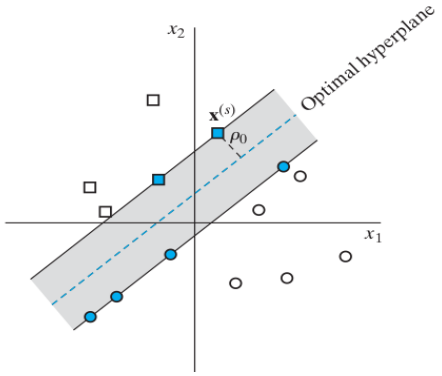
Em aplicações de duas classes os valores que as representam as classes são {1,-1}, conforme representado pelas equações (19) e (20) (HAYKIN, 2008).

$$W^T.X + b = 1 \tag{19}$$

$$W^T.X + b = -1 \tag{20}$$

A figura 32 mostra um exemplo de vetores de suporte e um hiperplano ótimo.

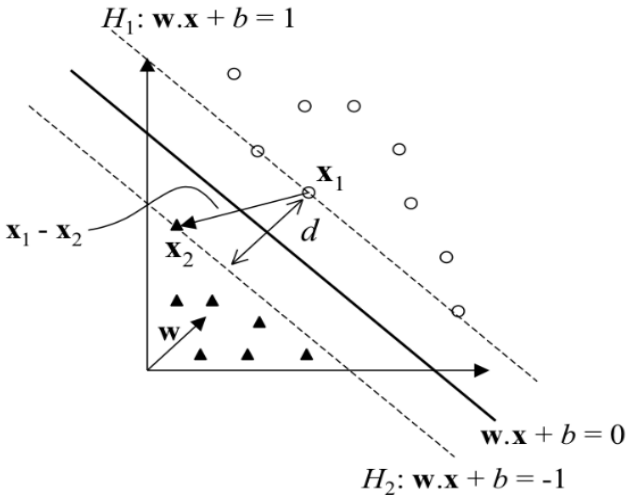
FIGURA 32: Vetores de suporte e hiperplano ótimo



FONTE: HAYKIN (2008).

Para o cálculo do hiperplano, a distância entre as margens dos planos que passam pelos vetores de suporte é utilizada, como a distância é 2, temos que o produto entre o vetor distância entre dois *support vectors* e o vetor normal do hiperplano deve ser dois, como indicado pela figura 33.

FIGURA 33 - Distância entre vetores de suporte



FONTE: CARVALHO e LORENA (2007).

Isolando o vetor X nas Equações (19) e (20), têm-se as Equações (21) e (22), a subtração resulta na equação (23).

$$X_1 = \frac{1-b}{W} \tag{21}$$

$$X_2 = \frac{-1-b}{W} \tag{22}$$

$$(X_1 - X_2).W = 2 \tag{23}$$

O produto escalar da distância entre X_1 e X_2 com o vetor normal do plano separador está representada pela Equação (24).

$$(X_1 - X_2).W = \frac{2.W}{W} \tag{24}$$

A equação 8 expande os termos da equação (25):

$$(X_1 - X_2).W = \frac{W.(X_1 - X_2)}{\|W\|\|X_1 - X_2\|} \tag{25}$$

Substituindo o numerador da equação (25) pela equação (23), tem-se:

$$\frac{2.(X_1 - X_2)}{\|W\|\|X_1 - X_2\|} \tag{26}$$

O módulo da equação (26) fica:

$$\frac{2}{\|W\|} \tag{27}$$

A maximização da equação (27) é a primeira restrição de otimização para encontrar o hiperplano ótimo, processo equivalente a minimização da equação (28) (JAMES et al., 2013).

$$\frac{1}{2}\|W\|^2 \tag{28}$$

A outra restrição é através da equação (29), generalização das equações 19 e 20, a variável “y” é o resultado de classificação do SVM:

$$y_I(W.X + b) - 1 \geq 0 \tag{29}$$

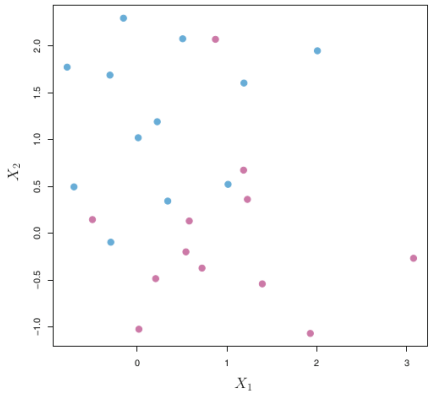
Aplicando a técnica de multiplicador de Lagrange na equação Lagrangiana, metodologia utilizada em problemas de otimização, na primeira restrição, representada pela equação (28), e na segunda restrição, pela (29), e, tem-se a equação (30):

$$L(a,W,b) = \frac{1}{2}\|W\|^2 - \sum a_i(y_i(W.X + b) - 1) \tag{30}$$

A resolução do multiplicador de Lagrange revela os valores de “W” e “b”. Tanto nas figuras 32 e 33 os casos são linearmente separáveis, o que permite a completa classificação dos exemplos, porém em casos não separáveis, como ilustrado pela figura 34, não há maneira de separar perfeitamente com um hiperplano linear, para este tipo de caso duas abordagens são comuns:

- *soft-margin;*
- *kernel trick*;

FIGURA 34 - Conjunto de dados não separáveis linearmente



FONTE: JAMES et al.(2013).

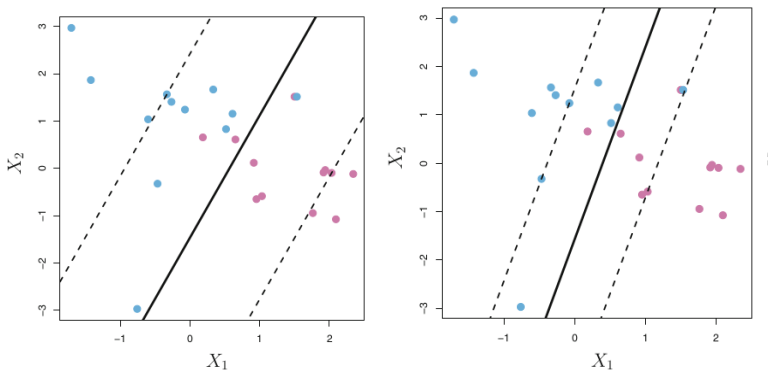
2.11.2.1 *Soft-Margin*

No *soft-margin SVM* , erros são permitidos a fim de aumentar o desempenho do algoritmo. Assim a equação 29, empregada no SVM de máxima margem, é alterada para o caso *soft-margin,* Equação 31, em que a margem de erro permitida é representada pela letra grega ε.

$$y_I (W.X + b) \geq 1 - \varepsilon \qquad (31)$$

Nas figura 35 as linhas tracejadas indicam as margens do algoritmo *soft-margin* para diferentes valores de ε, enquanto a contínua com tracejado mais forte é o hiperplano separador.

FIGURA 35 - Algoritmo de soft-margin para diferentes



FONTE: JAMES et. al (2013)

2.11.2.2 *Kernel trick*

O método de *kernel* é uma alternativa empírica para aumentar a eficiência na classificação de conjunto de dados não separáveis linearmente (Schölkopf e Smola, 2002). A técnica objetiva mudar o domínio da função, aumentando o número de dimensões através de transformadas não lineares, facilitando a separação dos dados pelo hiperplano (CARVALHO e LORENA, 2007).

Os *kernels* mais comuns são exemplificados pela tabela 3:

TABELA 3 - *KERNELS* DO ALGORTIMO SVM

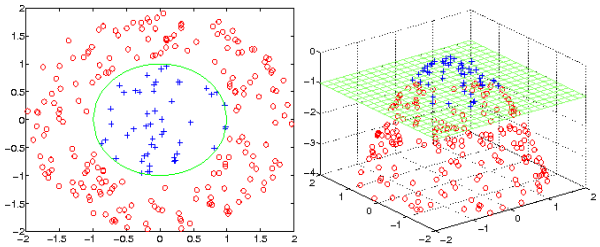
Kernel	Função
Polinomial	$(X_i X_J + k)^k$
RBF	$e^{-\sigma\|X_i - X_J\|^2}$
Sigmoid	$\tanh(\delta(X_i X_J) + k)$

FONTE: Uma introdução às *Support Vector Machines* (2019).

A escolha de qual *kernel* e quais hiperparâmetros devem ser utilizados é feita de acordo com resultados experimentais durante o processo de treinamento.

A figura 36 indica o uso do *kernel* rbf e a projeção dos dados bidimensionais em três dimensões, juntamente com o plano ótimo encontrado pelo SVM.

FIGURA 36 - SVM com *kernel* RBF



FONTE: Wu, Chang e Panda (2005).

2.12 PRINCIPAIS BIBLIOTECAS DA LINGUAGEM DE PROGRAMAÇÃO *PYTHON*

A linguagem de programa *Python* apresenta inúmeras bibliotecas, dentre elas se destacam:

- OpenCV: é uma biblioteca de visão computacional de código aberto. A biblioteca está escrita em C e C ++ e executa-se em sistemas Linux, Windows e Mac OS X. Existe um desenvolvimento ativo nas interfaces para Python, Ruby, Matlab e outras linguagens (BRADSKI e KAEHLER, 2008).

O OpenCV foi projetado para eficiência computacional e com forte foco em aplicativos em tempo real. O OpenCV é escrito em linguagem de programação C otimizada e pode tirar proveito dos processadores multicore. A biblioteca usa automaticamente a biblioteca IPP *(Intel's Integrated Performance*) apropriada em tempo de execução, se essa biblioteca estiver instalada. A figura 37 mostra a logo que ilustra a biblioteca OpenCV (BRADSKI e KAEHLER, 2008).

FIGURA 37 – Logotipo da biblioteca OpenCV

FONTE: OPENCV (2019).

- Numpy: O NumPy é o pacote fundamental para a computação científica com Python. Ele contém entre outras coisas:
  - objeto de matriz N-dimensional
  - ferramentas para integrar código C / C ++ e Fortran
  - recursos de álgebra linear, transformação de Fourier e números aleatórios

Além de seu uso científico, o NumPy também pode ser usado como um eficiente recipiente multidimensional de dados genéricos. Tipos de dados arbitrários podem ser definidos. Isso permite que a biblioteca se integre de maneira fácil e rápida a uma ampla variedade de bancos de dados (NUMPY, 2019).

- Pillow: o *Python Imaging Library*, ou PIL, é uma das bibliotecas principais para manipulação de imagens no *Python*. A *Python Imaging Library* adiciona

recursos de processamento de imagem ao seu intérprete Python (PYTHON-GUIDE, 2019).

Essa biblioteca oferece amplo suporte ao formato de arquivo, uma representação interna eficiente e recursos de processamento de imagem bastante poderosos (PYTHON, 2019).

A biblioteca de imagens principais foi projetada para acesso rápido aos dados armazenados em alguns formatos básicos de pixel. Deve fornecer uma base sólida para uma ferramenta geral de processamento de imagem (PYTHON, 2019).

- *O SciPy é o pacote básico da linguagem Python que implementa diversas técnicas úteis na computação científica. Utiliza como base o NumPy* para lidar eficientemente com grandes quantidades de números, e implementa em linguagem C diversos algoritmos numéricos e simbólicos para o processamento matemático. Entre as capacidades do SciPy, temos (SCIPY, 2019):
    - Estatísticas
    - Otimização
    - Integração numérica
    - Processamento de sinais e imagens
    - Solução de equações diferenciais
    - Funções especiais (Bessel, etc.)
    - Polinômios
- O Scikit-learn é uma biblioteca Python de código aberto de algoritmos populares de aprendizado de máquina. O projeto foi iniciado em 2007 como um projeto do Google Summer of Code de David Cournapeau. Mais tarde naquele ano, Matthieu Brucher começou a trabalhar neste projeto como parte de sua tese. Em 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort e Vincent Michel do INRIA assumiram a liderança do projeto e produziu o primeiro lançamento público. Atualmente, o projeto está sendo desenvolvido muito ativamente por uma comunidade entusiástica de colaboradores. É construído em NumPy e SciPy, as bibliotecas Python padrão para computação científica (GARRETA e MONCHECCHI, 2013).

3. MATERIAS E MÉTODOS

Para atingir os objetivos propostos para o desenvolvimento do sistema antifurto com reconhecimento biométrico da íris, são apresentados, neste capítulo, os materiais e métodos utilizados neste projeto.

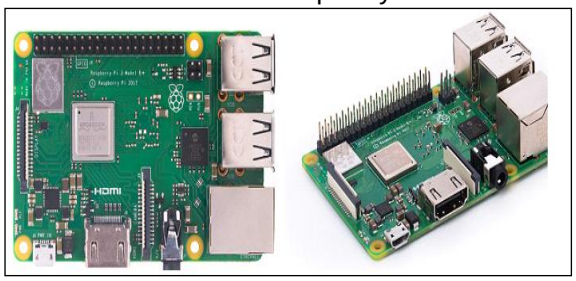
3.1 MATERIAIS UTILIZADOS

Nesta fase são detalhados os materiais necessários para a implementação do Sistema Antifurto para veículos, com base no uso de reconhecimento biométrico da íris.

Nesse sistema, desde a captura da imagem até a extração das características e autenticação, serão utilizados os seguintes componentes:

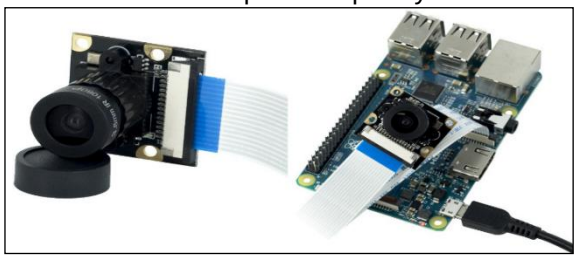
- Linguagem de programação de alto nível Python™, orientada a objetos;
- Módulo Raspberry Pi 3 Modelo B (Figura 38), com as especificações técnicas detalhadas no APÊNDICE “A”; e
- Câmera Raspberry Pi (Figura 39), com 5 Megapixel e com as especificações técnicas detalhadas no APÊNDICE “B”.

FIGURA 38 – Módulo Raspberry Pi 3 Modelo B



FONTE: www.raspberrypi.org/products/raspberry-pi-3-model-b/ (2019).

FIGURA 39 – Câmera para Raspberry Pi 3 Modelo B



FONTE: www.filipeflop.com/produto/camera-compativel-raspberry-pi-5mp/ (2019).

Nesse sentido, para que este projeto fosse viabilizado, tornaram-se necessárias as aquisições e capacitações a seguir discriminadas:

• Curso de capacitação e desenvolvimento em linguagem de programação Python, da equipe de projeto;

• Curso prático de configurações iniciais e programação utilizando o RASPBERRY Pi, dos membros da equipe de projeto;

• Compra do módulo RASPBERRY Pi 3 *Quadcore* 1.2GHz, 1GB, original ou similar para o processamento da imagem;

• Aquisição de câmera com resolução mínima de 5 Megapixels, compatível com a necessidade de captura e digitalização da imagem da íris;

• Fabricação do protótipo simulador do painel do veículo; e

• Compra de componentes eletroeletrônicos para conexões e montagens.

Os custos destas aquisições estão quantificados no APÊNDICE “C” a este trabalho.

3.2 FUNCIONAMENTO DO SISTEMA ANTIFURTO PROPOSTO

O modo de funcionamento do sistema antifurto é indicado através das etapas demonstradas no fluxograma constante da figura 41.

Ressalta-se que o funcionamento da ignição e desbloqueio é apenas uma proposta uma vez que o embarque veicular efetivo, do sistema de reconhecimento, não está contemplado neste trabalho, conforme definido nos objetivos.

O desenvolvimento do sistema antifurto concentra-se na implementação do sistema de reconhecimento biométrico da íris cuja saída possa, ou não, permitir a ignição e/ou dirigibilidade do veículo.

Caso não ocorra a autenticação do usuário, o sistema de reconhecimento é reiniciado (*reset*) e fica aguardando novo acionamento.

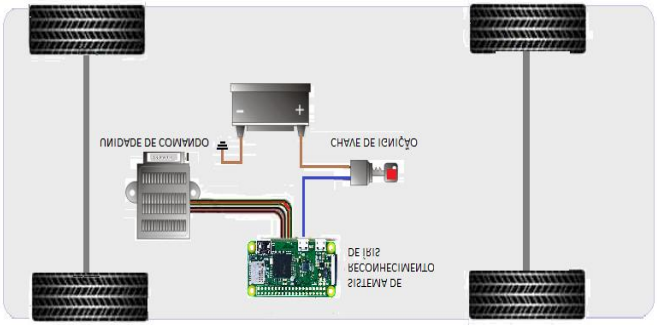
Propõe-se neste trabalho o desenvolvimento do sistema de reconhecimento biométrico da íris. As implementações veiculares não serão contempladas neste projeto, sendo estas consideradas, apenas, como saída possível do sistema de reconhecimento e serão previstas como sugestão de trabalhos futuros.

O esquemático do leiaute do módulo de reconhecimento da íris, uma vez embarcado, no veículo, pode ser exemplificado na figura 40.

A ligação e desbloqueio do veículo poderão seguir as seguintes etapas:

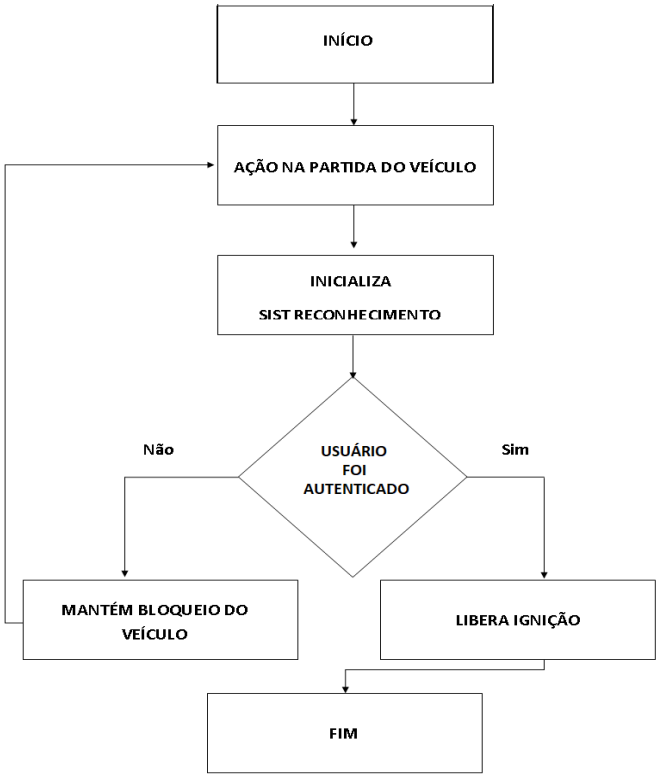
- A ignição atingirá o primeiro estágio onde são ligados os acessórios do veículo como luzes, rádio, etc. Neste estágio o Sistema de Reconhecimento inicializar-se-á e, após 5 segundos, fará a primeira tentativa de captura de imagem, repetindo procedimento por mais duas vezes, a cada 5 segundos.
- Restando infrutífera a aquisição da imagem, um led vermelho no painel indicará erro de identificação e a chave da ignição deverá ser reiniciada.
- Uma vez capturada a imagem do olho do usuário, o sistema efetua o processamento e, caso autenticado positivamente, a ignição é liberada para o segundo estágio. Como consequência ocorre o desbloqueio do veículo e fica possibilitado o acionamento do motor.
- Caso a autenticação seja negativa, o veículo permanece bloqueado e o procedimento pode ser reiniciado.
- Visando garantir a segurança do usuário em caso de roubo e/ou sequestro, há a possibilidade de que numa segunda tentativa, caso permaneça uma autenticação negativa, o motor seja desbloqueado e o veículo liberado, porém, seja enviada a identificação do condutor para a polícia com alerta de veículo roubado e/ou haja vítima, via *Short Message Service (*SMS) ou outra tecnologia GSM.

FIGURA 40 – Proposta de módulo de reconhecimento de íris embarcado



FONTE: Os Autores (2019).

FIGURA 41 – Fluxograma do sistema antifurto com reconhecimento de íris



FONTE: Os Autores (2019).

3.2.1 Esquemático proposto para embarque veicular do sistema de reconhecimento

Tendo em vista este projeto vislumbrar não somente um acessório antifurto, mas uma alternativa que desestimule o infrator de cometer o ato ilícito, dadas às dificuldades que o sistema oferece, sugere-se um indicador visual que alerte quanto à existência do dispositivo no veículo.

Tais dificuldades causam um desencorajamento à consumação do crime que, nos últimos anos vem atualizando suas técnicas, *modus operandi* e conhecimentos acerca de novas de tecnologias anticrimes existentes.

O adesivo, com cores fortes, possui uma dimensão de 0,20x0,12 metros (Figura 42), tornando possível sua identificação a uma distância de até 50 metros.

Essa fácil visualização permite impressionar os campos afetivo, cognitivo e reguladores de emoções do contraventor de modo que este não crie nenhuma empatia pelo veículo e venha a conceber o furto ou roubo.

Nesse sentido, propõe-se que o adesivo seja fixado no veículo (Figura 43), alertando que aquele automóvel possui Bloqueio Biométrico, sem necessidade de

fornecer maiores informações acerca dos(s) tipo(s) de identificação biométrica existentes no veículo.

FIGURA 42 – Adesivo sugerido para indicação do sistema de bloqueio biométrico no veículo

FONTE: Os Autores (2019).

FIGURA 43 – Adesivo fixado no veículo para alerta de veículo bloqueado

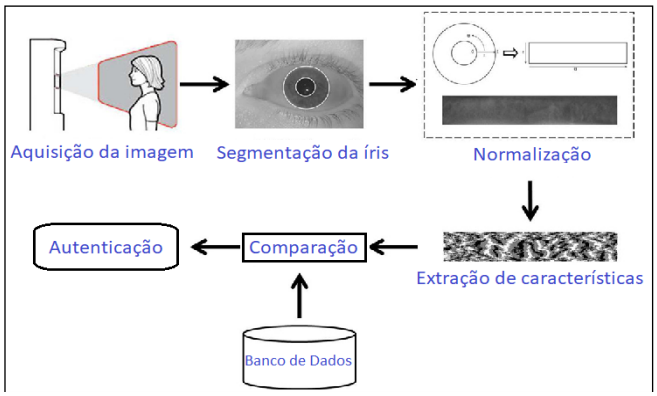
FONTE: Os Autores (2019).

3.3 FUNCIONAMENTO DO SISTEMA DE RECONHECIMENTO PROPOSTO

O funcionamento do sistema de reconhecimento biométrico da íris, este sim efetivamente implementado, consistirá, de forma geral, do processo de aquisição da imagem do olho do usuário, do processamento digital dessa imagem e posterior comparação com modelos de um banco de dados, concluindo com a autenticação do usuário.

As etapas do reconhecimento da íris inicia-se a aquisição da imagem, passa pelos processos de segmentação, normalização, extração das características e a comparação dessa imagem com outros modelos, processados da mesma maneira, constantes de banco de dados, resultando na autenticação do usuário, conforme diagrama de blocos, constante da figura 44.

FIGURA 44 – Diagrama de blocos de um sistema de reconhecimento de íris



FONTE: Adaptado de Jillela e Ross (2015, p 4-16)

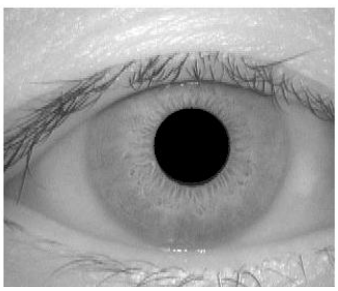
3.3.1 Detalhamento das etapas do funcionamento do sistema proposto

- **Acionamento da Ignição do veículo**

  Uma vez que o usuário adentre o veículo e acione o dispositivo de ignição, iniciar-se-á o sistema de reconhecimento biométrico, que poderá permitir ou não o processo de ignição.

- **Inicialização do Processo de Reconhecimento**
  - **Aquisição da imagem**: essa etapa corresponde à captura da imagem do olho do usuário, através da câmera digital fixada no para-brisa do veículo. Essa imagem é digitalizada em arquivo com formato de compressão de imagens fotográficas JPEG (*Joint Photographics Experts Group*), e armazenada para tratamentos posteriores. Um exemplo de imagem é mostrado na figura 45.

FIGURA 45 – Imagem em formato digital em formato JPEG

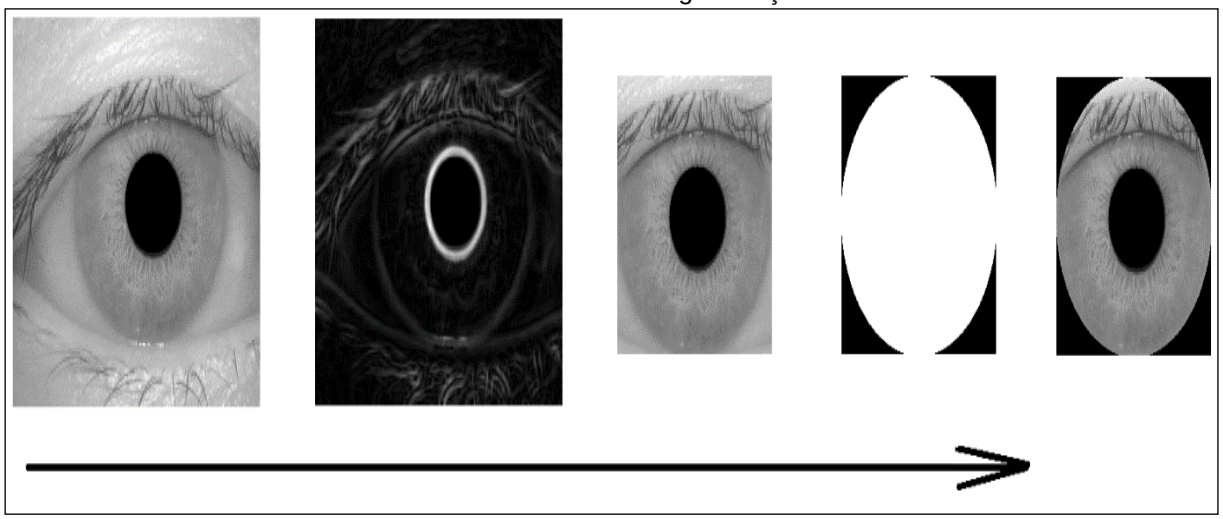


FONTE: Os Autores (2019).

- **Segmentação da íris:** essa etapa corresponde à localização e segmentação da íris, pupila e pálpebras. A segmentação é complexa e essencial em um processamento digital de imagem e pode definir sucesso ou fracasso nos resultados finais (GONZALES e WOODS, 2002);

  Utilizando-se da faixa mais baixa do histograma, identifica-se a pupila e, com a binarização da imagem e o uso de transformadas para destacar os círculos existentes na figura 46, encontra e vetoriza-se o possível centro da pupila. Com o uso do histograma da imagem e filtros para detecção de bordas, encontra-se o possível centro da íris e dimensiona-se o seu círculo. A partir da máscara encontrada, destaca-se o anel da íris extraído da imagem original.

FIGURA 46 – Processo de Segmentação da íris



FONTE: Os Autores (2019).

- **Normalização da íris:** a partir do anel da íris extraído na segmentação, possui possíveis inconsistências ainda como efeito das variações de distância da imagem, da iluminação, do posicionamento da cabeça, das dilatações da pupila, da rotação dos olhos etc. Para compensar essas inconsistências é realizada a transformação do Sistema de Coordenadas da imagem.
- **Extração de características:** esta etapa se mostra como uma das mais importantes etapas da autenticação desse sistema biométrico e baseia-se na extração das características mais distintas presentes em uma imagem.

As informações sobre a textura da íris devem ser extraídas e codificadas para possibilitar comparações corretas entre modelos. As características morfológicas são traduzidas para um padrão matemático, possível de utilização por um sistema de comparação.

- **Comparação**: realizada entre os códigos de Iris da imagem adquirida e as imagens do banco de dados. Nesse processo propõe-se o uso do SVM (*Support Vector Machines*), ou seja, os sistemas de aprendizagem de máquina treinados com algoritmo de Otimização Matemática, conforme apresentado no Capítulo 2.

Após a etapa de comparação tem-se o resultado do processamento do sistema reconhecimento da íris, cuja saída será uma característica autenticada ou não.

O resultado, no presente trabalho, definirá o desbloqueio da ignição do veículo e/ou possibilitará sua dirigibilidade.

De uma maneira geral, este trabalho teve a sua implementação balizada pelo atendimento a uma demanda do mercado, mas, prioritariamente, mantendo o foco em um design simples, seguro, eficiente e com baixo custo.

Sistemas baseados em sensores e leitores biométricos que objetivam proporcionar maior segurança, conforto e comodidade ao usuário estão cada vez mais comuns nos mais diversos campos, como por exemplo, na segurança de instalações, na domótica e no uso automotivo. A concepção do projeto baseou-se no aprimoramento e aplicações das tecnologias existentes, agregando uma característica biométrica, de modo a aumentar a segurança, ainda que em detrimento da elevação da complexidade e no custo de implementação.

No Estudo do Estado da Arte verificou-se que tem se intensificado os projetos automobilísticos, utilizando os princípios biométricos para determinadas funções como, por exemplo, permitir a ignição do automóvel, controlar climatização, GPS, ligar computador de bordo personalizado, dentre outras.

Um relatório da Equipe de Mobilidade Inteligente da empresa internacional de consultoria Frost & Sullivan, prevê que um terço dos carros novos incorporará sensores biométricos até 2025 (Revista Economia IG, 2017).

3.4 SISTEMA DE RECONHECIMENTO DE ÍRIS

O sistema de reconhecimento de íris proposto, é composto pelas etapas apresentadas na figura 7.

Primeiramente, a aquisição das imagens dos voluntários foi realizada por um Raspberry 3 Pi (autores desse trabalho). Nesta ocasião, foi usado um módulo que é composto por uma câmera e conjunto de *leds* mostrados na figura 39. A câmera realiza capturas de fotos em resolução de 5 Megapixels, além de ter dois *leds* que emitem luz em NIR (*Near Infrared*), mas precisamente na faixa de 850 nm no espectro eletromagnético.

Foi necessária a troca da lente do módulo por uma lente com uma distância focal que compreende a faixa de 6 a 22 mm, assim como as operações de foco e *zoom* da lente são feitas de maneira manual. Sendo assim, foi possível realizar a operação de *zoom* e foco na imagem dos olhos do voluntário com contraste e enquadramento necessários.

Fez-se necessário agregar às capturas de fotos dos voluntários, a primeira versão do banco de dados de íris CASIA. O qual é composto por 108 usuários, contendo 7 fotos por usuário, totalizando 756 fotos de íris.

A aquisição de imagem é uma das etapas mais importantes para o sistema proposto, pois a imagem deverá carregar as características do olho de um indivíduo com nitidez, contraste e qualidade de imagem. As imagens, objetos de estudo deste trabalho, possuem boa qualidade, apresentando rico conteúdo ou informação, principalmente na região da íris do olho.

As nervuras da íris podem ser visualizadas com exatidão, pois a câmera do sistema proposto tem resolução na escala de megapixel, uma maior quantidade de pixel revela mais conteúdo e torna o sistema mais confiável em termos de precisão.

As imagens adquiridas são objetos de trabalho para a etapa posterior que é a segmentação da íris. Geralmente a região das pálpebras, cílios, pupila, esclera e região do olho no rosto são descartadas, pois são facilmente mutáveis por fatores externos ao sistema. A íris é uma característica de nascença de uma pessoa e por isso é a razão primordial para sua escolha no sistema proposto neste trabalho.

A segmentação da íris é um processo que exige recursos matemáticos e de programação para serem usados. Nos Apêndice “D” encontra-se o código que

realiza a tarefa, cuja primeira etapa é a localização da pupila, pois o centro desta é a referência de coordenadas para o restante do processamento digital de imagem a seguir.

Primeiramente, a função **segmentacao_iris()** recebe como parâmetro a imagem, a qual é convertida de RGB (imagem colorida) para escala de cinza com 8 bits de profundidade. Tendo o entendimento de que a pupila geralmente tem um menor valor digital comparado ao restante da imagem do olho, é possível segrega-la em relação a toda a imagem pelo processo de binarização, no qual é utilizada uma função da biblioteca scipy em linguagem de programação *Python* para a construção de um histograma dividido em cinco partes. A função *numpy.histrogram* calcula o histograma de um conjunto de dados. Nesta função tem-se, como parâmetros, a divisão (*bins*) e a faixa de valores do histograma.

Para a seleção da pupila utilizou-se da menor parte do histograma, em valor digital, desta forma possibilitou isolar a pupila do restante da imagem, de forma que a área da pupila ficasse evidenciada. Para essa atividade multiplicou-se pixel por pixel da região do histograma, com os menores valores, pelo valor digital de 255.

A partir da representação da pupila é possível realizar a Transformada de *Hough* para detecção de círculos, utilizando-se de uma função da biblioteca *OpenCV chamada cv.HoughCircles* (OPENCV, 2019). Em suma os parâmetros de *cv.HoughCircles* são: a imagem de entrada, o método de detecção de círculos (CV_HOUGH_GRADIENT), a taxa de inversão da resolução, a mínima distância entre círculos detectados, os limites superior e inferior do Filtro de Canny e os raios máximo e mínimo dos círculos detectados.

Obtém-se um círculo resultante da Transformada de *Hough* que envolve a pupila e o centro desse círculo, usado para posteriores operações. Além disso a pupila é preenchida com a cor negra para eliminar possível reflexos de iluminação dos *leds*. Após esta etapa da segmentação da íris, é realizada a suavização da imagem através da função *cv.blur*, a qual desfoca uma imagem usando o filtro de caixa normalizado (OPENCV, 2019). A suavização é resultado de um filtro que realça os contornos da região dos olhos realçados. A função possui um núcleo que é determinado pelos parâmetros inseridos nela.

Assim sendo, o Filtro de Sobel, usado na detecção de contornos, é usado para realçar os contornos da imagem que compreendem as regiões onde há

mudanças bruscas de valor digital. O ecossistema baseado em *Python* de *software* de código aberto para matemática chamada *Scipy* tem uma função que aplica Sobel em uma imagem chamada *ndimage.sobel*. Essa função calcula o primeiro, o segundo, o terceiro ou os derivados de imagem mista usando um operador Sobel estendido (SCIPY, 2019).

A partir de então é aplicada novamente a TH para detecção de círculos. Nesta etapa o objetivo é detectar a circunferência da íris, com as coordenadas do centro que mais se aproximam das coordenadas do centro da pupila. Essa seleção é feita por um laço de repetição para todas as circunferências da íris detectada, encontrando, assim, a menor diferença entre o centro da pupila e da íris é selecionada, assumindo como o menor valor entre os raios detectados.

Detectadas as circunferências da pupila e íris, constrói-se uma máscara para a realização de uma operação lógica ***and*** com a imagem original, operação é realizada pela função da biblioteca *numpy,* em linguagem *Python,* chamada *numpy.copyto* que faz uma cópia da íris recortada sobre a imagem dos olhos, apenas onde a máscara está construída (NUMPY, 2019). O intuito desta operação é selecionar apenas o conteúdo da imagem que abrange a íris, pois a máscara contempla uma região da imagem com nível lógico alto que, multiplicado com a imagem original descarta toda a região fora da íris.

A normalização da íris, no sistema proposto, é realizada diferentemente do que é feito pelos sistemas de reconhecimento, na qual a íris é transformada de coordenadas polares para cartesianas.

Optou-se, por razões de otimização de processamento computacional por não realizar a transformação entre sistemas de coordenadas e selecionar uma Região de Interesse (*Region of Interest* – ROI) o que, neste caso, são as regiões laterais internas da íris. Essas ROI's foram construídas armazenando-se as medidas em comprimento e altura em uma variável, a qual é o parâmetro de entrada para a função da biblioteca *Pillow* em linguagem de programação *Python,* chamada *PIL.Image.crop* que realiza a destacamento da imagem original com as medidas passadas como parâmetro (PILLOW, 2019).

Faz-se necessário a segmentação de cada íris do banco de dados CASIA. É apresentado, no Apêndice "E", um código em linguagem de programação *Python que* contém uma função chamada **inscricao_modelo_casia1().** Essa função realiza

a segmentação e salvamento, em diretório, de cada uma das íris do banco de dados (CASIA) utilizando-se do conceito de Computação Paralela, para tornar esse processo o mais rápido possível.

A próxima etapa para o sistema proposto é a extração de características e o código para sua realização é mostrado em Apêndice “F”. A função **extracao_caracteristica()** é usada neste contexto para realizar a extração de características, abordando todos os processos necessários para tal tarefa. A característica a ser extraída no sistema proposto é a textura, que é a principal e mais usada característica a ser extraída por um sistema de reconhecimento de íris. Para a extração de características é utilizado o Filtro Gabor, o qual pode ser definido pela Equação 5.

O Filtro Gabor tem um ***kernel*** ou núcleo que é uma matriz com valores correspondentes à equação proposta por Dennis Gabor. Esses coeficientes ou valores são extraídos de uma rotina que os trata como dados de entrada na Equação 5 e, após essa operação matemática, cada posição do *kernel* assume um valor que é o resultado do cálculo.

A imagem resultante da segmentação é usada para realizar a convolução com *kernel* do Filtro Gabor. Foi utilizada uma função da biblioteca *OpenCV* chamada *cv.filter2D*, a qual convolui uma imagem com um *kernel* (OPENCV, 2019) e recebe como parâmetro a imagem e o *kernel* com os coeficientes do Filtro Gabor, assim como é definida a profundidade de bits da imagem resultante. A matriz resultante é composta por valores resultantes da convolução pixel por pixel entre *kernel* e imagem de entrada. A partir de então, é utilizado um laço de repetição percorrendo o vetor de características e, para cada posição o vetor que recebe uma matriz do tamanho do *kernel*, desta forma é calculada a média e desvio padrão com as funções do ecossistema *Scipy* chamadas *numpy.mean* e *numpy.std*.

O vetor resultante da extração de característica é usado na técnica de aprendizado de máquina supervisionado, chamada SVM (*Supported Vector Machine*). Para as rotinas relacionadas com aprendizado de máquina utilizou-se de uma biblioteca chamada *scikit-learn*. O vetor possui o tamanho de 288 valores que são características que o SVM contém, é dividido em 540 fotos cadastradas para treinamento e 216 fotos para teste da SVM. O código responsável pela aplicação de técnicas de aprendizado de máquina é encontrado no Apêndice “G”.

A função *main* do código do Apêndice “G” realiza toda a inscrição do banco de dados CASIA, extração de características das íris, aprendizado de máquina de todos os dados, bem como valida e apresenta o resultado final do sistema.

Primeiramente, todas as fotos do banco de dados de íris são armazenadas estrategicamente divididas entre dados de treinamento e teste, laços de repetição percorrem todo o diretório onde estão armazenados. Em cada imagem presente no diretório é feita a extração de característica tanto para treinamento quanto para teste. Para cada usuário é criado um diretório com sua classe, em suma, há 108 classes e 288 valores que correspondem às características. Para treinamento do classificador do SVM é necessário uma entrada e uma saída. A entrada corresponde aos dados acessados no diretório especificado e a saída são as classes conhecidas previamente pela máquina, pois é rotulado pelo programa. Assim sendo, o classificador do SVM utiliza parâmetros para sua implementação e esses podem assumir inúmeros valores. Para escolher os valores, optou-se por usar um método para estimar os parâmetros que geram uma maior precisão do classificador, este método é chamado de pesquisa exaustiva de grade. A pesquisa exaustiva de grade, fornecida por *GridSearchCV* (classe que implementa métodos de aprendizado de máquina), gera exaustivamente candidatos a partir de uma grade de valores de parâmetros especificados com o parâmetro *param_grid* (SCIKIT-LEARN, 2019). A variável *param_grid* tem como argumentos “C”, *gamma* e o *kernel*. O parâmetro “C” é a penalidade do termo de erro, *gamma* é coeficiente do *kernel* e o parâmetro kernel especifica o tipo de kernel a ser usado no algoritmo (SCIKIT-LEARN). Desta forma, foi determinada uma faixa de valores para cada parâmetro e os métodos da biblioteca se encarregam de escolher os valores que fornecem o melhor desempenho. A classe *GridSearchCV* possui o “método ***fit”*** que é usado para treinamento dos dados e “***predict”*** que faz a previsão de dados no classificador, com os parâmetros apresentados.

A verificação é feita a partir da geração de um arquivo proporcionado por um módulo das bibliotecas padrões da linguagem *Python* chamado *pickle*. Esse módulo implementa protocolos binários para “serializar” e “desserializar” uma estrutura de objeto *Python*. A função *pickle.dumps* é usada para serializar uma hierarquia de objeto e a função *pickle.loads* é usada para desserializar um fluxo de dados (*Python*,

2019). Logo após realizar essa rotina, a imagem de entrada é inserida como parâmetro na função *predict* e o resultado da classificação advém desta função.

A avalição do sistema de reconhecimento de íris é medida a partir das métricas da biblioteca *sklearn* para aprendizado de máquina: precision, recall e F1 score, que são explicados no capítulo 2 deste relatório.

A avaliação do modelo é feita pelo atributo *cv_results* da biblioteca *sklearn* da classe *GridSearchCV* de cada componente do modelo de aprendizado de máquina, assim como pelos métodos ***metrics.accuracy_score***, ***metrics.f1_score*** e **metrics.recall_score**.

3.5 PROTOTIPAGEM

Foi implementado um protótipo para simular o painel de um automóvel onde estaria embarcado o sistema de reconhecimento.

O protótipo foi construído em placa de fibra de média densidade (Medium-Density Fiberboard – MDF) e filme plástico transparente, como apresentado nas figuras 47 a 50.

FIGURA 47 – Protótipo de simulação com o sistema de reconhecimento acoplado

FONTE: Os Autores (2019).

O módulo com a câmera foram fixados na posição frontal, de maneira lateralizada ao usuário para que não houvesse empecilhos para a dirigibilidade do

suposto motorista. A lente da câmera é ajustada de maneira fixa, desta forma a lente foi configurada até que o enquadramento de um dos olhos do indivíduo ficasse nas proporções corretas para o processamento.

FIGURA 48 – Protótipo de simulação com o sistema de reconhecimento acoplado

FONTE: Os Autores (2019).

FIGURA 49 – Protótipo de simulação com o sistema de reconhecimento acoplado

FONTE: Os Autores (2019)

Os testes de validação do sistema foram configurados de forma a capturar a íris do suposto motorista que se posiciona na frente da câmera. As capturas são realizadas sempre que o usuário acessar a ignição do suposto veículo. A figura 50 mostra o posicionamento do motorista para aquisição de imagens.

FIGURA 50 – Protótipo de simulação com o sistema de reconhecimento acoplado

FONTE: Os Autores (2019).

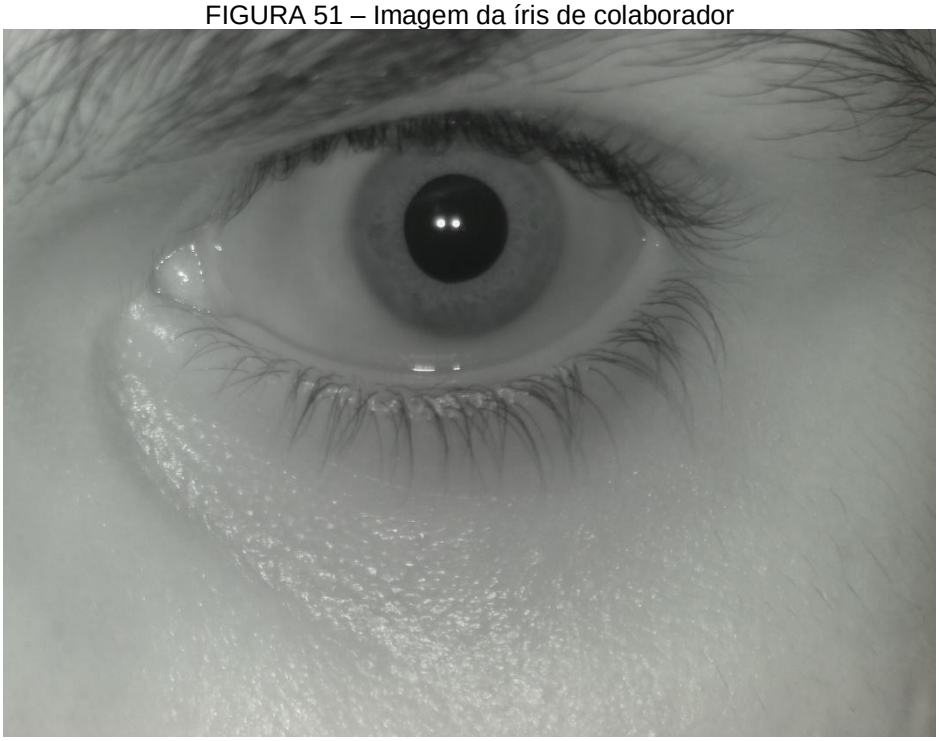
4 RESULTADOS E DISCUSSÃO

Os resultados obtidos são advindos da aquisição de fotos feita pelo módulo que é composto por uma câmera e conjunto de *leds* mostrados nas figuras 39 e 40.

As especificações do produto objetivam simular as condições apresentadas pela primeira versão do banco de dados de íris CASIA.

A primeira parte do sistema proposto é a aquisição de imagens, que na ocasião foi realizada com os autores deste trabalho. Na figura 51 é mostrada a imagem da íris de um dos voluntários (autores deste trabalho).

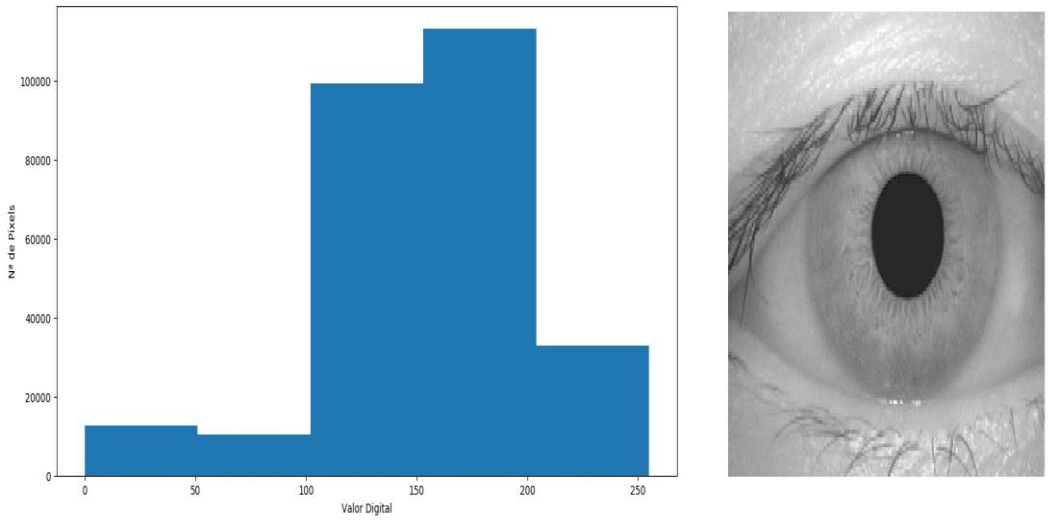
FIGURA 51 – Imagem da íris de colaborador



FONTE: Os Autores (2019).

A segmentação é outra etapa importante do sistema proposto, que é composta por uma série de tratamentos em imagens digitais. Neste contexto, a binarização é uma técnica de processamento digital de imagem usada para evidenciar uma região, resultando em uma imagem com valores binários, assim sendo é usada neste trabalho para destacar a pupila das demais regiões. Nota-se que a pupila foi identificada com êxito com esse tratamento, a figura 52 mostra a imagem original da íris de um voluntário do banco de dados CASIA e o histograma correspondente.

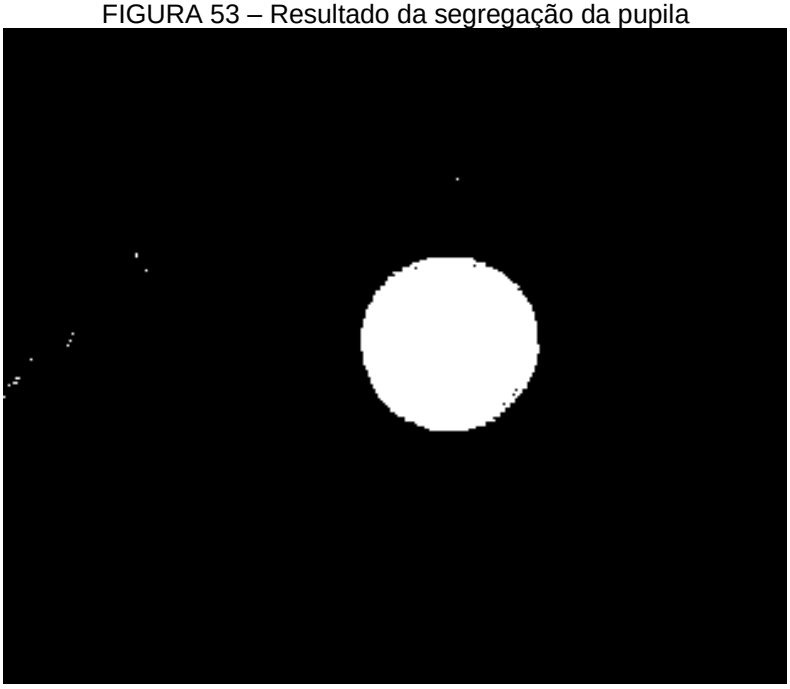
FIGURA 52 – Imagem da íris de colaborador e histograma



FONTE: Os Autores (2019).

A figura 53 mostra o resultado após a segregação da pupila através da construção de um histograma e posterior binarização da pupila.

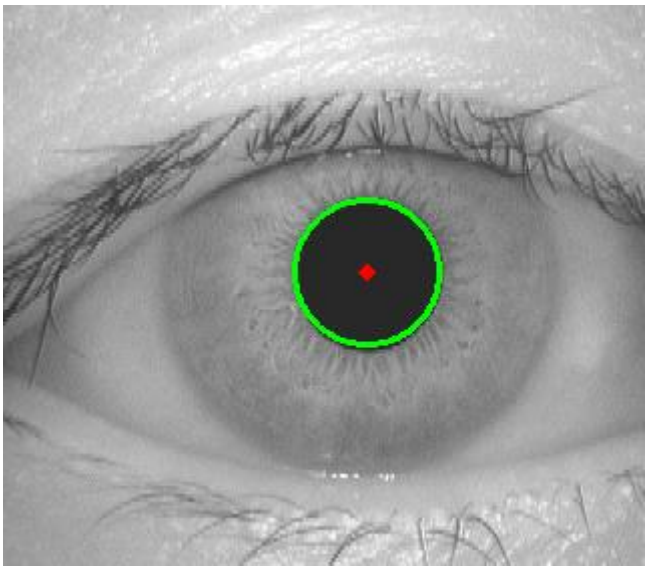
FIGURA 53 – Resultado da segregação da pupila



FONTE: Os Autores (2019).

A figura 54 mostra a imagem da pupila segregada com a indicação de uma circunferência em seu entorno, essa indicação é resultado da aplicação do conceito da Transformada de *Hough* para identificação de círculos.

FIGURA 54 – Imagem da pupila segregada com aplicação da Transformada de Hough



FONTE: Os Autores (2019).

O próximo passo do sistema proposto é a identificação de contornos na imagem e para isso fez-se necessário o uso de um filtro que suaviza os contornos da íris, como visto na figura 55, este tratamento é primordial para um melhor desempenho do filtro de detecção de contornos, o Filtro de Sobel.

FIGURA 55 – Imagem da pupila com contornos suavizados

FONTE: Os Autores (2019).

Na figura 56 é mostrado o resultado da aplicação do Filtro de Sobel. Os contornos observados na imagem são resultados da identificação dos gradientes de

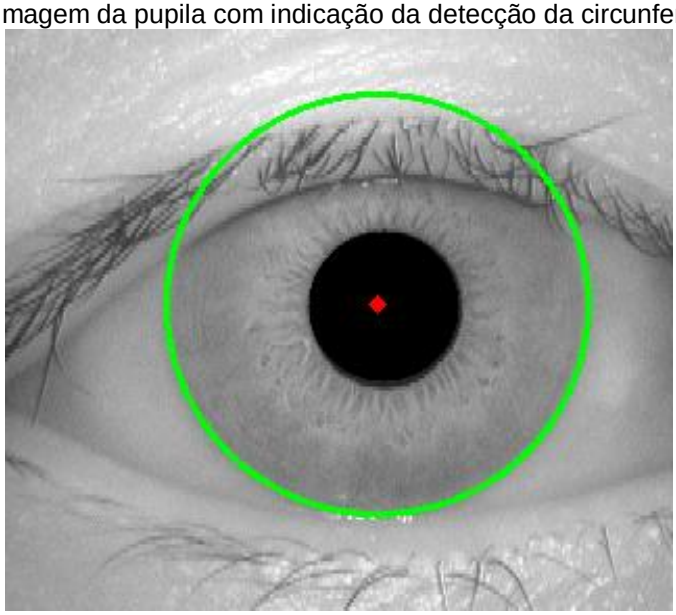
valor digital presentes na imagem, nessas regiões a mudança de uma região para outra é evidente e então esse filtro é responsável por evidenciar esse comportamento.

FIGURA 56 – Imagem da íris de colaborador com aplicação do Filtro de Sobel

FONTE: Os Autores (2019).

A aplicação da Transformada de Hough para a detecção de círculos se faz necessária novamente, desta vez para a identificação da circunferência da íris, na figura 57 é mostrado a indicação da detecção da circunferência da íris.

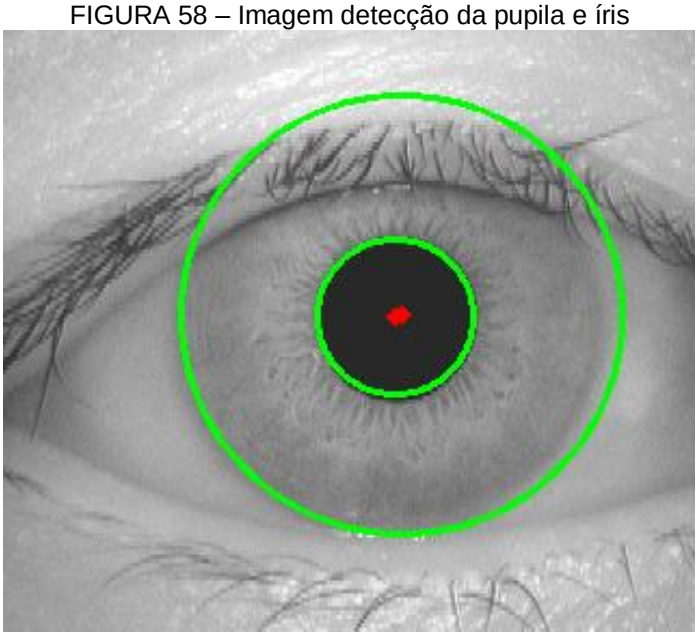
Figura 57 – Imagem da pupila com indicação da detecção da circunferência da íris



FONTE: Os Autores (2019).

Na figura 58 é apresentada a imagem com a detecção da pupila e íris.

FIGURA 58 – Imagem detecção da pupila e íris



FONTE: Os Autores (2019).

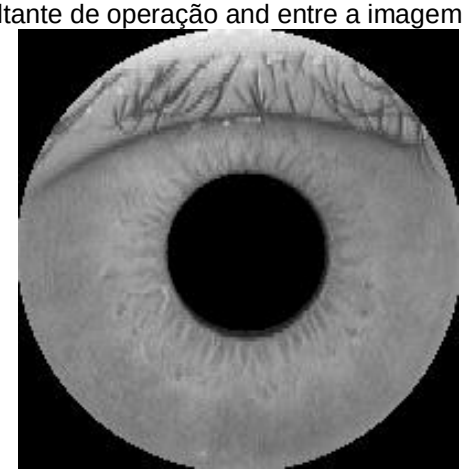
A partir de então, é criada uma máscara binária para destacamento da íris em relação ao restante da imagem, como mostrado na figura 59.

FIGURA 59 – Máscara binária criada para destacamento da íris

FONTE: Os Autores (2019).

A íris recortada é mostrada na figura 60, sendo resultante de uma operação binária ***and*** entre a imagem original e a máscara construída para esse propósito.

FIGURA 60 – Resultante de operação and entre a imagem original e a máscara

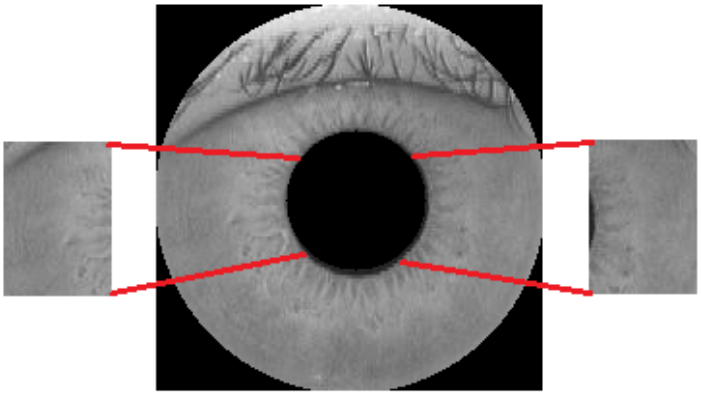


FONTE: Os Autores (2019).

Nota-se que a região da pupila e cílios ainda estão presentes na imagem e neste contexto segregou-se partes (laterais) da íris para descartar regiões irrelevantes para o processamento.

O resultado de toda a segmentação são dois segmentos de íris correspondentes ao usuário do sistema, como mostrado na figura 61. Essas imagens (segmentos de íris) é fonte para todo o processamento feito posteriormente, ou seja, a extração de característica destas imagens em questão.

FIGURA 61 – Resultado final da segmentação



FONTE: Os Autores (2019).

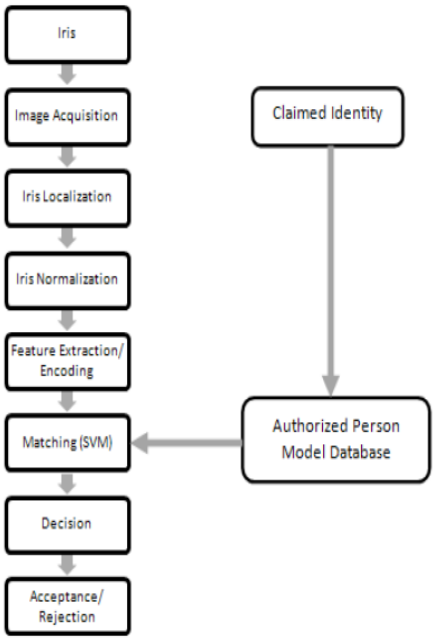
A extração de características para cada íris resultou em um vetor com coeficientes correspondentes a operações matemáticas com a expressão algébrica

que representa o Filtro Gabor. Cada vetor de característica é classificado através de técnicas de aprendizagem de máquina.

A verificação é uma etapa do sistema de reconhecimento de íris que compara todas as imagens do banco de dados de íris com a imagem de entrada do sistema. A verificação classifica em qual classe o usuário está inserido e partir de então, tendo conhecimento da classe que pertence ao dono do automóvel, o sistema libera ou bloqueia o suposto veículo.

Para a construção do sistema, toma-se como base o diagrama apresentado na figura 62.

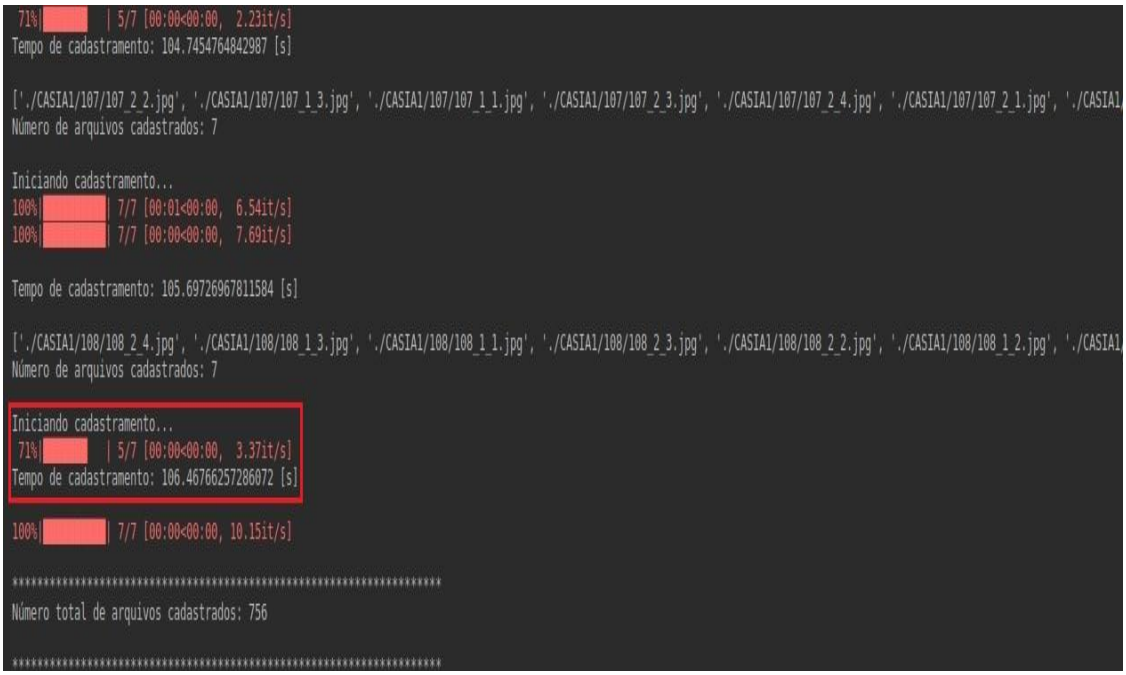
FIGURA 62 – Estrutura básica de sistemas de verificação de íris



FONTE: SALAMI e ALI (2011).

Primeiramente, na etapa de verificação é necessário realizar as operações de processamento digital de imagem para cada foto no banco de dados CASIA1, realizando uma operação denominada inscrição do banco de dados, na inscrição é feita de maneira automática a segmentação de todas as íris salvando-as em um diretório específico. Esse processo demanda um tempo de processamento que leva cerca de 106 segundos, o resultado da inscrição do banco de dados é mostrado na figura 63.

FIGURA 63 – Tempo de processamento da inscrição de imagens do CASIA



FONTE: Os Autores (2019).

De acordo com os testes feitos por SALAMI e ALI (2011), tendo como base o fluxograma da figura 62, para 5 usuários os resultados de classificação com algoritmos baseados em SVMs tem-se os dados apresentação na tabela 4.

TABELA 4 – Resultado de classificação do SVM proposto por SALAMI e ALI (2011)

Usuário autorizado	Time de treinamento (sec)	Resultado de classificação (%)
Usuário 1	0.0781	100
Usuário 2	0.0781	100
Usuário 3	0.0625	100
Usuário 4	0.1094	100
Usuário 5	0.0781	100
Média	0.0812	100

FONTE: SALAMI e ALI (2011).

Os dados extraídos do presente trabalho para a classificação de 5 usuários possui os resultados apresentados na tabela 5. Assim como, é mostrado no Apêndice “H”.

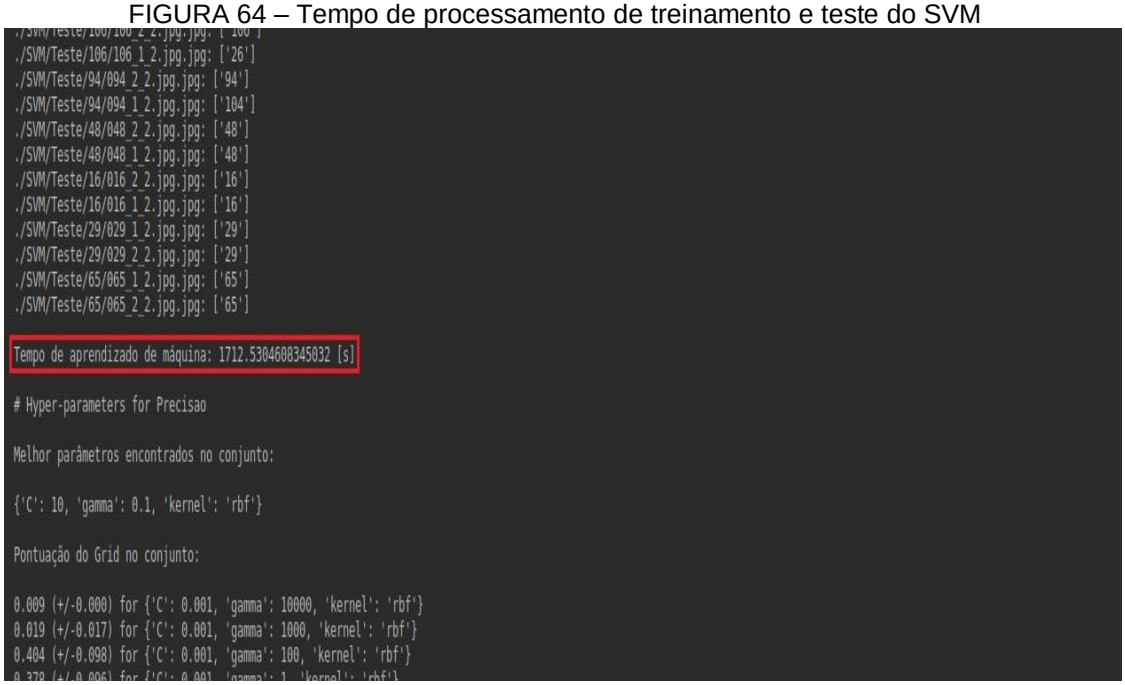
TABELA 5 – Resultado de classificação do SVM do sistema proposto

Usuário autorizado	Resultado de classificação (%)
Usuário 1	100
Usuário 2	100
Usuário 3	100
Usuário 4	100
Usuário 5	100
Média	100

FONTE: Os Autores (2019).

Após realizar a inscrição, são extraídas as características de todos os modelos de íris do banco de dados. O sistema de aprendizado de máquina foi dividido em dois setores que são treinamento e teste, todo o processo leva aproximadamente 1.712 segundos para ser realizado, como mostrado na figura 64. Considerando o tempo discorrido para treinamento e teste do classificador SVM, infere-se que o valor de tempo é muito alto e implica na usabilidade do sistema, aspecto que pode ser melhorado em trabalhos futuros.

FIGURA 64 – Tempo de processamento de treinamento e teste do SVM



FONTE: Os Autores (2019).

Faz-se necessário a avaliação do modelo de aprendizado de máquina para validar a precisão e funcionamento do sistema proposto. O *precision*, recall e *F1 score* são métricas utilizadas nos mais variados modelos de aprendizado de máquina para selecionar o melhor algoritmo para implantação em sistemas de tomada de decisão ou análise de dados. Neste caso, essas métricas são levadas em consideração para avaliar os resultados presentes neste trabalho. Essas métricas possuem uma faixa de porcentagem de 0 a 100%, portanto quanto maior o valor, maior será o índice de acerto do sistema.

Após tentativas de validação do sistema, conclui-se que a extração de características é uma etapa primordial para a validação do sistema. Ajustes nos parâmetros da Equação (5) foram realizados e chegou-se nos resultados apresentados na figura 65 e que também são apresentados na tabela 6

TABELA 6 – Métricas de classificação do SVM do sistema proposto

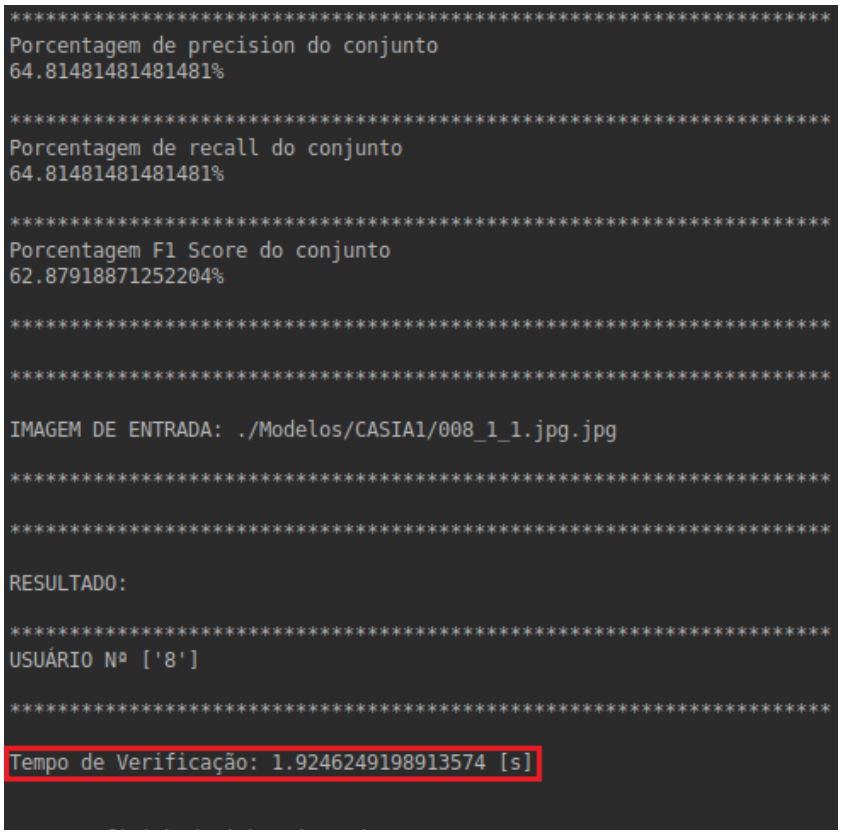
	Precision	***Recall***	***F1 Score***
Porcentagem	64,81 %	64,81 %	62,87 %

FONTE: Os Autores (2019).

Em Apêndice “I” encontra-se os resultados encontrados para cada classe de usuário do aprendizado de máquina, os melhores parâmetros (C, gamma e kernel) da classe *GridSearchCV*, bem como as porcentagem de *precision*, *recall* e *F1 score*. O *kernel* utilizado é o *rbf* (*Radial Basis Function*), que é um utilizado em algoritmos de aprendizado de máquina e o que apresentou melhores resultados (maior *precision, recall e F1 score*).

O resultado da etapa de aprendizado de máquina é um arquivo que é criado no diretório do programa principal no qual está inserido, Este arquivo carrega todos os dados de aprendizado de máquina do modelo treinado. Após a criação desse arquivo, ele é usado para a verificação da classe do suposto usuário. O resultado da comparação do modelo de íris de entrada (imagem da íris capturada pelo sistema) em relação aos modelos presentes no banco de dados CASIA1 é mostrado na figura 65. A verificação do usuário levou aproximadamente 2 segundos.

FIGURA 65 – Resultado final do processamento de reconhecimento



FONTE: Os Autores (2019).

Tendo como base o fluxograma apresentado na figura 62, faz-se necessário a implementação de um algoritmo de decisão baseado nos resultados do classificador SVM, que no caso é sugerido como objeto de estudo para trabalhos futuros.

5 CONCLUSÃO

O presente trabalho foi fundamental para criar uma percepção mais clara do papel da Engenharia que, ao aplicar e integrar tecnologias existentes, é possível oferecer um produto útil para a sociedade.

Propiciou, também, um aprimoramento do processo ensino/aprendizagem com a adequação da teoria à prática, em que foram utilizados conhecimentos adquiridos em grande parte das disciplinas ministradas no Curso de Engenharia Elétrica, com Ênfase em Sistemas Eletrônicos Embarcados. A construção de um sistema de reconhecimento demandou o uso de técnicas de PDI (Processamento Digital de Imagem), programação, eletrônica e demais conhecimentos desenvolvidos no âmbito da Engenharia.

O sistema de reconhecimento de íris proposto neste trabalho consolidou uma vasta gama de conhecimento advindo da literatura nas áreas da engenharia biomédica, eletrônica e da computação. Através dos estudos feitos com biometria, foi possível realizar a concepção do modelo proposto neste trabalho, assim sendo fez-se necessário selecionar a opção de biometria mais conveniente para a área automobilística. Diversos dispositivos para conferência da biometria existentes possuem características em comum e cada qual possui sua particularidade, mas a escolha da íris se deu pelo fato desta ser uma parte do corpo humano que é dificilmente alterada por fatores externos, além do fato, de câmeras digitais serem amplamente utilizadas no âmbito social. Esses dispositivos estão presentes nas mais diversas aplicações, seja na área da segurança pública, monitoramento de trânsito, ambientes indústriais, etc. A implementação de câmeras e sensores em automóveis está em voga entre os fabricantes destes, esses dispositivos cumprem funções estratégicas e funcionais para aumentar a segurança, conforto e dirigibilidade em um automóvel. Os dispositivos biométricos em automóveis também são amplamente utilizados e cumprem um papel importante para aumentar a segurança da população e reduzir os altos índices de furtos e roubos de veículos no Brasil.

Tendo em vista os problemas com falta de segurança e bem-estar da população, a tecnologia auxilia e promove a avaliação e tomada rápida de decisões em casos de roubos e furtos ou em situações de estresse oriundos de incidentes repentinos e indesejáveis para o ser humano. Os sistemas de reconhecimento e

verificação biométrica cumprem uma função fundamental para uma análise rápida em casos onde a percepção humana não se faz presente, substituindo com excelência a ação humana para evitar incidentes.

Foi possível com o advento deste trabalho descobrir e aprimorar conhecimentos na área de PDI, sendo assim esses fundamentos e conceitos são amplamente difundidos nas referências bibliográficas utilizadas neste trabalho. O tratamento de informações em forma de imagem é um campo muito vasto da computação e engenharia, utilizando-se do conceito e aplicação de filtros digitais, histogramas, transformadas e demais operações matemáticas, as quais só foram possíveis com os estudos na área e o avanço da tecnologia como um todo. Outra parte importante do conhecimento adquirido foi o uso da linguagem de programação *Python* e de suas bibliotecas, pois proporcionou flexibilidade e otimização de processos e tarefas no ambiente computacional.

No contexto das ferramentas computacionais, foi necessário a utilização de técnicas de aprendizado de máquina, as quais estão amplamente difundidas em diversas aplicações de cunho tecnológico em meio a sociedade. Técnicas avançadas de aprendizado de máquina e inteligência artificial estão cada vez mais difusas na engenharia e computação e fazem parte da vida do ser humano em todos os aspectos possíveis, por muitas vezes realizando tarefas rotineiras e complexas para o ser humano, mas que em poucos segundos são exequíveis por máquinas. O aprendizado de máquina foi utilizado neste trabalho para a verificação de usuários do sistema, neste caso exemplificando e validando um potencial enorme do uso deste tipo de ciência computacional para a análise de dados e tomada de decisões.

Os sistemas de reconhecimento biométrico estão cada vez mais precisos e dificilmente violáveis, graças ao estudo e desenvolvimento de técnicas e tecnologias por diversos cientistas e engenheiros pelo mundo, que trazem inúmeros benefícios para a sociedade e proporcionam mais segurança, conforto e bem-estar para as pessoas.

5.1 TRABALHOS FUTUROS

A principal sugestão a ser citada consiste no desenvolvimento de um sistema de identificação criminal do infrator que acessa o veículo que pode ser atribuída a um banco de dados de infratores e criminosos que serão rapidamente identificados.

Nesse sentido, com a finalidade de proporcionar um ambiente seguro e economicamente viável, para o usuário final, será priorizada a proposta de uma solução simples, de baixo custo e que possibilite agregar tecnologias de leitura biométrica em um sistema embarcado.

As sugestões para trabalhos futuros residem na concepção do modelo e nos códigos apresentados neste trabalho que podem, certamente, ser aprimorados em termos de exequibilidade, tempo e otimização computacional. As métricas que avaliaram o sistema apresentaram um desempenho satisfatório do modelo, porém medianos em termos de produto para a comercialização, desta forma recomenda-se o estudo mais aprofundado da melhor técnica de aprendizado de máquina e conjunto de recursos computacionais para a melhoria da precisão do modelo, assim como a implementação de um bom modelo de tomada de decisão tendo como base os resultados advindos do classificador SVM.

O desenvolvimento e concepção deste modelo de reconhecimento de íris é deixado como legado para o âmbito acadêmico e da sociedade, trazendo conceitos importantes para o aprendizado em diversas áreas da tecnologia.

REFERÊNCIAS

AUTO ESPORTE. **Chevrolet revela carro elétrico autônomo com visual futurista**. Disponível em: <http://g1.globo.com/carros/noticia/2015/04/chevrolet-revela-carro-eletrico-autonomo-com-visual-futurista.html> Acesso em: 17 ago. 2019.

AUTOS NOVOS. **Tecnologia de reconhecimento facial chega para os carros**. Disponível em: <https://www.autosnovos.com/tecnologia-de-reconhecimento-facial-chega-para-o-carros/> Acesso em: 16 jul. 2019.

BACKES, A.R. **Introdução à visão computacional usando MATLAB**. Rio de Janeiro. Alta Books, 2016.

BARELLI, F. **Introdução à Visão Computacional: ma abordagem prática com Python e OpenCV.** Editora Casa do Código, 2018.

BODADE, R.M. et TALBAR, S.N. **Iris analysis for biometric recognition systems.** Londres. Springer, p. 8, 2014.

BOLLE. R.M et al. **Guide to biometrics**. Springer Professional Computing, 1st edition. 2004.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV: Computer Vision with the OpenCV Library**. O'Reilly, 2008, p.1.

BRANDEWINDER, M. **Machine learning projects for .NET developers**. Apress, 2015.

BRASIL. Decreto-Lei 2.848, de 07 de dezembro de 1940. **Código Penal**. Diário Oficial da União, Rio de Janeiro, 1940.

BRASIL ECONÔMICO. **Biometria nos veículos: veja como a experiência de transporte será impactada**. Disponível em:<https://economia.ig.com.br/2017-08-03/biometria-carro-futuro.html> Acesso em: 10 jul. 2019.

BURGER, W. et BURGE, M.J. **Principles of digital image processing**. Londres, Springer, 2009.

CANEDO, J. A. **Visão geral de um sistema biométrico**. Fórum Biometria, 2010. Disponível em: <http://www.forumbiometria.com/fundamentos-de-biometria/129-visao-geral-de-um-sistema-biometrico.html> Acesso em: 12 jul. 2019.

CANNY, John. **A computational approach to edge detection**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Washington, 1986.

CARVALHO, M.A.G. **Processamento digital de imagens – ST061**. Universidade Estadual de Campinas (UNICAMP). Centro Superior de Educação Tecnológica (CESET). Campinas, 2014.

CERQUEIRA, D.R.C. **Trajetórias individuais, criminalidade e o papel da educação**. Boletim de análise político-institucional, n° 9, p. 36, Jan a Jun 2016.

COURROL, L.C et PRETO, A. O. **Óptica geométrica**. São Paulo, Editora Unifesp, 2011.

DAUGMAN, J.G. **High confidence visual recognition of persons by a test of statistical independence**. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, nº 11, pp. 1148-1161, 1993.

_____. **How Iris Recognition Works**. IEEE Transactions on Circuits and Systems for Video Technology, vol 14, n° 1, pp. 21-30, 2004.

DREAMSTIME. **Fotos de stock seleccionadas por designers**. Disponível em: <https://pt.dreamstime.com/imagem-de-stock-royalty-free-biometria-image16799056> Acesso em: 12 jul. 2019.

DUDA, R. O. et HART, P. E. **Use of the Hough Transformation to detect lines and curves in pictures**. California, Communications of the ACM, 15:11–15, 1972.

ESQUEF, I.A. et al. **Processamento digital de Imagens**. Centro Brasileiro de Pesquisas Físicas (CBPF) e Universidade Estadual Fluminense (UENF), 2003. Disponível em: <http://www.cbpf.br/cat/pdsi/pdf/cap3webfinal.pdf> Acesso em: 26 ago. 2019.

FÓRUM BRASILEIRO DE SEGURANÇA PÚBLICA. **Anuário Brasileiro de Segurança Pública 2019.** 13ª Edição, 2019. Disponível em:< http://www.forumseguranca.org.br/wp-content/uploads/2019/10/Anuario-2019-FINAL_21.10.19.pdf> Acesso em: 3 nov. 2019.

GANG, Wu; CHANG, Edward Y; PANDA, Navneet**. Formulating distance functions via the kernel trick**. Knowledge Discovery and Data Mining, 2005, p. 704.

GARRETA, Raúl; MONCHECCHI, Guillermo. **Learning scikit-learn: Machine Learning in Python**. PACKT Publishing, 2013, p.6.

GOMES, O.F.M**. Processamento e análise de imagens aplicados à caracterização automática de materiais**, Dissertação de Mestrado, PUC/Rio, Rio de Janeiro, 2001.

_____**. Processamento digital de imagens com FIGI/ImageJ.** Universidade Federal de Minas Gerais. Centro de Microscopia, Workshop MÊS 2018. Disponível em: < http://www.microscopia.ufmg.br/images/downloads/Workshop2018_Curso_PI_FIJI.pdf> Acesso em: 10 ago. 2019.

HASTIE, Trevor; TIBSHIRANI, Robert, FRIEDMAN, Jerome. **The elements of statistical Learning**.

HAYKIN, Simon. **Neural networks and machine learning**. Pearson, 2008, p. 269-270.

HEDGECOE, John. **The Book of Photography: Simple Techniques for Taking Better Pictures**. DK Adult, 2005, p. 14.

HUGOO. D. **Biometria: confira quais são as principais vantagens sobre outros métodos de autenticação**. Security Brasil, São Paulo, 2016. Disponível em: < http://revistasecurity.com.br/blog/especialista-narra-vantagens-da-biometria-sobre-outros-metodos-de-autenticacao/> Acesso em: 10 jul. 2019.

IPEA. INSTITUTO DE PESQUISA ECONÔMICA APLICADA. **Atlas da Violência**. Rio de Janeiro, Fórum Brasileiro de Segurança Pública, 2018.

JAIN, A.K. et al. **Handbook of Biometrics**. New York: Springer, 2008, 72 p.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An introduction to statistical learning with applications in R**. Springer, 2003, p. 345.

JILLELA, R.R. et ROSS, A. Pattern Recognition Letters. **Segmenting iris images in the visible spectrum with applications in mobile biometrics**. Elsevier. Volume 57, p. 4-16, 2015.

JUNIOR. R. M. C. Olhos nos olhos – ensinando o computador a reconhecer pessoas. **Ciência hoje**. São Paulo, v. 29, n. 174, p. 25-29, 2001.

KRUEGER, M.L. et al. Um estudo de caso sobre autenticação biométrica em dispositivos móveis. **Computer on the beach – artigos completos.** Universidade do Vale do Itajaí (UNIVALI), Itajaí-SC, 4ª Ed, p. 98-107, 2013.

MINISTÉRIO DA JUSTIÇA. Secretaria Executiva. **RT** - **Estudo sobre técnicas de fusão em multibiometria e o desempenho de sistemas multimodais.** Versão 0.1. Universidade de Brasília – UnB. Brasília, 2015.

LEE, C. J.et WANG, S. D. **Fingerprint feature extraction using Gabor filters**. Electronic Letters, v. 35, n. 4, p. 288-290, 1999.

LORENA, Ana Carolina; DE CARVALHO, André C. P. L. F.. **Uma introdução às support vector machines**. Revista de Informática Teórica e Aplicada, UFRGS, 2007, p. 54-62.

MARTINS, S.B. **Introdução ao processamento digital de imagens** Instituto de Computação da Universidade Estadual de Campinas, 2019. Disponível em < https://www.ic.unicamp.br/~ra144681/misc/files/ApostilaProcDeImagesParteI.pdf > Acesso em: 23 ago. 2019.

MINISTÉRIO DA JUSTIÇA E SEGURANÇA PÚBLICA. **Sistema Nacional de Informações de Segurança Pública, Prisionais e sobre Drogas – Sinesp**. Brasília, 2019.

MOBYLE ID WORLD. **Gentex Shows Off Connected Car, Biometric Tech at CES 2019**. Disponível em: https://mobileidworld.com/gentex-connected-car-biometric-tech-ces-2019-801091/ Acesso em: 23 jul. 2019.

NEWMAN. R. **Security and access control using biometric technologies**. Boston, Course Technology, 2010.

NUMPY. **Array manipulation routines**. Disponível em: < https://docs.scipy.org/doc/numpy/reference/generated/numpy.copyto.html> Acesso em: 7 nov. 2019.

OPENCV. **Canny edge detector**. OpenCV Tutorials, 2019. Disponível em: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html.> Acesso em: 6 nov. 2019.

_____. **OpenCV**. Disponível em: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html.> Acesso em: 6 nov. 2019.

_____. **Feature Detection**. Disponível em: <https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles#houghcircles> Acesso em: 6 nov. 2019.

_____. **Smoothing Images**. Disponível em: <https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html> Acesso em: 6 nov. 2019.

_____. **Image Filtering**. Disponível em: <https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html> Acesso em: 6 nov. 2019.

OPTICS, Edmund. **Cameras**. Disponível em: < https://www.edmundoptics.com/c/cameras/1012/> Acesso em: 5 set. 2019.

PANDA, M. et al. **Big data analytics: a social network approach**. Flórida: CRC Press, 2009.

PETROVSKA-DELACRÉTAZ, D. et al. **Guide to biometric reference systems and performance evaluation**. Londres. Springer Science & Business, p. 30, 2009.

PILLOW. **Image Module**. Disponível em: < https://pillow.readthedocs.io/en/3.1.x/reference/Image.html> Acesso em: 7 nov. 2019.

PINOCHET, Luis H.C. Tecnologias emergentes. In: PINOCHET, 7. **Tecnologia da informação e comunicação**. Rio de Janeiro: Elsevier, 2014, p. 230-286.

QIDWAI, U et CHEN, C. H. **Digital image processing: an algorithmic approach with MATLAB**. Flórida: CRC Press, 2009.

ROMANI, B. **Segurança e o mundo digital.** JORNAL FOLHA DE S. PAULO. Disponível em: < http://temas.folha.uol.com.br/futuro-digital/seguranca-e-o-mundo-digital/selfie-vira-senha-para-cartao-de-credito-com-uso-de-biometria.shtml> Acesso em: 18 ago. 2019.

ROSS, A et JAIN, A. K. In: EUROPEAN SIGNAL PROCESSING CONFERENCE. **Multimodal biometrics: an overview**. Vienna, 2004.

ROSS, J. **Alarmes**. Rio de Janeiro: Antenna Edições Técnicas, 2008.

SAEED, K; NAGASHIMA, T. **Biometrics and Kansei Engeneering**. New York: Springer Science & Business Media. 2012.

SALAMI, Momoh; ALI, Hasimah. **Iris Recognition System Using Support Vector Machines**. Biometric Systems, Design and Applications, 2011, p.175, 180.

SANTHI, V et al. **Emerging technologies in intelligent applications for image and video processing**. EUA. IGI Global, 2016.

SANTOS, A. **Gerenciamento de Identidades**. Rio de Janeiro: Brasport, 2007.

SCHÖLKOPF, Bernhard; SMOLA, Alexander J.. Learning with kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, 2002, p.7.

SCIKIT-LEARN. **Model Selection**. Disponível em: < https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html> Acesso em: 7 nov. 2019.

_____. **Support Vector Machines**. Disponível em: < https://scikit-learn.org/stable/modules/svm.html#classification > Acesso em: 7 nov. 2019.

SCIPY. **Multidimensional image processing**. Disponível em: < https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.sobel.html> Acesso em: 7 nov. 2019.

SHALEV-SCHWARTZ, Shai; BEN-DAVID, Shai. **Understanding machine learning theory algorithms**. Cambridge University Press, 2014, p. 2.

VANPUTTE. C. L. et al. **Anatomia e fisiologia de SEELEY**. São Paulo, AMGH Editora Ltda, 2016.

VIANA, A. **Direito penal: teoria geral do fato punível e das sanções penais.** Curitiba: Juruá, 2008, p. 27).

WAYMAN. J. L. et al. **Biometric systems**. London: Springer Science & Business Media: 2005.

WHITE, Ron. **Click! the no nonsense guide to Digital Camera**. McGraw-Hill, 2003, p.3

WILDES, R.P.**Iris Recognition: An Emerging Biometric Technology**. Proceedings of the IEEE, vol. 85, no. 9, 1997.

ZAIN, J.M. et al. **Software engineering and computer systems**. Londres. Springer, p. 705, 2011.

ANEXO A – NÚMERO E TAXA DE ROUBO E FURTO DE VEÍCULOS NO BRASIL

TABELA 12

Crimes violentos não letais contra o patrimônio: roubo e furto de veículos [1]
Brasil e Unidades da Federação – 2017-2018

Brasil e Unidades da Federação	Roubo de veículo					Furto de veículo					Roubo e Furto de Veículo				
	Ns. Absolutos		Taxas [2]		Variação	Ns. Absolutos		Taxas [2]		Variação	Ns. Absolutos		Taxas [2]		Variação
	2017 [3]	2018	2017	2018	(%)	2017 [3]	2018	2017	2018	(%)	2017 [3]	2018	2017	2018	(%)
Brasil	**280.392**	**247.148**	**292,5**	**245,3**	**-16,1**	**265.504**	**243.808**	**277,0**	**242,0**	**-12,6**	**551.163**	**490.956**	**567,7**	**487,3**	**-14,2**
Acre	1.415	1.220	536,3	439,1	-18,1	793	618	300,6	222,4	-26,0	2.208	1.838	836,9	661,6	-20,9
Alagoas	3.133	3.475	395,7	416,3	5,2	1.195	1.417	150,9	169,7	12,5	4.328	4.892	546,6	586,0	7,2
Amapá	267	340	143,3	174,3	21,7	721	649	387,0	332,8	-14,0	988	989	530,3	507,1	-4,4
Amazonas	4.597	3.080	542,8	348,8	-35,7	3.542	2.258	418,2	255,7	-38,9	8.139	5.338	961,0	604,5	-37,1
Bahia	13.506	13.226	340,6	319,5	-6,2	5.510	5.247	139,0	126,8	-8,8	19.016	18.473	479,6	446,3	-6,9
Ceará	11.133	9.319	368,0	296,0	-19,6	4.997	4.184	165,2	132,9	-19,5	16.130	13.503	533,2	428,9	-19,6
Distrito Federal	4.851	3.986	276,9	219,9	-20,6	5.726	5.283	326,9	291,5	-10,8	10.577	9.269	603,8	511,4	-15,3
Espírito Santo	6.079	4.286	325,3	221,3	-32,0	4.709	4.049	252,0	209,0	-17,0	10.788	8.335	577,3	430,3	-25,5
Goiás	15.633	11.272	414,3	288,3	-30,4	12.284	11.272	325,6	288,3	-11,4	27.917	22.544	739,9	576,7	-22,1
Maranhão	4.477	4.093	276,5	241,2	-12,8	3.130	2.980	193,3	175,6	-9,2	7.607	7.073	469,9	416,9	-11,3
Mato Grosso	2.585	2.288	131,5	110,0	-16,4	2.817	2.438	143,3	117,2	-18,2	5.402	4.726	274,8	227,1	-17,4
Mato Grosso do Sul	907	794	59,7	50,2	-16,0	3.640	3.702	239,7	233,8	-2,5	4.547	4.496	299,5	284,0	-5,2
Minas Gerais	13.015	9.504	121,5	84,9	-30,1	25.471	21.288	237,8	190,2	-20,0	38.486	30.792	359,3	275,1	-23,4
Pará	8.593	6.694	448,0	332,4	-25,8	5.079	4.297	264,8	213,4	-19,4	13.672	10.991	712,8	545,7	-23,4
Paraíba [4]	...	3.834	...	296,4	...	...	1.566	...	121,1	...	5.267	5.400	425,5	417,4	-1,9
Paraná	11.368	7.874	155,0	104,0	-32,9	19.354	17.620	263,9	232,7	-11,8	30.722	25.494	419,0	336,7	-19,6
Pernambuco	19.701	15.522	677,8	515,6	-23,9	6.860	5.534	236,0	183,8	-22,1	26.561	21.056	913,8	699,4	-23,5
Piauí	3.114	3.884	273,0	324,7	19,0	2.563	3.214	224,7	268,7	19,6	5.677	7.098	497,6	593,4	19,2
Rio de Janeiro	54.366	52.097	831,4	774,6	-6,8	15.708	15.794	240,2	234,8	-2,2	70.074	67.891	1.071,6	1.009,4	-5,8
Rio Grande do Norte	6.992	6.982	564,7	540,9	-4,2	1.329	1.043	107,3	80,8	-24,7	8.321	8.025	672,1	621,7	-7,5
Rio Grande do Sul	17.881	16.127	261,0	227,8	-12,7	16.919	14.430	247,0	203,9	-17,4	34.800	30.557	507,9	431,7	-15,0
Rondônia	1.811	1.816	192,4	184,4	-4,2	3.120	3.103	331,5	315,0	-5,0	4.931	4.919	523,9	499,4	-4,7
Roraima	358	529	170,0	241,2	41,9	790	636	375,2	290,0	-22,7	1.148	1.165	545,3	531,3	-2,6
Santa Catarina	3.014	2.214	60,9	43,0	-29,5	11.988	9.082	242,3	176,3	-27,3	15.002	11.296	303,3	219,2	-27,7
São Paulo	67.964	58.970	241,5	202,9	-16,0	104.829	99.346	372,5	341,9	-8,2	172.793	158.316	614,1	544,8	-11,3
Sergipe	2.815	2.639	379,8	341,7	-10,0	886	776	119,5	100,5	-16,0	3.701	3.415	499,4	442,1	-11,5
Tocantins	817	1.083	123,5	156,9	27,1	1.544	1.982	233,3	287,2	23,1	2.361	3.065	356,8	444,1	24,5

Fonte: Secretarias Estaduais de Segurança Pública e/ou Defesa Social; Fórum Brasileiro de Segurança Pública.
(...) Informação não disponível.
(1) Os dados informados correspondem ao volume de ocorrências policiais registradas.
(2) Taxas por 100 mil veículos, calculadas a partir da frota de veículos informada pelo Departamento Nacional de Trânsito (Denatran) em dezembro/2017 e dezembro/2018.
(3) Atualização das informações publicadas no Anuário Brasileiro de Segurança Pública, ano 12, 2018.
(4) Até 2017, as ocorrências de roubo e furto de veículos eram computadas conjuntamente pelo sistema de inclusão de bloqueios no DETRAN, portanto, só há dados desagregados para as duas categorias a partir de 2018.

APÊNDICE A – ESPECIFICAÇÕES TÉCNICAS DO MÓDULO RASPBERRY PI 3 MODELO B+

Especificações Técnicas:

- Quad Core 1.2GHz Broadcom BCM2837 64bit Cpu
- Memória Ram: 1GB
- BCM43438 WiFi e Bluetooth Low Energy (Ble) on board
- Gpu Broadcom VideoCore Iv
- 4 Portas USB2.0 com saída de até 1,2A
- Conector expandido de 40 pinos Gpio
- Saída de Video/Audio Out via conector de 4 polos 3,5mm, Hdmi, ou Raw Lcd (Dsi)
- Armazenamento: microSD
- 10/100 Ethernet (RJ45)
- Tensão de operação: Micro Usb socket 5V/2A
- Dimensões: 85 x 56 x 17mm
- Suporte a Windows 10 Lot, Debian Gnu/Linux, Fedora, Arch Linux, Risc Os entre outros

Periféricos:

- 4 portas Usb
- Gpio de 40 pinos
- Full Hdmi
- Ethernet 10/100 (RJ45)
- Saída de vídeo via Hdmi, Composite (Pal e Ntsc) ou Raw Lcd (Dsi)
- Saída de áudio via conector de 3,5mm
- Camera interface (Csi)
- Slot MicroSD
- Video Core Iv 3D graphics core
- Necessidades de alimentação: 5V / 1,8 Amp via MicroUSB ou conector Gpio

Fonte: www.americanas.com.br/produto/33998320/kit-basico-raspberry-pi-3-pi3-16gb-case-black

APÊNDICE B – FICHA TÉCNICA DA CÂMERA RASPBERRY PI 5MP

Especificações Técnicas:

- Câmera Raspberry Pi 5MP
- Sensor OV5647
- Resolução: 5MP
- CCD Size: 1/4 polegadas
- Abertura (F): 1.8
- Comprimento focal: 3.6mm (ajustável)
- Ângulo de visão (Diagonal): 75.7 graus
- Dimensões: 25 x 24 x 23,5 mm
- Cabo flat: 15cm

Fonte: www.filipeflop.com/produto/camera-compativel-raspberry-pi-5mp/

APÊNDICE C – PLANILHA DE CUSTOS

Descrição	Valor
1) Curso de linguagem de programação Python 3, com duração de 45 horas	US$ 379.95
2) Curso prático de introdução, configurações iniciais e programação utilizando o Raspberry Pi 3	US R$ 6.17
3) Módulo Raspberry Pi 3 modelo B+ original, com manual, fonte 5V/3A (bivolt), par de dissipadores compatíveis, Case Black e cartão sandisk	US$ 83.00
4) Câmera Raspberry Pi 5MP	US$ 37.21
5) Produção do protótipo em Medium Density Fiberboard (MDF) com revestimento, espessura de 15mm com painel em PVC	US$ 28.64
6) Compra de componentes eletroeletrônicos para conexões e montagens	US$ 11.93
Custo Total	**R$ 546.95**

*Valores atualizados em 10 de novembro de 2019, em Curitiba/PR.

APÊNDICE D – CÓDIGO EM LINGUAGEM DE PROGRAMAÇÃO PYTHON PARA SEGMENTAÇÃO DA ÍRIS

```
##-------------------------------------------------------------------------
##  Importação de bibliotecas
##-------------------------------------------------------------------------
from PIL import Image
import cv2
from scipy import ndimage
import numpy as np

import warnings
warnings.filterwarnings("ignore")


"""
   Descrição: Função que faz o processamento da imagem dos olhos
   Parâmetro: imagem que é capturada através da câmera
   Retorno: Nenhum
"""
def segmentacao_iris(imagem):

   # Variável que armazena a imagem original convertida para escala de cinza
   imagem_escala_cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)

   # Histograma da imagem original
   histograma, intervalos = np.histogram(imagem_escala_cinza, 5, [0, 255])


   """
         Matriz de limiar de histograma para pontos da imagem que estão abaixo
     de 255/5
   """
   matriz_limiar_histograma = imagem_escala_cinza < intervalos[1]


   """
      Cópia da matriz de valores binários de limiar do histograma
      Converte a matriz de valores binários para valores inteiros
      de 8 bits não sinalizados
   """
   imagem_binaria = matriz_limiar_histograma.astype(np.uint8)


   """
      Converte para branco (valor digital 255) valores binários que
      estão abaixo do limiar
   """
   imagem_binaria *= 255


   """
      Reconhece círculos na imagem que correspondem a pupíla
      Variável armazena localização do centro do círculo (X Y em pixels)
      Também armazena o raio do círculo em pixels
   """
   circulos_pupila = cv2.HoughCircles(imagem_binaria, cv2.HOUGH_GRADIENT, 2, 80, param1=50,
param2=8,
                                      minRadius=30, maxRadius=150)
```

```
    # Arrendonda e converter para o vetor para valores de 16 bits
    circulos_pupila = np.uint16(np.around(circulos_pupila))

    # Variável que armazena o tamanho (em pixels) do recorte dos olhos
    area_olhos = (650, 650)


    """
        Desenha um círculo preto delimitado pela pupíla para diminuir a
        influência da iluminação dos leds
    """
    cv2.circle(imagem, (circulos_pupila[0][0][0], circulos_pupila[0][0][1]), circulos_pupila[0][0][2], (0, 0,
0), -1)

    # Armazena medidas de recorte dos olhos
    medidas_recorte_olhos = (max(0, circulos_pupila[0][0][1] - area_olhos[0] // 2),
min(imagem.shape[0], circulos_pupila[0][0][1] + area_olhos[0] // 2),
                             max(0, circulos_pupila[0][0][0] - area_olhos[1] // 2), min(imagem.shape[1],
circulos_pupila[0][0][0] + area_olhos[1] // 2))

    # Define novas medidas para imagem dos olhos
    recorte_imagem_olhos = imagem[medidas_recorte_olhos[0]:medidas_recorte_olhos[1],
                           medidas_recorte_olhos[2]:medidas_recorte_olhos[3]]

    # Variável que armazena o valor do raio da pupíla
    raio_pupila = circulos_pupila[0][0][2]

    # Variável que armazena o valor da coordenada x do centro da pupila
    x_pupila = circulos_pupila[0][0][0]

    # Variável que armazena o valor da coordenada x do centro da pupila
    y_pupila = circulos_pupila[0][0][1]

    # Filtro para ruídos na imagem ou suavização da imagem
    recorte_olhos_imagem_filtrada = cv2.blur(recorte_imagem_olhos, (7,7))

    recorte_olhos_imagem_filtrada_int32 = recorte_olhos_imagem_filtrada.astype('int32')
    # Orientação horizontal
    dx = ndimage.sobel(recorte_olhos_imagem_filtrada_int32, 0)
    # Orientação vertical
    dy = ndimage.sobel(recorte_olhos_imagem_filtrada_int32, 1)
    # Magnitude
    matriz_filtro_sobel = np.hypot(dx, dy)
    # Normalização
    matriz_filtro_sobel *= 255 / np.max(matriz_filtro_sobel)

    # Converte matriz para profundidade de 8 bits
    matriz_filtro_sobel = np.uint8(matriz_filtro_sobel)

    # Converter imagem colorida para escala de cinza
    imagem_filtro_sobel_int8 = cv2.cvtColor(matriz_filtro_sobel, cv2.COLOR_RGB2GRAY)

    """
        Reconhece círculos na imagem que correspondem a íris
        Variável armazena localização do centro do círculo (X Y em pixels)
        Também armazena o raio do círculo em pixels
    """
    circulos_iris = cv2.HoughCircles(imagem_filtro_sobel_int8, cv2.HOUGH_GRADIENT, 1, 30,
param1=40, param2=30,
```

```
                minRadius=80, maxRadius=120)

    # Armazena as coordenadas da pupila
    # Coordenada X
    x_aux = x_pupila
    # Coordenada Y
    y_aux = y_pupila

    # Armazena medidas da íris
    # Coordenada X
    medida_iris_x = 0
    # Coordenada Y
    medida_iris_y = 0
    # Raio da íris
    raio_iris_aux = circulos_iris[0][0][1]


"""
        Laço de repetição para pegar as coordenadas da circuferência da pupila que mais se aproximam
        das coordenadas da pupila
"""
    for i in range(circulos_iris.shape[1]):

        # Armazena a diferença entre o centro da íris e da pupila
        # Coordenada X
        diferenca_centro_x = circulos_iris[0][i][0] - x_pupila
        # Coordenada Y
        diferenca_centro_y = circulos_iris[0][i][1] - y_pupila

        # Checa na coordenada X se a diferença atual é o menor
        if((np.abs(diferenca_centro_x) < x_aux)):
            # Recebe valor absoluto da diferença em X
            x_aux = np.abs(diferenca_centro_x)
            # Armazena medida da coordenada X do centro da pupila
            medida_iris_x = circulos_iris[0][i][0]
        # Checa na coordenada Y se a diferença atual é o menor
        elif((np.abs(diferenca_centro_y) < y_aux)):
            # Recebe valor absoluto da diferença em Y
            y_aux = np.abs(diferenca_centro_y)
            # Armazena medida da coordenada Y do centro da pupila
            medida_iris_y = circulos_iris[0][i][1]
        # Checa se o raio da íris é o menor
        elif(circulos_iris[0][i][2] < raio_iris_aux):
            # Armazena o raio da íris
            raio_iris_aux = circulos_iris[0][i][2]
        # Checa se a diferença das coordenadas do centro é nula
        elif((diferenca_centro_x == 0) and (diferenca_centro_y == 0)):
            # Armazena medidas da coordenada em X
            medida_iris_x = np.abs(circulos_iris[0][i][0])
            # Armazena medidas da coordenada em Y
            medida_iris_y = np.abs(circulos_iris[0][i][1])
            # Sai do laço de repetição
            break

    # Armazena as coordenadas da íris
    # Coordenada X
    circulos_iris[0][0][0] = medida_iris_x
    # Coordenada Y
    circulos_iris[0][0][1] = medida_iris_y
```

```
    # Raio da íris
    circulos_iris[0][0][2] = raio_iris_aux + 20

    # Arrendonda e converte o vetor para valores de 16 bits
    circulos_iris = np.uint16(np.around(circulos_iris))

    # Armazena valores da imagem (altura, largura e número de canais)
    altura, largura, canais = recorte_imagem_olhos.shape

    """
        Cria máscara (matriz com zeros com dimensões da imagem recortada anteriormente)
    """
    mascara = np.zeros((altura, largura), dtype=np.uint8)

    """
        Laço de repetição para construção dos círculos de acordo com as medidas
        obtidas atráves da transformada de Hough para a íris
    """

    # Desenha máscara
    cv2.circle(mascara, (circulos_iris[0][0][0], circulos_iris[0][0][1]), circulos_iris[0][0][2], (255, 255,
255), thickness=-1)

    # Armazena medidas da máscara
    medidas_mascara_iris = (max(0, circulos_iris[0][0][1] - 200 // 2), min(mascara.shape[0],
circulos_iris[0][0][1] + 200 // 2),
                            max(0, circulos_iris[0][0][0] - 200 // 2), min(mascara.shape[1],
circulos_iris[0][0][0] + 200 // 2))

    # Define novas medidas para imagem dos olhos
    mascara_recorte_iris = mascara[medidas_mascara_iris[0]:medidas_mascara_iris[1],
                           medidas_mascara_iris[2]:medidas_mascara_iris[3]]

    # Define as medidas para recorte da imagem da íris
    recorte_iris = (max(0, circulos_iris[0][0][1] - 200 // 2), min(recorte_imagem_olhos.shape[0],
circulos_iris[0][0][1] + 200 // 2),
                    max(0, circulos_iris[0][0][0] - 200 // 2), min(recorte_imagem_olhos.shape[1],
circulos_iris[0][0][0] + 200 // 2))

    # Define novas medidas para imagem da íris
    imagem_recorte_iris = recorte_imagem_olhos[recorte_iris[0]:recorte_iris[1],
                          recorte_iris[2]:recorte_iris[3]]

    # Converte o recorte da imagem da íris para escala de cinza
    recorte_iris_escala_cinza = cv2.cvtColor(imagem_recorte_iris, cv2.COLOR_BGR2GRAY)

    # Armazena matriz de íris com zeros
    iris_imagem = np.zeros(recorte_iris_escala_cinza.shape, dtype=np.uint8)

    # Realiza cópia do máscara da íris na imagem da íris, onde existe nível lógico alto na máscara
    np.copyto(iris_imagem, recorte_iris_escala_cinza, where=mascara_recorte_iris.astype(np.bool))


    # Variável que armazena a matriz convertida para imagem
    imagem_iris_normalizada = Image.fromarray(iris_imagem)


    # Variável que armazena a recortagem do primeiro segmento de íris
    recorte_iris_fracao_parte1 = (6, iris_imagem.shape[0] / 2 - 35, 31,
                                  iris_imagem.shape[0]/2 + 35)
```

```
    # Variável que armazena a recortagem do segundo segmento de íris
    recorte_iris_fracao_parte2 = (iris_imagem.shape[1] / 2 + 69, iris_imagem.shape[0] / 2 - 35,
iris_imagem.shape[1] - 6,
                                  iris_imagem.shape[0] / 2 + 35)

    # Variável que armezana a imagem recortada da íris "normalizada"
    imagem_iris_recortada_parte1 = imagem_iris_normalizada.crop(recorte_iris_fracao_parte1)

    # Variável que armezana a imagem recortada da íris "normalizada"
    imagem_iris_recortada_parte2 = imagem_iris_normalizada.crop(recorte_iris_fracao_parte2)

    # Concatena os dois segmentos de íris em uma imagem
    imagem_iris_recortada_total = np.hstack((imagem_iris_recortada_parte1,
imagem_iris_recortada_parte2))

    # Converte a imagem em uma matriz
    matriz_iris_recortada = np.array(imagem_iris_recortada_total)

    # Segmento de íris para extração de característica
    return matriz_iris_recortada
```

APÊNDICE E – CÓDIGO EM LINGUAGEM DE PROGRAMAÇÃO PYTHON PARA INSCRIÇÃO DO BANCO DE DADOS CASIA

```
##-------------------------------------------------------------------------
##  Importação de bibliotecas
##-------------------------------------------------------------------------
import argparse, os
from glob import glob
from tqdm import tqdm
from time import time
from multiprocessing import cpu_count, Pool
from extracao_caracteristica_modelo import *
from reconhecimento_iris import *


"""
        Descrição: Função executada de forma pararela para inscrição dos     dados do banco de dados de íris
        Parâmetro: arquivo a ser inscrito
        Retorno: Nenhum
"""
def pool_func(arquivo):
        # Argumentos do sistema
        argumentos = argumentos_modelo()

        # Armazena iris segmentada
        iris_segmentada = segmentacao_iris(cv2.imread(arquivo))

        # Armazena nome do arquivo
        nome_base = os.path.basename(arquivo)

        # Armazena arquivo
        saida_arquivo = os.path.join(argumentos.diretorio_temporario, "%s.jpg" % (nome_base))

        # Grava arquivo no diretório específicado
        cv2.imwrite(os.path.join(saida_arquivo), iris_segmentada)


"""
   Descrição: Inscrição de banco de dados de íris
   Parâmetro: Nenhum
   Retorno: Nenhum
"""
def inscricao_modelo_casia1():
        #       Argumentos gerais para o sistema
        argumentos = argumentos_modelo()

        ##-------------------------------------------------------------------------
        ##  Execução
        ##-------------------------------------------------------------------------

        # Armazena tempo de inicio da inscrição do banco de dados
        inicio = time()

        # Checa a existência do diretório temporário
        if not os.path.exists(argumentos.diretorio_temporario):
                print("Criando diretório...", argumentos.diretorio_temporario)
                os.makedirs(argumentos.diretorio_temporario)
```

```
        # Percorre diretório de banco de dados para inscrição
        for i in range(numero_diretorios(argumentos.diretorio_dado)[1]):

                # Armazena todas as imagens .jpg do banco de dados
                arquivos = glob(os.path.join(argumentos.diretorio_dado, "{}/".format(i+1)+"*.jpg"))

                # Printa arquivos do diretório
                print(arquivos)

                # Armazena número de arquivos no diretório
                numero_arquivos = len(arquivos)

                print("Número de arquivos cadastrados:", numero_arquivos)

                print("\nIniciando cadastramento...")

                # Pools: Função paralela para inscrição de modelos
                pools = Pool(processes=argumentos.numero_nucleo)
                for _ in tqdm(pools.imap_unordered(pool_func, arquivos), total=numero_arquivos):
                        pass

                # Armazena tempo final da inscrição do banco de dados
                fim = time()

                print('\nTempo de cadastramento: {} [s]\n'.format(fim - inicio))

        numero_total_arquivos, _ = numero_diretorios(argumentos.diretorio_dado)

        print("\n**************************************************************************")
        print("Número total de arquivos cadastrados:", numero_total_arquivos)
        print("\n**************************************************************************")


"""
   Descrição: Função que calcula o número de diretórios
   Parâmetro: caminho de diretório para banco de dados
   Retorno: Número de diretórios
"""
def numero_diretorios(caminho_diretorio):
        # Armazena número de arquivos e diretórios
        num_arquivos = numero_dir = 0

        # Laço de repetição para cálculo do número de diretórios
        for _, dir_nome, arq_nome in os.walk(caminho_diretorio):
                # Armazena número de arquivos
                num_arquivos += len(arq_nome)
                # Armzena número de diretórios
                numero_dir += len(dir_nome)
        # Retorna número de diretórios
        return num_arquivos, numero_dir
```

APÊNDICE F – CÓDIGO EM LINGUAGEM DE PROGRAMAÇÃO PYTHON PARA EXTRAÇÃO DE CARACTERÍSTICA

```
##-------------------------------------------------------------------------
##  Importação de bibliotecas
##-------------------------------------------------------------------------
import math
from segmentacao_iris import *

import warnings
warnings.filterwarnings("ignore")


"""
    Descrição: Função para construção de expressão matemática coseno
    Parâmetro: x, y e frequencia
    Retorno: Função coseno
"""
def funcao_coseno(x, y, f):
    # Armazena resultado da expressão matemática coseno
    expressao_matematica = np.cos(2*np.pi*f*math.sqrt(x ** 2 + y ** 2))

    # Retorna resultado da expressão matemática coseno
    return expressao_matematica


"""
    Descrição: Função para construção do filtro gabor
    Parâmetro: x, y, dx, dy e frequencia
    Retorno: Função do filtro gabor
"""
def filtro_gabor(x, y, dx, dy, f):
    # Armazena resultado da equação do filtro gabor
    equacao_gabor = (1/(2*math.pi*dx*dy)) * np.exp(-0.5 * (x ** 2 / dx ** 2 + y ** 2 / dy ** 2)) *
funcao_coseno(x, y, f)

    # Retorna resultado da expressão do filtro gabor
    return equacao_gabor


"""
    Descrição: Função para construção do filtro gabor
    Parâmetro: imagem do segmento de íris para extração de características
    Retorno: Vetor de características
"""
def extracao_caracteristica(imagem_iris):

    # Parâmetros da equação do filtro gabor
    dx1 = 1.8
    dx2 = 2.1
    dy = 1.3
    f = 1/dy

    # Armazena coeficientes do filtro gabor
    # Vetor1
    caracteristica_vetor1 = np.zeros_like(imagem_iris)

    # Vetor2
    caracteristica_vetor2 = np.zeros_like(imagem_iris)


"""
```

```
        Laço de repetição para construção do kernel com coeficientes do filtro com tamanho de 8 x 8,
        Convoluindo o kernel com a imagem original do segmento de íris
"""
    i = 4
    while i < (imagem_iris.shape[0]-4):
        j = 4
        while j < (imagem_iris.shape[1]-4):
            # Altura do kernel
            altura = 4

            # Largura do kernel
            largura = 4

            # Variável para kernel1 com zeros
            kernel1 = np.zeros((altura*2, largura*2))

            # Variável para kernel2 com zeros
            kernel2 = np.zeros((altura*2, largura*2))

            # Laço de repetição para construção do kernel com coeficientes do filtro com tamanho de 8 x
8,
            for y in range(-altura, altura):
                for x in range(-largura, largura):
                    # Kernel1 com coeficientes do filtro gabor
                    kernel1[x+largura][y+altura] = filtro_gabor(altura+y+1, largura+x+1, dx1, dy, f)

                    # Kernel2 com coeficientes do filtro gabor
                    kernel2[x+largura][y+altura] = filtro_gabor(altura+y+1, largura+x+1, dx2, dy, f)

            # Imagem com coeficientes do filtro gabor
            bloco_olhos = imagem_iris[i-altura:i+altura, j-largura:j+largura]

            # Processo de convolução de imagem original com o kernel construído

            # Primeira convolução
            vetor1 = cv2.filter2D(src=bloco_olhos, kernel=kernel1, ddepth=-1)

            # Vetor1 de características
            caracteristica_vetor1[i-altura:i+altura, j-largura:j+largura] = vetor1

            # Segunda convolução
            vetor2 = cv2.filter2D(src=bloco_olhos, kernel=kernel2, ddepth=-1)

            # Vetor1 de características
            caracteristica_vetor2[i-altura:i+altura, j-largura:j+largura] = vetor2
            j = j+1
        i = i+1

    # Vetor para armazenamento de características
    vetor_caracteristica = []

    # Variável para contagem
    i = 0


"""
        Laço de repetição que converte o vetor de características em um  vetor com média aritmética
        e desvio padrão de cada componente do vetor
"""
    while i < caracteristica_vetor1.shape[0]:
```

```
    j = 0
    while j < caracteristica_vetor1.shape[1]:
        # Armazena vetor de características
        # Primeiro bloco
        bloco1 = caracteristica_vetor1[i:i+8, j:j+8]

        # Segundo bloco
        bloco2 = caracteristica_vetor2[i:i+8, j:j+8]

        # Armazena média aritmética do vetor de características
        # Média do primeiro bloco
        media1 = bloco1.mean()

        # Média do segundo bloco
        media2 = bloco2.mean()

        # Armazena média do vetor1 no vetor de características
        vetor_caracteristica.append(media1)

        # Armazena média do vetor2 no vetor de características
        vetor_caracteristica.append(media2)

        # Armazena desvio padrão do vetor de características
        # Desvio padrão do primeiro bloco
        desvio_padrao1 = bloco1.std()

        # Desvio padrão do segundo bloco
        desvio_padrao2 = bloco2.std()

        # Armazena desvio padrão do vetor1 no vetor de características
        vetor_caracteristica.append(desvio_padrao1)

        # Armazena desvio padrão do vetor2 no vetor de características
        vetor_caracteristica.append(desvio_padrao2)
        j = j + 8
    i = i + 8

# Returna vetor de característica para aprendizado de máquina
return vetor_caracteristica
```

APÊNDICE G – CÓDIGO EM LINGUAGEM DE PROGRAMAÇÃO PYTHON PARA APRENDIZADO DE MÁQUINA E VERIFICAÇÃO DA ÍRIS

```
##-------------------------------------------------------------------------
##  Importação de bibliotecas
##-------------------------------------------------------------------------
import argparse
import scipy.io as sio
import os
import pickle
from inscricao_casia1 import *
from sklearn.svm import SVC
from imutils import paths
from multiprocessing import cpu_count, Pool
from extracao_caracteristica_modelo import *
from time import time
from segmentacao_iris import *
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import *

import warnings
warnings.filterwarnings("ignore")

parser = argparse.ArgumentParser()


"""
    Descrição: Armazena argumentos para o sistema
    Parâmetro: Nenhum
    Retorno: Argumentos do sistema
"""
def argumentos_modelo():
    # --------------------------------------------------------------------------
    #       Argumentos gerais para o sistema
    # --------------------------------------------------------------------------
    parametros = argparse.ArgumentParser()

    # Argumento para diretório que contém os dados (base de dados da íris)
    parametros.add_argument("--diretorio_dado", type=str, default="./CASIA1/",
                            help="Caminho para diretório que contém imagens do banco de dados CASIA1")
    # Argumento para diretório temporário que armazena o resultado do processo de segmentação no
banco de dados
    parametros.add_argument("--diretorio_temporario", type=str, default="./Modelos/CASIA1/",
                            help="Caminho para diretórios que contém modelos")
    # Argumento para número de núcleos do dispositivo
    parametros.add_argument("--numero_nucleo", type=int, default=cpu_count(),
                            help="Número de núcleos usados para inscrição do modelo")

    # Armezena argumentos gerais para o sistema
    argumentos = parametros.parse_args()

    # Retorna argumentos gerais para o sistema
    return argumentos


"""
    Descrição: Armazena argumentos para aprendizado de máquina
    Parâmetro: Nenhum
    Retorno: Argumentos do aprendizado de máquina
"""
def argumentos_modelo_SVM():

    parametros = argparse.ArgumentParser()
```

```
    # Argumento para diretório que contém os dados para treinamento
    parametros.add_argument("--treinamento", type=str, default="./SVM/Treinamento",
                            help="caminho de imagens para treinamento")
    # Argumento para diretório que contém os dados para teste
    parametros.add_argument("--teste", type=str, default="./SVM/Teste",
                            help="caminho de imagens para teste")

    # Armezena argumentos para aprendizado de máquina
    argumentos_SVM = vars(parametros.parse_args())

    # Retorna argumentos para aprendizado de máquina
    return argumentos_SVM


"""
    Descrição: Função para recorte da íris para extração de característica
    Parâmetro: Matriz da imagem da íris
    Retorno: Dois segmentos de íris
"""
def recorte_imagem(iris_imagem_array):
    # Converte matriz para imagem da íris
    iris_imagem = Image.fromarray(iris_imagem_array)

    # Armazena medidas de recorte do primeiro segmento de íris
    recorte_iris_fracao_parte1 = (0, 0, 70, 25)

    # Armazena medidas de recorte do segundo segmento de íris
    recorte_iris_fracao_parte2 = (25, 0, 50, 70)

    # Realiza recorte da imagem para formar o primeiro segmento de íris
    imagem_iris_recortada_parte1 = iris_imagem.crop(recorte_iris_fracao_parte1)

    # Converte imagem para matriz
    imagem_iris_recortada_parte1 = np.asarray(imagem_iris_recortada_parte1)

    # Realiza recorte da imagem para formar o primeiro segmento de íri
    imagem_iris_recortada_parte2 = iris_imagem.crop(recorte_iris_fracao_parte2)

    # Converte imagem para matriz
    imagem_iris_recortada_parte2 = np.asarray(imagem_iris_recortada_parte2)

    # Retorna segmento de íris
    return imagem_iris_recortada_parte1, imagem_iris_recortada_parte2

##-----------------------------------------------------------------------------
##  Execução - Função Principal
##-----------------------------------------------------------------------------
if __name__ == '__main__':
    # Psyco: Aceleramento de programas Python
    try:
        import psyco
        psyco.full()
    except ImportError:
        pass

    print("\n*********************************************************************")
    print("\nIniciando inscrição de banco de dados de íris...")
    print("\n*********************************************************************")
    inscricao_modelo_casia1()
```

```
print("\n**************************************************************************")
    print("\nIniciando aprendizado de máquina...")

    # Armazena tempo início de aprendizado de máquina
    inicio_tempo_aprendizado = time()

    # Argumentos gerais do sistema
    argumentos = argumentos_modelo()

    # Argumentos do aprendizado de máquina
    argumentos_SVM = argumentos_modelo_SVM()

    # Vetor para armazenamento de dados (vetores de características)
    dados = []

    # Vetor para armazenamento de classes
    classes = []

    print("\n**************************************************************************")
    print("\nEtapa de treinamento...")
    print("\n**************************************************************************")

    # Descrição: Laço de repetição que percorre diretório com imagens de treinamento da SVM
    for caminho_diretorio in paths.list_images(argumentos_SVM["treinamento"]):
        # Faz leitura de imagem no caminho especificado
        imagem = cv2.imread(caminho_diretorio)

        # Armazena segmentos de íris
        imagem1, imagem2 = recorte_imagem(imagem)

        # Armazena vetor de característica do primeiro segmento
        vetor_caracteristica1 = extracao_caracteristica(imagem1)

        # Armazena vetor de característica do segundo segmento
        vetor_caracteristica2 = extracao_caracteristica(imagem2)

        # Concatena os dois vetores de características
        vetor_caracteristica = vetor_caracteristica1 + vetor_caracteristica2

        # Construção de vetor com classe do atual segmento de íris concatenado
        classes.append(caminho_diretorio.split(os.path.sep)[-2])

        # Construção de vetor com características
        dados.append(vetor_caracteristica)

    # Parâmetros para SVM (Support Vector Machine)
    param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],
                  'gamma': [10000, 1000, 100, 1, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001],
                  'kernel': ['rbf']}

    # Função que realiza varredura de parâmetros para SVM (Support Vector Machine)
    grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=3, cv=5)

    # Função para treinamento do modelo com o grid de coeficientes
    grid.fit(dados, classes)

    # Vetor para armazenamento de dados de saída do teste
    saida_teste = []

    # Vetor para armazenamento de dados de predição do modelo
```

```
    saida_predicao = []

    print('\n**************************************************************************')
    print('\nEtapa de teste...')
    print('\n**************************************************************************')

    # Descrição: Laço de repetição que percorre diretório com imagens de teste da SVM
    for caminho_diretorio in paths.list_images(argumentos_SVM["teste"]):
        # Faz leitura de imagem no caminho especificado
        imagem = cv2.imread(caminho_diretorio)

        # Armazena segmentos de íris
        imagem1, imagem2 = recorte_imagem(imagem)

        # Armazena vetor de característica do primeiro segmento
        vetor_caracteristica1 = extracao_caracteristica(imagem1)

        # Armazena vetor de característica do segundo segmento
        vetor_caracteristica2 = extracao_caracteristica(imagem2)

        # Concatena os dois vetores de características
        vetor_caracteristica = vetor_caracteristica1 + vetor_caracteristica2

        # Transforma vetor em matriz
        vetor_caracteristica = np.asarray(vetor_caracteristica)

        # Realiza predição do modelo treinado
        predicao = grid.predict(vetor_caracteristica.reshape(1, -1))

        # Armazena resultados da predição
        saida_predicao.append(predicao)

        # Armazena classes a predição
        saida_teste.append(caminho_diretorio.split(os.path.sep)[-2])

        # Printa arquivo original ao lado do arquivo predito
        print('{}: {}'.format(caminho_diretorio, predicao))

    # Armazena tempo final de aprendizado de máquina
    fim_tempo_aprendizado = time()

    print('\nTempo de aprendizado de máquina: {} [s]\n'.format(fim_tempo_aprendizado -
inicio_tempo_aprendizado))

    scores = ['Precisao', 'Recall']

    # Laço de avaliação os resultados da SVM
    for score in scores:
        print("# Hyper-parameters for %s" % score)
        print()

        print("Melhor parâmetros encontrados no conjunto:")
        print()
        print(grid.best_params_)
        print()
        print("Pontuação do Grid no conjunto:")
        print()
        means = grid.cv_results_['mean_test_score']
        stds = grid.cv_results_['std_test_score']
        for mean, std, params in zip(means, stds, grid.cv_results_['params']):
```

```
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()

    print("Relatório detalhado de classificação:")
    print()
    print(classification_report(saida_teste, saida_predicao))
    print()

print("\n*********************************************************************")
print('Porcentagem de precision do conjunto')
precisao = accuracy_score(saida_teste, saida_predicao) * 100
print('{}%'.format(precisao))
print("\n*********************************************************************")
print('Porcentagem de recall do conjunto')
recall = recall_score(saida_teste, saida_predicao, average='weighted') * 100
print('{}%'.format(recall))
print("\n*********************************************************************")
print('Porcentagem F1 Score do conjunto')
f1_score = f1_score(saida_teste, saida_predicao, average='weighted') * 100
print('{}%'.format(f1_score))
print("\n*********************************************************************")

# Serializacao do resultado
with open('modelo_pickle', 'wb') as serial:
    pickle.dump(grid, serial)

# Desserializacao do resultado
with open('modelo_pickle', 'rb') as serial:
    modelo_pickle = pickle.load(serial)

# Armazena tempo de início de verificação da imagem de entrada
inicio_tempo_verificacao = time()

entrada = './Modelos/CASIA1/008_1_1.jpg.jpg'

# Leitura da imagem de entrada
imagem = cv2.imread(entrada)
print("\n*********************************************************************")
print('\nIMAGEM DE ENTRADA: {}'.format(entrada))
print("\n*********************************************************************")

# Armazena segmentos de íris
imagem1, imagem2 = recorte_imagem(imagem)

# Armazena vetor de característica do primeiro segmento
vetor_caracteristica1 = extracao_caracteristica(imagem1)

# Armazena vetor de característica do segundo segmento
vetor_caracteristica2 = extracao_caracteristica(imagem2)

# Concatena os dois vetores de características
vetor_caracteristica = vetor_caracteristica1 + vetor_caracteristica2

# Transforma vetor em matriz
vetor_caracteristica = np.asarray(vetor_caracteristica)

# Realiza predição do modelo treinado (arquivo desserializado)
resultado = modelo_pickle.predict(vetor_caracteristica.reshape(1, -1))
```

```
print("\n**************************************************************************")
print("\nRESULTADO:")
print("\n**************************************************************************")
# Mostra RESULTADO da verificação
print('USUÁRIO Nª {}'.format(resultado))

# Armazena tempo final de verificação da imagem de entrada
fim_tempo_verificacao = time()

print("\n**************************************************************************")
print('\nTempo de Verificação: {} [s]\n'.format(fim_tempo_verificacao - inicio_tempo_verificacao))
```

APÊNDICE H – IMPRESSÃO DE RESULTADOS PARA CLASSIFICADOR SVM COM 5 USUÁRIOS

****

Etapa de teste...

****
./SVM/Teste/3/003_1_2.jpg.jpg: ['3']
./SVM/Teste/3/003_2_2.jpg.jpg: ['3']
./SVM/Teste/5/005_1_2.jpg.jpg: ['5']
./SVM/Teste/5/005_2_2.jpg.jpg: ['5']
./SVM/Teste/2/002_1_2.jpg.jpg: ['2']
./SVM/Teste/2/002_2_2.jpg.jpg: ['2']
./SVM/Teste/1/001_1_2.jpg.jpg: ['1']
./SVM/Teste/1/001_2_2.jpg.jpg: ['1']
./SVM/Teste/4/004_2_2.jpg.jpg: ['4']
./SVM/Teste/4/004_1_2.jpg.jpg: ['4']

Tempo de aprendizado de máquina: 73.81361365318298 [s]

Hyper-parameters para Precision

Melhor parâmetros encontrados no conjunto:

{'C': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}

Pontuação do Grid no conjunto:

Relatório detalhado de classificação:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2
2	1.00	1.00	1.00	2
3	1.00	1.00	1.00	2
4	1.00	1.00	1.00	2
5	1.00	1.00	1.00	2
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10

Hyper-parameters para Recall

Melhor parâmetros encontrados no conjunto:

{'C': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}

Pontuação do Grid no conjunto:

Relatório detalhado de classificação:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2
2	1.00	1.00	1.00	2
3	1.00	1.00	1.00	2
4	1.00	1.00	1.00	2
5	1.00	1.00	1.00	2
accuracy			1.00	10
macro avg	1.00	1.00	1.00	10
weighted avg	1.00	1.00	1.00	10

****
Porcentagem de precision do conjunto
100.0%

****
Porcentagem de recall do conjunto
100.0%

****
Porcentagem F1 Score do conjunto
100.0%

****

Process finished with exit code 0

APÊNDICE I – IMPRESSÃO DE RESULTADOS PARA CLASSIFICADOR SVM COM 109 USUÁRIOS

```
# Hyper-parameters para precision
```

Melhor parâmetros encontrados no conjunto:

{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Pontuação do Grid no conjunto:

Relatório detalhado de classificação:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
10	0.00	0.00	0.00	2
100	0.50	1.00	0.67	2
101	1.00	0.50	0.67	2
102	0.33	0.50	0.40	2
103	0.00	0.00	0.00	2
104	0.33	0.50	0.40	2
105	1.00	0.50	0.67	2
106	1.00	0.50	0.67	2
107	1.00	0.50	0.67	2
108	1.00	1.00	1.00	2
11	0.50	0.50	0.50	2
12	1.00	0.50	0.67	2
13	0.00	0.00	0.00	2
14	0.50	1.00	0.67	2
15	1.00	1.00	1.00	2
16	0.50	1.00	0.67	2
17	0.20	0.50	0.29	2
18	0.00	0.00	0.00	2
19	1.00	0.50	0.67	2
2	0.50	0.50	0.50	2
20	0.50	0.50	0.50	2
21	1.00	1.00	1.00	2
22	1.00	0.50	0.67	2
23	0.00	0.00	0.00	2
24	1.00	0.50	0.67	2
25	0.00	0.00	0.00	2
26	0.29	1.00	0.44	2
27	1.00	1.00	1.00	2
28	1.00	1.00	1.00	2
29	0.67	1.00	0.80	2
3	0.67	1.00	0.80	2
30	0.50	1.00	0.67	2
31	0.50	1.00	0.67	2
32	0.00	0.00	0.00	2
33	1.00	0.50	0.67	2
34	1.00	1.00	1.00	2
35	0.50	0.50	0.50	2
36	1.00	1.00	1.00	2
37	0.67	1.00	0.80	2
38	0.50	0.50	0.50	2
39	0.00	0.00	0.00	2
4	1.00	1.00	1.00	2
40	0.33	0.50	0.40	2
41	1.00	1.00	1.00	2
42	1.00	0.50	0.67	2
43	0.00	0.00	0.00	2
44	1.00	0.50	0.67	2

45	0.50	0.50	0.50	2
46	0.50	1.00	0.67	2
47	1.00	1.00	1.00	2
48	1.00	1.00	1.00	2
49	1.00	1.00	1.00	2
5	0.67	1.00	0.80	2
50	0.00	0.00	0.00	2
51	0.25	0.50	0.33	2
52	1.00	0.50	0.67	2
53	1.00	0.50	0.67	2
54	1.00	0.50	0.67	2
55	1.00	0.50	0.67	2
56	0.50	0.50	0.50	2
57	0.33	0.50	0.40	2
58	1.00	1.00	1.00	2
59	0.00	0.00	0.00	2
6	1.00	0.50	0.67	2
60	0.50	1.00	0.67	2
61	1.00	0.50	0.67	2
62	0.50	0.50	0.50	2
63	0.20	0.50	0.29	2
64	1.00	0.50	0.67	2
65	1.00	1.00	1.00	2
66	0.50	1.00	0.67	2
67	0.67	1.00	0.80	2
68	1.00	1.00	1.00	2
69	0.00	0.00	0.00	2
7	0.50	0.50	0.50	2
70	0.00	0.00	0.00	2
71	1.00	1.00	1.00	2
72	0.50	0.50	0.50	2
73	0.33	0.50	0.40	2
74	0.50	0.50	0.50	2
75	1.00	1.00	1.00	2
76	0.00	0.00	0.00	2
77	1.00	1.00	1.00	2
78	0.00	0.00	0.00	2
79	0.50	0.50	0.50	2
8	0.67	1.00	0.80	2
80	0.50	1.00	0.67	2
81	0.00	0.00	0.00	2
82	0.67	1.00	0.80	2
83	0.29	1.00	0.44	2
84	1.00	0.50	0.67	2
85	1.00	0.50	0.67	2
86	1.00	1.00	1.00	2
87	0.50	1.00	0.67	2
88	0.50	0.50	0.50	2
89	1.00	1.00	1.00	2
9	1.00	1.00	1.00	2
90	0.00	0.00	0.00	2
91	0.50	0.50	0.50	2
92	1.00	0.50	0.67	2
93	0.67	1.00	0.80	2
94	1.00	0.50	0.67	2
95	0.67	1.00	0.80	2
96	0.33	0.50	0.40	2
97	0.00	0.00	0.00	2
98	0.00	0.00	0.00	2
99	1.00	1.00	1.00	2

accuracy			0.61	216
macro avg	0.60	0.61	0.57	216
weighted avg	0.60	0.61	0.57	216

Hyper-parameters para Recall

Melhor parâmetros encontrados no conjunto:

{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Relatório detalhado de classificação:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
10	0.00	0.00	0.00	2
100	0.50	1.00	0.67	2
101	1.00	0.50	0.67	2
102	0.33	0.50	0.40	2
103	0.00	0.00	0.00	2
104	0.33	0.50	0.40	2
105	1.00	0.50	0.67	2
106	1.00	0.50	0.67	2
107	1.00	0.50	0.67	2
108	1.00	1.00	1.00	2
11	0.50	0.50	0.50	2
12	1.00	0.50	0.67	2
13	0.00	0.00	0.00	2
14	0.50	1.00	0.67	2
15	1.00	1.00	1.00	2
16	0.50	1.00	0.67	2
17	0.20	0.50	0.29	2
18	0.00	0.00	0.00	2
19	1.00	0.50	0.67	2
2	0.50	0.50	0.50	2
20	0.50	0.50	0.50	2
21	1.00	1.00	1.00	2
22	1.00	0.50	0.67	2
23	0.00	0.00	0.00	2
24	1.00	0.50	0.67	2
25	0.00	0.00	0.00	2
26	0.29	1.00	0.44	2
27	1.00	1.00	1.00	2
28	1.00	1.00	1.00	2
29	0.67	1.00	0.80	2
3	0.67	1.00	0.80	2
30	0.50	1.00	0.67	2
31	0.50	1.00	0.67	2
32	0.00	0.00	0.00	2
33	1.00	0.50	0.67	2
34	1.00	1.00	1.00	2
35	0.50	0.50	0.50	2
36	1.00	1.00	1.00	2
37	0.67	1.00	0.80	2
38	0.50	0.50	0.50	2
39	0.00	0.00	0.00	2
4	1.00	1.00	1.00	2
40	0.33	0.50	0.40	2

41	1.00	1.00	1.00	2
42	1.00	0.50	0.67	2
43	0.00	0.00	0.00	2
44	1.00	0.50	0.67	2
45	0.50	0.50	0.50	2
46	0.50	1.00	0.67	2
47	1.00	1.00	1.00	2
48	1.00	1.00	1.00	2
49	1.00	1.00	1.00	2
5	0.67	1.00	0.80	2
50	0.00	0.00	0.00	2
51	0.25	0.50	0.33	2
52	1.00	0.50	0.67	2
53	1.00	0.50	0.67	2
54	1.00	0.50	0.67	2
55	1.00	0.50	0.67	2
56	0.50	0.50	0.50	2
57	0.33	0.50	0.40	2
58	1.00	1.00	1.00	2
59	0.00	0.00	0.00	2
6	1.00	0.50	0.67	2
60	0.50	1.00	0.67	2
61	1.00	0.50	0.67	2
62	0.50	0.50	0.50	2
63	0.20	0.50	0.29	2
64	1.00	0.50	0.67	2
65	1.00	1.00	1.00	2
66	0.50	1.00	0.67	2
67	0.67	1.00	0.80	2
68	1.00	1.00	1.00	2
69	0.00	0.00	0.00	2
7	0.50	0.50	0.50	2
70	0.00	0.00	0.00	2
71	1.00	1.00	1.00	2
72	0.50	0.50	0.50	2
73	0.33	0.50	0.40	2
74	0.50	0.50	0.50	2
75	1.00	1.00	1.00	2
76	0.00	0.00	0.00	2
77	1.00	1.00	1.00	2
78	0.00	0.00	0.00	2
79	0.50	0.50	0.50	2
8	0.67	1.00	0.80	2
80	0.50	1.00	0.67	2
81	0.00	0.00	0.00	2
82	0.67	1.00	0.80	2
83	0.29	1.00	0.44	2
84	1.00	0.50	0.67	2
85	1.00	0.50	0.67	2
86	1.00	1.00	1.00	2
87	0.50	1.00	0.67	2
88	0.50	0.50	0.50	2
89	1.00	1.00	1.00	2
9	1.00	1.00	1.00	2
90	0.00	0.00	0.00	2
91	0.50	0.50	0.50	2
92	1.00	0.50	0.67	2
93	0.67	1.00	0.80	2
94	1.00	0.50	0.67	2
95	0.67	1.00	0.80	2

96	0.33	0.50	0.40	2
97	0.00	0.00	0.00	2
98	0.00	0.00	0.00	2
99	1.00	1.00	1.00	2
accuracy			0.61	216
macro avg	0.60	0.61	0.57	216
weighted avg	0.60	0.61	0.57	216

****
Porcentagem de precision do conjunto
60.64814814814815%

****
Porcentagem de recall do conjunto
60.64814814814815%

****
Porcentagem F1 score do conjunto
57.15461493239271%

****

****

IMAGEM DE ENTRADA: ./Modelos/CASIA1/008_1_1.jpg.jpg

****

****

RESULTADO:

****
USUÁRIO Nª ['8']

****

Tempo de Verificação: 1.9561185836791992 [s]

Process finished with exit code 0