UNIVERSIDADE DE PASSO FUNDO

Vinícius Kerber Paranhos

SISTEMA DE DETECÇÃO DE SONOLÊNCIA

Passo Fundo

2019

Vinícius Kerber Paranhos

SISTEMA DE DETECÇÃO DE SONOLÊNCIA

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Me. Joan Michel Levandoski.

Passo Fundo

2019

Vinícius Kerber Paranhos

Sistema de Detecção de Sonolência

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Ms Joan Michel Levandoski.

Aprovado em ____ de _____ de _____.

BANCA EXAMINADORA

_____

Prof. Me. Joan Michel Levandoski - UPF

_____

Prof. Dr. Blanca Rosa Maquera Sosa - UPF

_____

Prof. Dr. Carlos Alberto Ramirez Behaine - UPF

Este trabalho é dedicado aos meus pais Edmar e Sandra por todo o esforço e dedicação.

“Um homem é bem-sucedido se acorda de manhã, vai para a cama de noite e persegue seus sonhos enquanto isso.”

Bob Dylan

RESUMO

O avanço tecnológico na indústria automotiva fez com que os sistemas de auxílio ao condutor estejam cada vez mais presentes nos veículos atuais. Estes tipos de dispositivo visam garantir a segurança do usuário, buscando uma melhor sinergia entre homem e máquina em tempos onde as estradas são o maior causador de mortes no país. Pensando nisso, originou-se a ideia de desenvolver um dispositivo de baixo custo que detecta sinais de sonolência por meio de técnicas de processamento digital de imagens, essas imagens são capturadas no interior do veículo e são processadas para a obtenção dos parâmetros EAR e MAR que analisam condições fisiológicas do motorista, e caso sejam identificados sinais de sonolência o sistema gera alertas sonoros que avisam o condutor sobre os possíveis riscos de dirigir sonolento. Para realizar tal desafio, foi utilizado um microcomputador Raspberry Pi em conjunto com o módulo de câmera Raspberry Pi câmera V2 noir, também foi projeto um sistema que contempla alertas sonoros e visuais. A detecção de sonolência acontece por meio da aplicação dos algoritmos de Viola Jones e dos *Facial Landmarks,* que compõem o software embarcado no microcomputador.

Palavras-Chave: Raspberry Pi, Detector de Sonolência, Viola Jones, *Facial Landmarks*.

ABSTRACT

Technological advances in the automotive industry have meant that driver assistance systems are increasingly present in today's vehicles. These types of devices aim to ensure the safety of the user, seeking a better synergy between man and machine in times where roads are the biggest cause of death in the country. With this in mind, the idea of developing a low-cost device that detects signs of drowsiness by means of digital image processing techniques originated. These images are captured inside the vehicle and are processed to obtain EAR and MAR parameters that analyze the physiological conditions of the driver, and if sleepiness signals are identified, the system generates sound alerts that warn the driver about the possible risks of sleepy driving. To meet this challenge, a Raspberry Pi microcomputer was used in conjunction with the Raspberry Pi camera module V2 noir camera, a system was also designed that includes sound and visual alerts. The detection of drowsiness happens through the application of the algorithms of Viola Jones and Facial Landmarks, which make up the software embedded in the microcomputer.

Keywords: Raspberry Pi, Driver Drowsiness, Viola Jones, Facial Landmarks.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fases do Sono 16
Figura 2 – Sistema utilizando sinais de ECG 18
Figura 3 – Detector de Sonolência da Bosch 19
Figura 4 – Sistema de detecção de sonolência por imagens da Harman 20
Figura 5 – Passos fundamentais em processamento digital de imagens 21
Figura 6 – Processo de aquisição de uma imagem digital 23
Figura 7 – Comparativo das diferenças entre CCD e CMOS 24
Figura 8 – Funcionamento de um filtro espacial linear 26
Figura 9 – Analogia sistema de visão humana e computacional 30
Figura 10 – Diagrama de funcionamento do projeto 31
Figura 11 – Características Haar 32
Figura 12 – Extração dos recursos de uma imagem 33
Figura 13 – Imagem integral 34
Figura 14 – Classificador em cascata 35
Figura 15 – Facial Landmark 37
Figura 16 – Pontos de referência olho aberto 37
Figura 17 – Gráfico EAR olho fechado 38
Figura 18 – Ponto de referência na região da boca 39
Figura 19 – Gráfico da relação de aspecto da boca (MAR) 40
Figura 20 – Raspberry Pi 3 model B+ 41
Figura 21 – Raspberry Pi 3 model B+ GPIO 41
Figura 22 – Módulo de Câmera V2 8mp 42
Figura 23 – Fluxograma do software 47
Figura 24 – Protótipo do Hardware 48
Figura 25 – Processo de conversão 49
Figura 26 – Filtro de mediana na imagem L.A.B 50
Figura 27 – Iluminação invertida 50
Figura 28 – Nova imagem gerada 51
Figura 29 – Módulo de Câmera V2 8mp 52
Figura 30 – Face com facial Landmarks 52
Figura 31 – Alertas 54

Figura 32 – Configurações dos testes 55
Figura 33 – Gráfico do EAR obtido 56
Figura 34 – Gráfico do MAR obtido 57
Figura 35 – Resultado do teste com distância 50 cm 58
Figura 36 – Resultado do teste realizado com veículo em movimento 59
Figura 37 – Detecção usando óculos 60

LISTA DE QUADROS

Quadro 1 - Especificações técnicas do Raspberry Pi 3 model B+ 40
Quadro 2 - Características Técnicas do Módulo V2 43
Quadro 3 - Teste de validação com distância 60 cm 57
Quadro 4 - Teste de validação com distância 50 cm 58
Quadro 5 – Teste com veículo em movimento 59

LISTA DE ABREVIATURAS

ABRAMET – Associação Brasileira de Medicina de Tráfego

ECG – Eletrocardiograma

EEG – Eletroencefalograma

CCD – Charge Couple Device – Dispositivo de carga acoplada

CES – Consumer Eletronic Show

CMOS – Complementary Metal-Oxide Semiconductor – Metal-óxido semicondutor complementar

DDD – Driver Drowsiness Detection – Detecção de Sonolência do Condutor

DETRAN – Departamento Estadual de Trânsito

FPS – Frames per second – Quadros por segundo

HF – High Frequencies – Frequencias Altas

HRV – Heart Rate Variability – Variabilidade da Frequência Cardíaca

IA – Artificial Intelligence – Inteligencia Artificial

LED – Light Emitting Diode – Diodo Emissor de Luz

LF – Low Frequencies – Baixas frequencias

NREM – Non Rapid Eye Moviment – Movimento Não Rápido dos Olhos

REM – Rapid Eye Moviment – Movimento Rápido dos Olhos

EAR – Eye Aspect Ratio – Relação de aspecto do olho

MAR – Mounth Aspect Ratio – Relação de aspecto da boca

SUMÁRIO

1 INTRODUÇÃO 13
1.1 CONTEXTO 13
1.2 OBJETIVO GERAL 13
1.3 OBJETIVOS ESPECÍFICOS 14
1.4 JUSTIFICATIVA 14
2 REVISÃO DA LITERATURA 15
2.1 SONO 15
2.1.1 Acidentes causados por sonolência 16
2.2 SISTEMAS DE DETECÇÃO DE SONOLÊNCIA 17
2.2.1 Método baseado na análise Fisiológica 17
2.2.2 Método baseado na análise do Veículo 18
2.2.3 Método baseado na análise da Face 19
2.3 PROCESSAMENTO DIGITAL DE IMAGENS 21
2.3.1 Introdução 21
2.3.2 Aquisição de imagens 22
2.3.3 Histograma 24
2.3.4 Segmentação 25
2.3.5 Filtros 26
2.4 COMPUTAÇÃO VISUAL 29
3 MATERIAIS E MÉTODOS 31
3.1 ALGORITMO DE VIOLA JONES 32
3.1.1 Características Haar 32
3.1.2 Imagem Integral 33
3.1.3 CLASSIFICADORES E CASCATAS HAAR 35

3.2 DETECÇÃO DE PONTOS DE REFERÊNCIA FACIAIS 36
3.3 EYE ASPECT RATIO 37
3.4 MOUTH ASPECT RATIO 39
3.5 RASPBERRY PI 40
3.6 RASPBERRY PI CÂMERA 42
3.7 ALERTAS 43
3.8 FONTE DE ALIMENTAÇÃO 43
3.9 PYTHON 43
3.10 BIBLIOTECAS DE PYTHON 44
3.10.1 SciPy 44
3.10.2 NumPy 44
3.10.3 Imutils 44
3.10.4 GPIO 45
3.10.5 OpenCV 45
3.10.6 Dlib 46
4 DESENVOLVIMENTO DO PROJETO 47
5 RESULTADOS E DISCUSSÕES 55
5.1 CONFIGURAÇÕES E LOCAIS UTILIZADOS 55
5.2 OS TESTES 56
5.3 AS ADVERSIDADES 59
6 CONCLUSÕES FINAIS 61
REFERÊNCIAS 62
APÊNDICE A – SCRIPT DA BIBLIOTECA IMUTILS ALTERADA 65

1 INTRODUÇÃO

Neste capítulo serão abordados os objetivos, a justificativa e o contexto do projeto. Essas são informações crucias para o desenvolvimento deste trabalho

1.1 CONTEXTO

Uma das maiores causas de acidentes de trânsito no Brasil é a sonolência, segundo o Departamento Estadual de Trânsito de São Paulo (Detran.SP) cerca de 20 % dos acidentes que ocorrem em estradas brasileiras são causados por motoristas que dormiram ao volante. Situações deste tipo são cada vez mais frequentes, segundo dados da Associação Brasileira de Medicina de tráfego (ABRAMET), em pesquisa realizada com 495 pessoas no de 2017, identificou que cerca 40% dos entrevistados já vivenciou situações de perigo por dirigir com sono.

A evolução tecnológica vivenciada nos últimos anos possibilitou o surgimento de métodos sofisticados que são capazes de detectar sinais de sonolência. Um exemplo é sistema desenvolvido pela Bosch que analisa por meio de sensores o comportamento do veículo durante uma viagem e dependendo dos dados obtidos pode identificar que o motorista está com sinais de fadiga.

No Brasil esse sistema foi implantando 2012 pela Volkswagen, como o nome de detector de fadiga. Ele monitora o ângulo de esterçamento do volante, a pressão aplicada nos pedais e as acelerações laterais e longitudinal do veículo. Ao longo percurso o sistema compara os dados atuais com os de início de viagem, e sugere em caso de detecção de fadiga que o condutor realize uma parada.

1.2 OBJETIVO GERAL

O projeto visa desenvolver um sistema veicular embarcado de baixo custo, que seja capaz de detectar sinais de sonolência através de imagens capturadas por uma câmera instalada no interior do veículo.

1.3 OBJETIVOS ESPECÍFICOS

a) Capturar imagens do condutor através de câmera instalada no interior do veículo;
b) Detectar face e o nível de abertura dos olhos e da boca;
c) Desenvolver um software que detecte sonolência pelos dados coletados;
d) Criar um protótipo de baixo custo para implementar o software;
e) Realizar testes para validar o funcionamento do protótipo desenvolvido;

1.4 JUSTIFICATIVA

De acordo com o que foi mencionado na introdução, dormir ao volante é um dos maiores causadores de acidentes automotivos. Desta forma ficou evidente a necessidade de desenvolver um dispositivo que seja capaz de detectar sinais de sonolência por meio de imagens, e que tenha como diferenciais um tempo de resposta menor, se comparado com os modelos comerciais, além de possuir a capacidade de detectar micros sonos. Evitando um risco desnecessário para o condutor.

2 REVISÃO DA LITERATURA

Neste capítulo serão apresentados os conceitos que serão uteis para elaboração do detector de fadiga. Buscando informações de como a sonolência pode afetar os condutores, além de descrever a diferentes técnicas que podem ser empregadas a fim de detectar sinais de fadiga.

2.1 SONO

O sono pode ser categorizado em duas fases: Movimento não rápido dos olhos (NREM – *Non Rapid Eye Moviment*) e Movimento rápido dos olhos (REM - *Rapid Eye Moviment*). O estado NREM, que equivale a cerca de 75% do período de sono, pode ser dividido em 4 estágios (DANTAS, 2018):

- Estágio 1: É a etapa da sonolência, onde começam os primeiros sinais de fadiga. Neste estágio o indivíduo pode ser acordado facilmente.
- Estágio 2: Com duração de 5 a 25 minutos. A atividade cardíaca diminui, ocorre um relaxamento dos músculos e a temperatura corporal cai. Poder ser denominado de sono leve, pois, já não é possível despertar facilmente.
- Estágio 3: Pode ser caracterizado como um período transitório entre o sono leve e o sono profundo. O ritmo cardíaco diminui e a respiração fica mais lenta.
- Estágio 4: Denominado de sono profundo. Tem duração de cerca de 40 minutos, sendo muito semelhante ao estágio 3, porém, o nível de sonolência está em um nível ainda mais profundo.

O sono REM dura 25% do tempo de repouso, é nesta fase onde os sonhos ocorrem. Pode ser especificado pela ocorrência de movimentos oculares rápidos, contração muscular e a respiração se torna rápida e irregular. Essas características são resultado de uma intensa atividade cerebral (DANTAS, 2018).

Os sonos NREM e REM costumam se alternar ciclicamente durante o período de descanso. Juntos eles formam um ciclo completo, que dura em média 90 minutos e se repete entre 4 e 6 vezes por noite. Esses ciclos podem sofrer alteração com relação a sua duração ou mesmo composição por fatores como a idade ou mesmo doenças. A Figura 1 ilustra um ciclo completo de sono (TERRA, 2016).

Figura 1 – Fases do sono

Fonte: https://www.bedtime.com.br/blog/curiosidade-do-sono/quais-sao-as-fases-do-sono/

2.1.1 Acidentes causados por sonolência

Para analisar os efeitos da sonolência nos condutores foram realizados estudos, que identificaram que as colisões que ocorrem devido aos efeitos da sonolência têm várias características (SAHAYADHAS, 2012):

- Ocorre tarde da noite (das 0h às 07h) ou no meio da tarde (das 14h às 16h);
- Envolver um único veículo que sai da pista;
- Ocorre em rodovias;
- O motorista está frequentemente sozinho;
- O motorista é geralmente um jovem de 16 a 25 anos;
- Sem marcas de derrapagem ou indicação de freada brusca;

Em relação a essas características, os bancos de dados da *Southwest England* e da *Midlands Police* usam os seguintes critérios para identificar acidentes causados por sonolência (SAHAYADHAS, 2012):

- Nível de álcool no sangue abaixo do limite legal de direção;
- Veículo não tem defeito mecânico;
- Boas condições meteorológicas e visibilidade clara;
- Eliminação do “excesso de velocidade” ou “dirigir muito perto do veículo na frente” como causas potenciais;

A partir do que foi coletado nós estudos, foi possível observar que os acidentes causando por fadiga geralmente acontecem em um estágio inicial do sono, um período de transição do acordado para o sono leve. Outro ponto está relacionado ao fato de que este tipo de acidente geralmente acontece como condutores jovem, o que se presume uma certa falta de experiência.

2.2 SISTEMAS DE DETECÇÃO DE SONOLÊNCIA

O termo sistema de Detecção de Sonolência do Condutor (DDD – *Driver Drowsiness Detection*) refere-se aos sistemas embarcados no veículo que monitoram o comportamento do motorista ou mesmo do veículo. Esses dispositivos monitoram o desempenho do condutor e fornecem alertas ou estímulos se ele parecer estar com problemas. Esse sistema emite alertas quando identifica sinais de fadiga no motorista.

2.2.1 Método baseado na análise Fisiológica

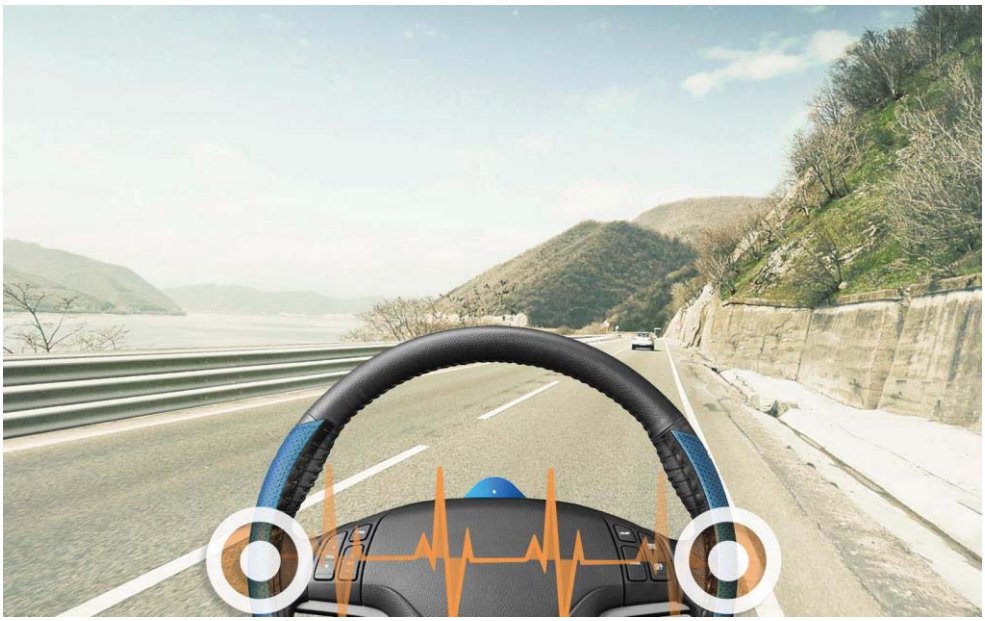
Sinais fisiológicos começam a mudar em estágios iniciais de sonolência. Sendo assim, os mais adequados para detectar a sonolência com poucos falsos positivos tornando possível alertar um motorista sonolento em tempo hábil e, assim, evitar muitos acidentes rodoviários. Muitos pesquisadores consideram que os sinais biomédicos ideias para realizar a detecção de sonolência são eletrocardiograma (ECG), eletroencefalograma (EEG) (SAHAYADHAS, 2012).

A frequência cardíaca (*HR – Heart rate*) varia significativamente entre os diferentes estágios de sonolência, como alerta e fadiga. Portanto, frequência cardíaca, que pode ser facilmente determinada pelo Sinal de ECG, também pode ser usada para detectar a sonolência. Outras formas de mensurar o estágio de sono é utilizando Variabilidade da Frequência Cardíaca (HRV – *Heart Rate Variability*), em que as frequências baixas (LF – *Low Frequencies*) e altas (HF – *High Frequencies*) caem no intervalo de 0.04–0.15 Hz e 0.14–0.4Hz (SAHAYADHAS, 2012).

O sinal EEG possui várias bandas de frequência, incluindo a banda delta (0,5 a 4 Hz), que corresponde à atividade do sono, a banda teta (4 a 8 Hz), que está relacionada à sonolência, a banda alfa (8 a 13 Hz) , que representa relaxamento e criatividade, e a banda beta (13–25 Hz), que corresponde ao estado de alerta. Uma diminuição nas mudanças de poder na banda de frequência alfa e um aumento na banda de frequência teta indica sonolência (SAHAYADHAS, 2012).

O grande problema destes métodos é a natureza intrusiva, já que existe a necessidade de conectar elétrodos ao corpo do indivíduo. Para solucionar este tipo problema já foram elaborados sistemas onde os elétrodos foram colocados no volante ou mesmo no banco do motorista, a Figura 2 ilustra um modelo desenvolvido pela CardioWhel que utiliza sensores no volante para detectar sinais de ECG. (SAHAYADHAS, 2012).

Figura 2 – Sistema utilizando sinais de ECG



Fonte: https://www.cardio-id.com/cardiowheel

2.2.2 Método baseado na análise do Veículo

Esse método é baseado em um algoritmo, que começa a registrar comportamento de direção do motorista quando a viagem começa, a partir de sensores instalados no veículo. Em seguida, reconhece as mudanças durante longas viagens e, portanto, também o nível de fadiga do motorista. Sinais típicos de concentração decrescente são fases durante as quais o motorista mal dirige, combinado com movimentos de direção leves (entre 0.5º e 5º), mas rápidos e abruptos, para manter o carro na pista (SAHAYADHAS, 2012).

Com base na frequência desses movimentos e outros parâmetros, entre eles a duração de uma viagem, o uso de sinais de volta e a hora do dia, a função calcula o nível de fadiga do motorista. Se esse nível exceder um determinado valor, um ícone, como uma xícara de café, piscará no painel de instrumentos para avisar os motoristas que precisam de um descanso (BOSCH, 2018).

Este tipo de sistema é o mais empregado comercialmente. Sendo utilizado por grandes empresas como por exemplo a Bosch, Volkswagen e a Mercedez Benz. Porém este método está sujeito a várias limitações como o tipo de veículo, experiência do motorista, características geométricas, condição da estrada (SAHAYADHAS, 2012).

Além de que, esses procedimentos exigem um tempo considerável para analisar o comportamento do usuário e, portanto, não funcionam com os chamados micro sonos, que é quando um motorista sonolento adormece por alguns segundos em uma seção de estrada muito reta sem alterar os sinais do veículo (BOSCH, 2018).

Figura 3 – Detector de Sonolencia da Bosch

Fonte: https://plus.google.com/photos/photo/118368718422318898642/6253037441454667970

2.2.3 Método baseado na análise da Face

Uma pessoa sonolenta exibe vários movimentos faciais característicos, incluindo piscar rápido e constante, balançar a cabeça ou bocejar frequentemente. Abordagens comportamentais computadorizadas, não intrusivas, são amplamente utilizadas para determinar o nível de sonolência dos motoristas medindo seus comportamentos anormais. A maioria dos estudos publicados sobre o uso de abordagens comportamentais para determinar a sonolência, se concentra no método que analise o Fechamento das Pálpebras (PERCLOS – *Percentage of Eye*

Closure). Essa medida foi considerada confiável para prever a sonolência e tem sido usada em produtos comerciais produzidos Harman, Seeing Machines e a Lexus (SAHAYADHAS, 2012).

A principal limitação do uso de uma abordagem baseada na visão é a iluminação. Câmeras normais não funcionam bem à noite. A fim de superar essa limitação, alguns pesquisadores usaram a iluminação ativa utilizando um diodo emissor de luz (LED) infravermelho. No entanto, embora funcionem razoavelmente bem à noite, os LEDs são considerados menos robustos durante o dia (SAHAYADHAS, 2012).

Depois de capturar o vídeo, algumas técnicas, incluindo Análise de Componentes Conectados, Cascata de Classificadores ou Transformação Hough, Filtro Gabor, Algoritmo Haar são aplicadas para detectar a face, olhos ou boca. Depois de localizar a região de interesse específica na imagem, recursos como PERCLOS, frequência de bocejo e ângulo de cabeça são extraídos (SAHAYADHAS, 2012).

O comportamento é então analisado e classificado como normal, levemente sonolento, altamente sonolento. No entanto, verificou-se que a taxa de detecção do recurso correto, ou a porcentagem de sucesso entre um número de tentativas de detecção, varia dependendo da aplicação e do número de classes. A determinação da sonolência usando o método PERCLOS tem uma taxa de sucesso próxima a 98%. Contudo, deve notar-se que a elevada taxa de detecção ocorre em situações ideais onde, por exemplo, os indivíduos não usam óculos (SAHAYADHAS, 2012).

A Figura 4 ilustra um sistema de detecção de sonolência por meio do fechamento das pálpebras desenvolvido pela Harman, e demostrado no ano de 2016 na Consumer Eletronic Show (CES).

Figura 4 – Sistema de detecção de sonolência por imagens da Harman

Fonte: https://www.gizmodo.com.au/2016/01/the-first-pupil-based-monitoring-system-can-tell-when-a-driver-is-too-distracted/

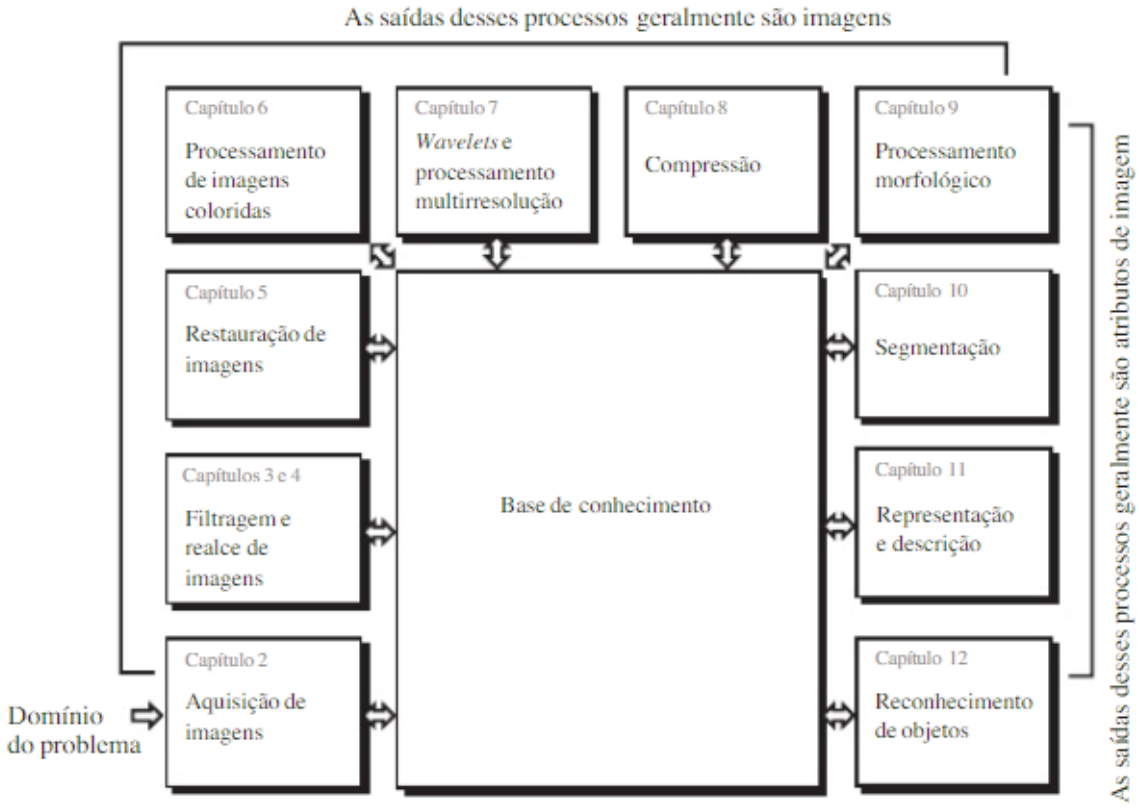
2.3 PROCESSAMENTO DIGITAL DE IMAGENS

2.3.1 Introdução

A definição de imagem pode ser obtida por uma função bidimensional $f\ (x, y)$, onde x e y são coordenadas espaciais no plano, e a amplitude de f em qualquer par de coordenadas (x, y) é nomeada de intensidade ou nível de cinza da imagem nesse ponto. Denomina-se de imagem digital quantidades finitas e discretas de x, y e f. Com o auxilio de um computador digital pode-se processar imagens digitais que são formadas por um número finito de elementos cada um com uma localização e com um valor específico, essas medidas podem ser nomeadas como elementos pictóricos, elementos de imagem, pels ou *pixels*, onde *pixel* é a palavra mais utilizada para definir uma imagem digital, e a esse campo recebe o nome de processamento digital de imagens (GONZALEZ e WOODS, 2010).

Alguns dos passos principais do processamento digital de imagens estão ilustrados na Figura 5.

Figura 5 – Passos fundamentais em processamento digital de imagens



Fonte: Gonzales e Woods (2010, p. 16)

2.3.2 Aquisição de imagens

2.3.2.1 Aquisição de imagens utilizando sensores matriciais

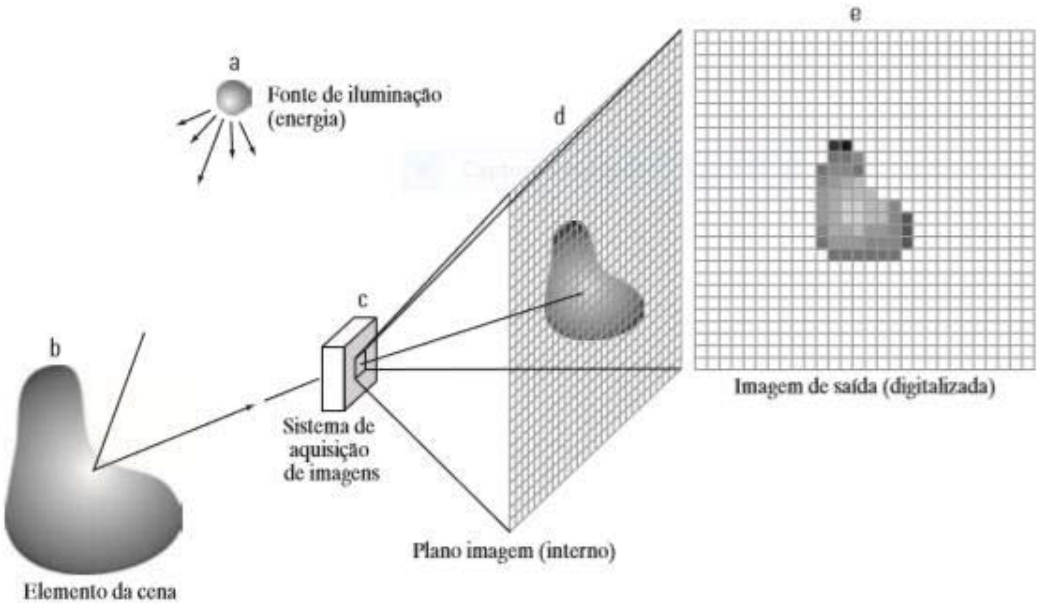
Existem inúmeros dispositivos sensores eletromagnéticos e alguns ultrassônicos que são utilizados de forma matricial. Essa organização também é encontrada nas câmeras digitais. A resposta de cada sensor é proporcional à integral da energia luminosa projetada sobre sua superfície. (GONZALEZ; WOODS, 2010)

O esquema de utilização dos sensores matriciais é indicado na Figura 6 (c): a energia de uma fonte de iluminação é refletida de um elemento da cena. A primeira atribuição realizada pelo sistema de aquisição de imagens da figura é coletar a energia de entrada e projetá-la num plano. Considerando a fonte de iluminação, a entrada frontal do sistema de aquisição de imagens é constituída por uma lente ótica que projeta a cena vista sobre o plano focal da lente, como é indicado na Figura 6 (d). O conjunto de sensores, que corresponde com o plano focal, produz saídas integrais da luz recebida em cada sensor. Enquanto isso, circuitos digitais e analógicos realizam uma varredura nas saídas de cada sensor e as convertem em um sinal analógico, que em sequência é digitalizado por outro componente do sistema de aquisição de imagem. (GONZALEZ; WOODS, 2010)

O conceito fundamental de amostragem e de quantização é representado conforme a Figura 6. No item ‘a’, é representada uma imagem continua f que se deseja converter em formato digital. A mesma pode ser continua tanto nas coordenadas (x,y) quanto em sua amplitude. Para realizar a conversão de uma imagem para o formato digital deve-se fazer a digitalização dos valores das coordenadas da função (amostragem) em ambas às coordenadas e na amplitude (quantização). (GONZALEZ; WOODS, 2010)

A primeira função realizada pelo sistema, como destacado na Figura 6 (c), é coletar a entrada de energia e focalizá-la em um plano de imagem. Se a iluminação é leve, a frente fim do sistema de imagem é uma lente óptica que projeta a cena vista no plano focal da lente, como mostra a Figura 6 (d). A matriz de sensores, que é coincidente com o plano focal, produz saídas proporcionais à integral da luz recebida em cada sensor. Circuitos digitais e analógicos varrem essas saídas e fazem a conversão para um sinal analógico, que é, finalmente, digitalizado por outra seção do sistema de imagem. A saída é uma imagem digital, como mostrado esquematicamente na Figura 6 (e). (GONZALEZ; WOODS, 2010)

Figura 6 - Processo de aquisição de uma imagem digital: (a) Fonte de energia (“iluminação”). (b) Elemento de uma cena. (c) Sistema de aquisição de imagem. (d) Projeção da cena no plano imagem. (e) Imagem digitalizada.



Fonte: Gonzales e Woods (2010, p. 32)

2.3.2.2 Diferenças entre dispositivos CCD e CMOS

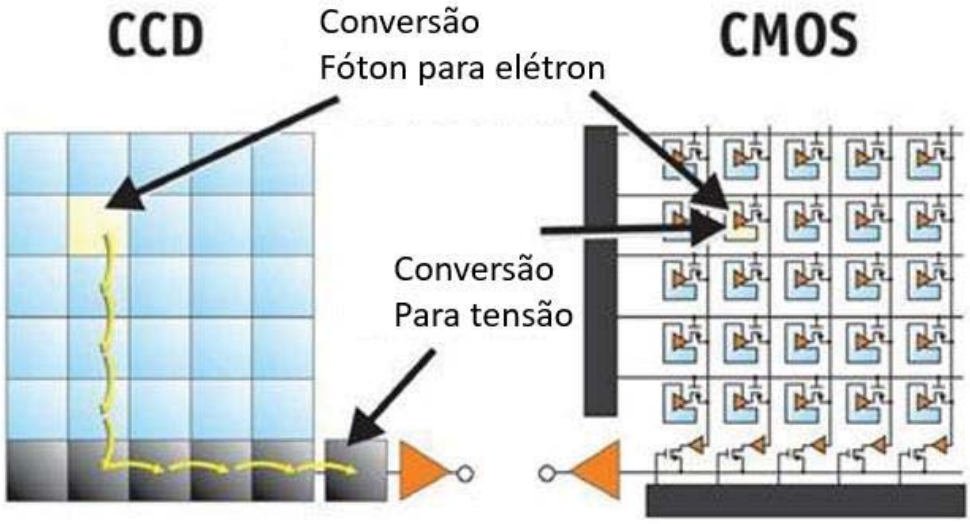
Existem duas principais tecnologias que integram os dispositivos que capturam imagens, CMOS e CCD. A câmera *Charge Couple Device* (CCD), que consiste em uma matriz de células semicondutoras fotossensíveis, que tem a função de capacitores, armazenando a carga elétrica proporcional a energia elétrica incidente (AZEVEDO, 2008).

A câmera CCD monocromática possui um conjunto de lentes que focalizam a imagem na área fotossensível do sensor. Quando a captura é de imagens coloridas se faz necessário a utilização de prismas e filtros que farão a decomposição da imagem colorida em suas componentes *Red*, *Green* e *Blue* (RGB). Para cada um dos seus componentes RGB se faz necessário um sensor CCD independente (AZEVEDO, 2008).

Os sensores *Complementary Metal-Oxide Semicondu*ctor (CMOS) contêm fileiras de fotodiodos, que convertem a luz (fótons) em carga elétrica (elétrons). O sensor faz uma varredura, lendo cada fileira de fotodiodos uma a uma, e envia os dados para um processador, que monta a imagem completa. Para capturar as cores, cada pixel é coberto por um filtro RGB. Eles estão organizados no que se chama “matriz de Bayer”: para cada par de pixels vermelho e azul, há dois pixels verdes (VENTURA,2018).

Pode-se visualizar a diferença no processo de ambas as tecnologias na Figura 7.

Figura 7 – Comparativo das diferenças entre CCD e CMOS



Fonte: Adaptado de http://meroli.web.cern.ch/lecture_cmos_vs_ccd_pixel_sensor.html

2.3.3 Histograma

Um histograma de uma imagem é um gráfico de frequência relativa de ocorrência de cada um dos valores de pixel permitidos na imagem na função dos próprios valores. Se normalizarmos tal gráfico de frequência, de modo que a soma total de todas as entradas de frequência na faixa permitida seja unitária, podemos tratar o histograma da imagem como uma função densidade de probabilidade discreta, que define a probabilidade de ocorrência de um determinado valor de pixel na imagem. A inspeção visual de um histograma de imagem pode revelar o contraste básico presente na imagem, assim como qualquer diferença potencial na distribuição de cores de componentes da cena na frente e no fundo da imagem. (SOLOMON; BRECKON, 2013)

No caso de uma simples imagem em escala de cinza o histograma pode ser construído com a contagem do número de vezes que cada valor da escala de cinza (0-255) ocorre na imagem. A cada vez que um determinado valor é encontrado, o correspondente intervalo de valores no histograma sofre um incremento. (SOLOMON; BRECKON, 2013)

2.3.4 Segmentação

A segmentação subdivide uma imagem em regiões ou objetos que a compõem. O nível de detalhe em que a subdivisão é realizada depende do problema a ser resolvido. Ou seja, a segmentação deve parar quando os objetos ou as regiões de interesse de uma aplicação forem detectados. Por exemplo, na inspeção automatizada de componentes eletrônicos, o interesse está em analisar as imagens dos produtos com o intuito de determinar a presença ou ausência de anomalias específicas, como a falta de componentes ou circuitos de conexão interrompidos.

A segmentação de imagens não triviais é uma das tarefas mais difíceis no processamento de imagens. A precisão da segmentação determina o sucesso ou o fracasso final dos procedimentos de análise computadorizada. Por essa razão, deve-se tomar muito cuidado para aumentar a probabilidade de se obter uma segmentação precisa. (GONZALEZ; WOODS, 2010)

2.3.2.1 Segmentação por Limiarização

A Limiarização consiste em uma técnica de separação de regiões interesse de uma área presente no fundo da imagem. Existem inúmeros métodos de empregar a limirização, um dos métodos mais simples de aplicar essa técnica é por meio da bipartição do histograma. A extração de objetos do fundo de uma imagem funciona selecionando um limiar T, que separa em modos. Então, qualquer ponto (x, y) na imagem em que f (x, y) > T é chamado de ponto do objeto; caso contrário, o ponto é chamado ponto de fundo. Em outras palavras, a imagem segmentada, g (x, y), é dada pela seguinte equação: (GONZALEZ; WOODS, 2010)

$$g(x,y) = \begin{cases} 1, & se\ f(x,y) > T \\ 0, & se\ f(x,y) \leq T \end{cases} \quad (1)$$

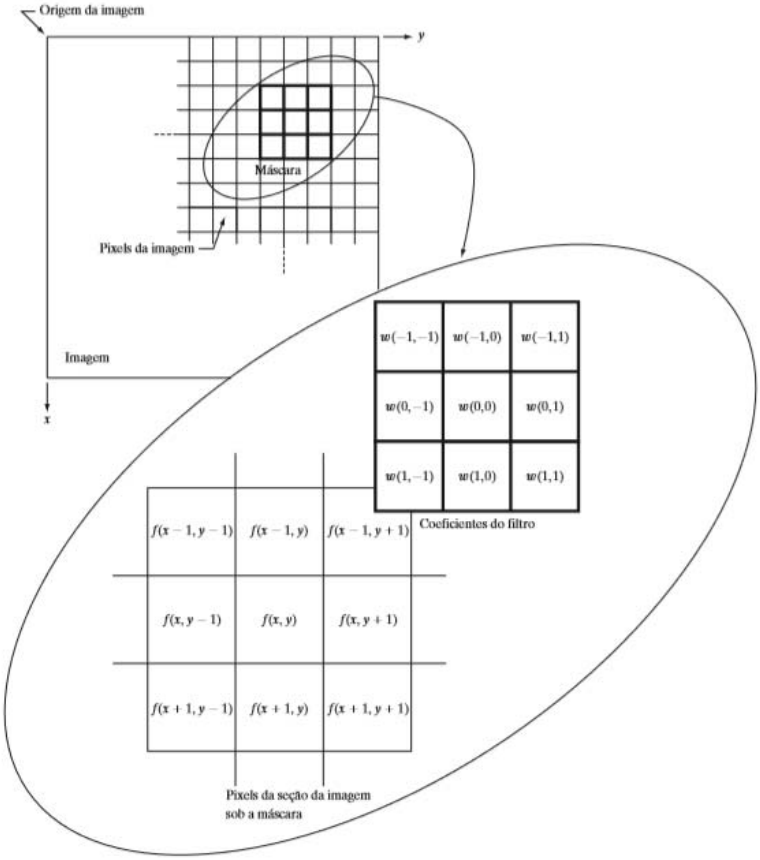
Quando o limiar (T) é uma constante aplicável a uma imagem inteira, esse processo é conhecido como limiarização global. Quando o valor do limiar muda ao longo da imagem ele é denominado como limiarização variável. (GONZALES; WOODS, 2010)

2.3.5 Filtros

A filtragem espacial é uma das principais ferramentas utilizadas na área para uma ampla gama de aplicações, de forma que recomendamos fortemente que você desenvolva uma sólida compreensão desses conceitos.

Um filtro espacial consiste em uma vizinhança (normalmente um pequeno retângulo), uma operação predefinida realizada sobre os pixels da imagem incluídos na vizinhança. A filtragem cria um pixel com coordenadas iguais às coordenadas do centro da vizinhança, e cujo valor é o resultado da operação de filtragem. Uma imagem processada (filtrada) é gerada à medida que o centro do filtro percorre cada pixel na imagem de entrada. Se a operação realizada sobre os pixels da imagem for linear, o filtro é chamado de filtro espacial linear. Caso contrário, o filtro é não linear. A Figura 8 ilustra o funcionamento da filtragem espacial linear utilizando uma vizinhança 3 × 3. (GONZALES; WOODS, 2010)

Figura 8 – Funcionamento de um filtro espacial linear



Fonte: Gonzales e Woods (2010, p. 32)

Para o tamanho de máscara *m x n* consideramos que m= 2a +1 e n= 2b+1, onde a e b são valores inteiros positivos Em geral, a filtragem espacial linear de uma imagem de dimensões M × N com um filtro de dimensões m×n é dada pela expressão, x e y variam de forma que cada pixel em w percorre todos os pixels em f: (GONZALES; WOODS, 2010)

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t) \tag{2}$$

2.3.5.1 Filtros de Média Aritmética

Este é o mais simples dos filtros de média. Seja S_{xy}o conjunto de coordenadas em uma janela de subimagem retangular (vizinhança) de tamanho m × n, centrada no ponto (x, y). O filtro de média aritmética calcula o valor médio da imagem corrompida *g (s, t)* na área definida por S_{xy}. O valor da imagem restaurada $\hat{f}$ no ponto (x, y) é simplesmente a média aritmética calculada utilizando os pixels da região definida por S_{xy}. (GONZALES; WOODS, 2010)

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}}g(s,t) \tag{3}$$

Essa operação pode ser implementada utilizando um filtro espacial de tamanho m × n no qual todos os coeficientes apresentam valor 1/mn. Um filtro de média atenua variações locais em uma imagem, e o ruído é reduzido em consequência do borramento.

2.3.5.2 Filtros de Média Geométrica

Cada pixel restaurado é determinado pelo produto dos pixels na janela de sub imagem, elevado à potência de 1/mn. Um filtro de média geométrica obtém uma suavização comparável ao filtro de média aritmética, mas tende a perder menos detalhes da imagem no processo. A equação 4 ilustra uma imagem restaurada por esse filtro, onde *g (s, t)* é o valor médio da imagem corrompida. (GONZALES; WOODS, 2010)

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}} \quad (4)$$

2.3.5.3 Filtro de média harmônica

O filtro de média harmônica funciona bem para o ruído de sal, mas falha para o ruído de pimenta. Ele também apresenta um bom desempenho com outros tipos de ruído, como o gaussiano. A operação de filtragem de média harmônica é determinada pela equação 5, onde *g (s, t)* é o valor médio da imagem corrompida. (GONZALES; WOODS, 2010)

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}} \quad (5)$$

2.3.5.4 Filtro de mediana

O filtro de mediana, substitui o valor de um pixel pela mediana dos níveis de intensidade na vizinhança desse pixel, conforme a equação 6, onde *g (s, t)* é o valor médio da imagem corrompida.

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{mediana}} \{g(s,t)\} \quad (6)$$

O valor do pixel em (x, y) é incluído no cálculo da mediana. Os filtros de mediana são bastante populares porque, para certos tipos de ruído aleatório, eles proporcionam excelentes recursos de redução de ruído, com consideravelmente menos borramento do que os filtros lineares de suavização de tamanho similar. Os filtros de mediana são particularmente eficazes na presença do ruído tanto impulsivo bipolar quanto unipolar. Com efeito, o filtro de mediana gera excelentes resultados para imagens corrompidas por esse tipo de ruído. (GONZALES; WOODS, 2010)

2.4 COMPUTAÇÃO VISUAL

Essa área trata Extração de informações de imagens e identificação e classificação de objetos nesta imagem. Os sistemas de visão computacional vêm sendo usados em projetos que envolvem o reconhecimento de pessoas, de assinaturas e de objetos; inspeção de peças em linhas de montagem; orientação de movimentos de robôs em indústrias automatizadas. Utiliza técnicas de inteligência artificial (IA) ou de tomada de decisões que permitem a identificação ou mesmo a classificação de imagens ou objetos. (AZEVEDO, 2008):

Um sistema de visão computacional é organizado a partir da aplicação que ele irá exercer. Entretanto, existem funções típicas que são encontradas nos sistemas de visão computacional:

- Pré-processamento: antes de um método de visão computacional ser aplicado em uma imagem para extrair informação, é geralmente necessário processar a imagem para assegurar-se que ela satisfaz as condições do método. Aplicações dessa etapa são mapeamento, redução de ruídos;
- Extração de características: Nesta etapa são retiradas características matemáticas da imagem em vários níveis de complexidade são extraídas. Processos como detecção de bordas, cantos ou pontos ocorrem neste ponto do processo;
- Detecção e segmentação: é realizada um processo de decisão sobre a relevância de regiões da imagem para processamento posterior. Nesta etapa incluem processo de seleção de regiões de interesse específicos e segmentação de uma ou mais regiões que contém um objeto de interesse;
- Processamento de alto nível: a entrada é geralmente um conjunto pequeno de dados. O processo posterior inclui a verificação da satisfação dos dados, a estimativa de parâmetros sobre a imagem e a classificação dos objetos detectados em diferentes categorias.

Ao desenvolver algoritmos de Visão Computacional, é preciso enfrentar diferentes problemas e desafios, relacionados à própria natureza dos dados ou a evento que serão gerados pela aplicação, dentre estes desafios estão (HAYO, 2017):

- Dados ruidosos ou incompletos;
- Processamento em tempo real;
- Recursos limitados: poder, memória.

A Figura 9 ilustra uma analogia entre a visão humana e a visão computacional.

Figura 9 – Analogia sistema de visão humana e computacional

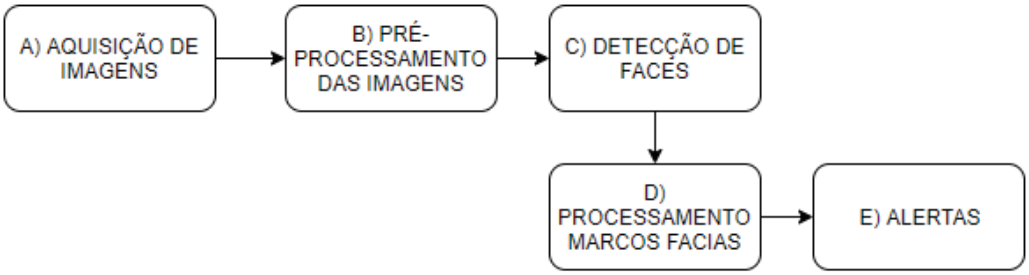
Fonte: Azevedo (2008, p. 63)

3 MATERIAIS E MÉTODOS

Nesse capítulo, será apresentado de forma detalhada uma descrição dos materiais e softwares que compõem o projeto. A síntese do sistema proposto por ser visualizada na Figura 10.

A ideia, do projeto, é desenvolver um sistema que seja capaz de identificar, por meio de técnicas de processamento digital de imagens e de visão computacional, sinais de sonolência. A identificação desses sinais deverá ocorrer de forma rápida e mais precisa possível, para que detecção da sonolência esteja dentro do esperado.

Figura 10 – Diagrama de funcionamento do projeto



Fonte: Próprio autor.

Nos itens abaixo, serão colocadas breves descrições sobre os blocos do diagrama da Figura 10:

A. Aquisição de imagens: Este bloco é responsável pela aquisição das imagens a partir de uma câmera;
B. Pré-Processamento: Neste bloco são realizados os processos de pré-processamento das imagens adquiridas pela câmera, nesses processos constam ajuste, correções, conversões nas imagens que foram coletadas;
C. Detecção de faces: Nesta etapa é implementado o algoritmo de Violas Jones para a detecção de faces, utilizando a biblioteca opencv além de um arquivo pré-treinado para o processo de detecção;
D. Processamento dos pontos de referência faciais: Neste bloco é implementado a localização de coordenadas dos pontos de referência faciais, além de ajustar a imagens e aplicar as equações para o cálculo da relação de aspecto do olho e da boca;
E. Alertas: Nesta etapa são implementados os sinais alerta ao usuário caso seja detectado sinais de sonolência.

3.1 ALGORITMO DE VIOLA JONES

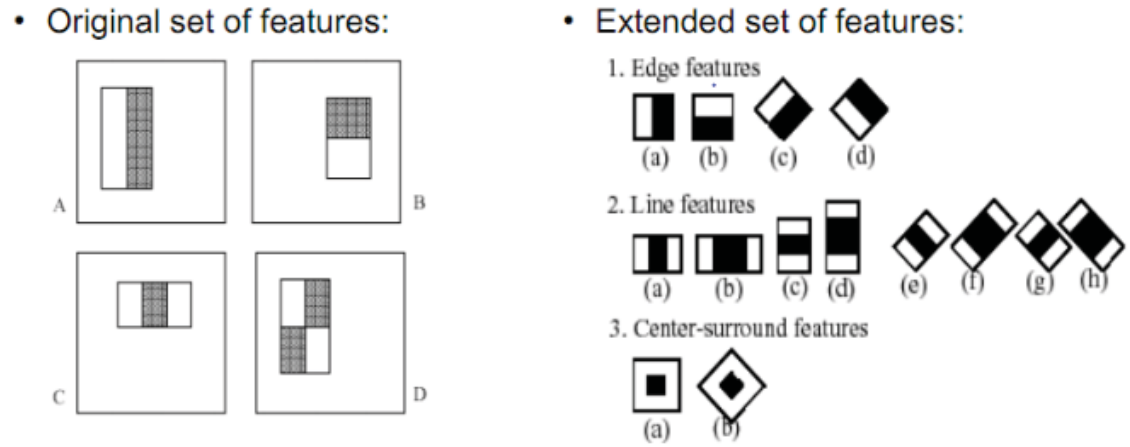
Técnica de reconhecimento de padrões, desenvolvida de por Paul Viola e Michael Jones, que consiste em encontrar frações da imagem que apresentam variação ou diferenças de contraste constante. Essas frações são denominadas como características, e são encontradas durante o processo de treinamento. Esse algoritmo possui como característica bons resultados na detecção de objeto, resultantes de uma técnica de treinamento que demanda muitos recursos computacionais e que por consequência deixa o processo de detecção mais simples.

3.1.1 Características Haar

As características Haar são a base dos sistemas de detecção de objetos utilizando o algoritmo de Viola Jones. Ela consiste na diferença de intensidade entre regiões retangulares de uma imagem, essa diferença se dá pela soma pixels que situam de dentro uma área retangular e são subtraídos pela soma dos pixels situados na outra área retangular. Essas características foram baseadas na transformada de Haar. (RODRIGUES, 2012)

Na Figura 11 observa-se as características Haar, que são utilizadas para extração de características de uma imagem para posteriormente servir no processo de detecção de objetos.

Figura 11 – Características Haar

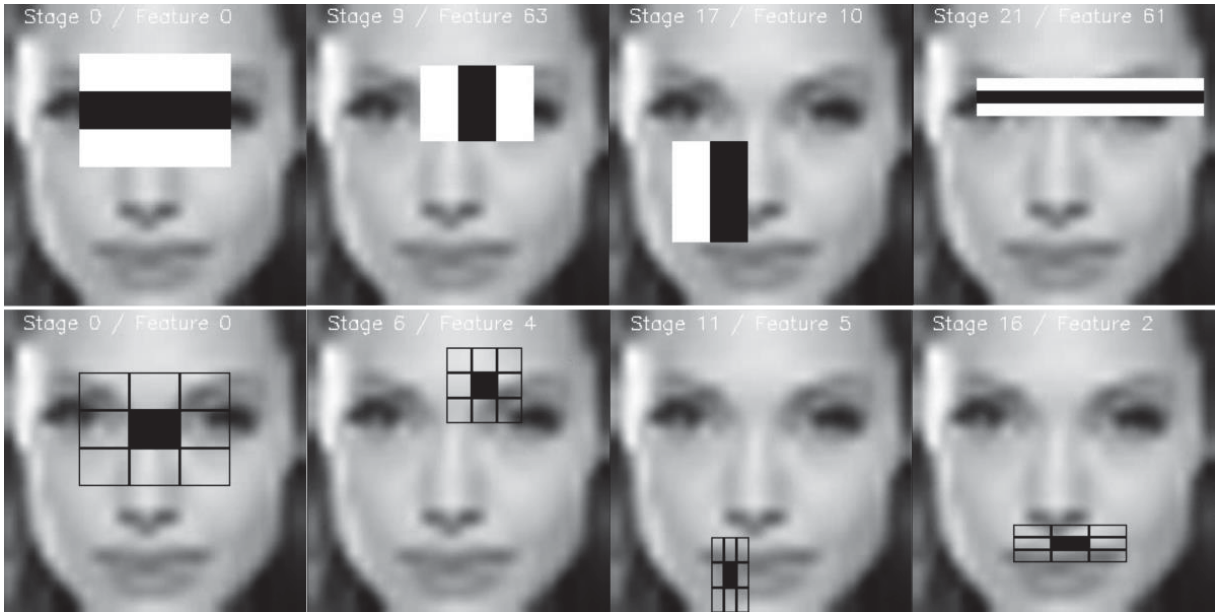


Fonte: https://stackoverflow.com/questions/5808434/how-does-the-viola-jones-face-detection-method-work

Dentre as vantagens do uso das características Haar está a eficiência computacional, a independência da escala de cor além do fato de ser orientada a diferença de intensidade locais. As relações mútuas de intensidade de várias regiões pouco variam à medida que a intensidade absoluta de diferentes regiões da imagem muda drasticamente. (RODRIGUES, 2012)

A Figura 12 ilustra como é feito o processo de extração de recursos de uma imagem utilizando as características Haar.

Figura 12 – Extração dos recursos de uma imagem



Fonte: https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1

No exemplo da figura 12, o primeiro recurso mede a diferença na intensidade entre a região dos olhos e a região superior da bochecha. Esse processo será aplicado como um núcleo convolucional sobre toda a imagem. Mesmo em imagens pequenas o número de recursos utilizado é gigantesco, devido ao extenso processo matemático necessário para retirar as características de uma determinada região. O cálculo dos recursos utilizados para a extração das características Haar é demonstrado na equação (7). (FABIEN, 2019)

$$Recurso(retângulo) = \Sigma pixels_{areabranca} - \Sigma pixels_{areapreta} \quad (7)$$

3.1.2 Imagem Integral

A computação dos recursos do retângulo em um estilo de núcleo convolucional pode ser longa. Por essa razão, Viola e Jones, propuseram uma representação intermediária para a imagem: a imagem integral. O papel da imagem integral é permitir que qualquer soma retangular seja computada simplesmente, usando apenas quatro valores. (FABIEN, 2019)

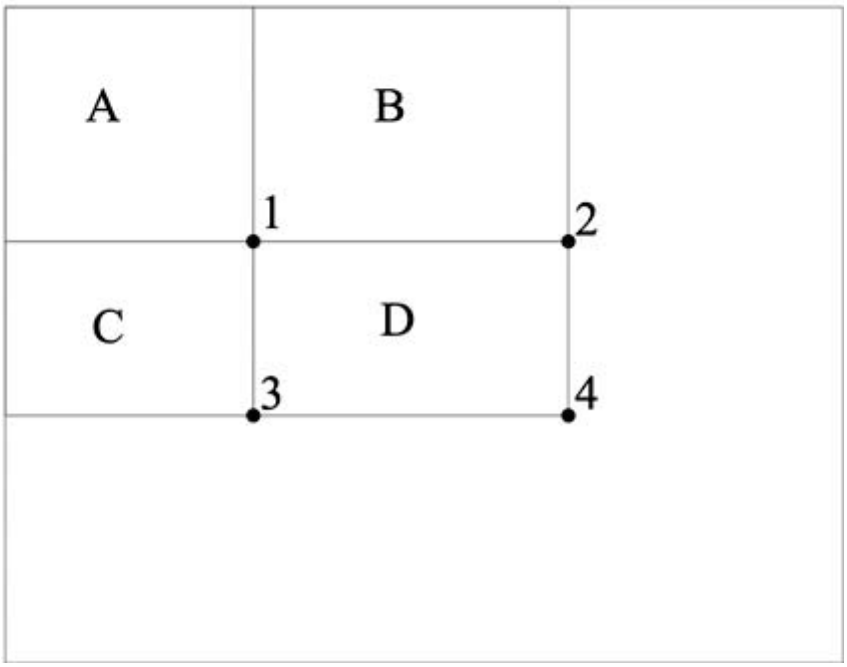
A estrutura de dados, consiste no somatório de todos os pixels que estejam acima e posicionados à esquerda do ponto onde se deseja o valor. Formalmente, Viola e Jones definiram a imagem integral pela seguinte relação recursiva:

$$\begin{aligned} &ii(-1, y) = 0 \\ &s(x, -1) = 0 \\ &s(x, y) = s(x, y\text{-}1) + i(x, y) \\ &ii(x, y) = ii(x\text{-}1, y) + s(x,y) \end{aligned} \tag{8}$$

Onde s (x, y) é a soma da linha cumulativa no ponto (x, y), ii (x, y) é o valor da imagem integral no mesmo ponto e i (x, y) é o valor do pixel nesse ponto.

Fazendo uso da imagem integral, se torna possível calcular o somatório dos pixels de qualquer forma retangular em tempo constante, fazendo-se a leitura somente de 4 pontos. Usando a Figura 13 como exemplo, para se obter a soma de todos os pixels do retângulo D, usando uma imagem integral, somente se necessita saber dos valores desta nos pontos 1, 2, 3 e 4. O valor da imagem integral no ponto 1 é a soma dos píxels no retângulo A, o valor no ponto 2 é A+B, no ponto 3 é A+C e no ponto 4 é A+B+C+D. Então, o somatório D pode ser computado como 4+1-(2+3). (FABIEN, 2019)

Figura 13 – Imagem integral



Fonte: https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1

3.1.3 CLASSIFICADORES E CASCATAS HAAR

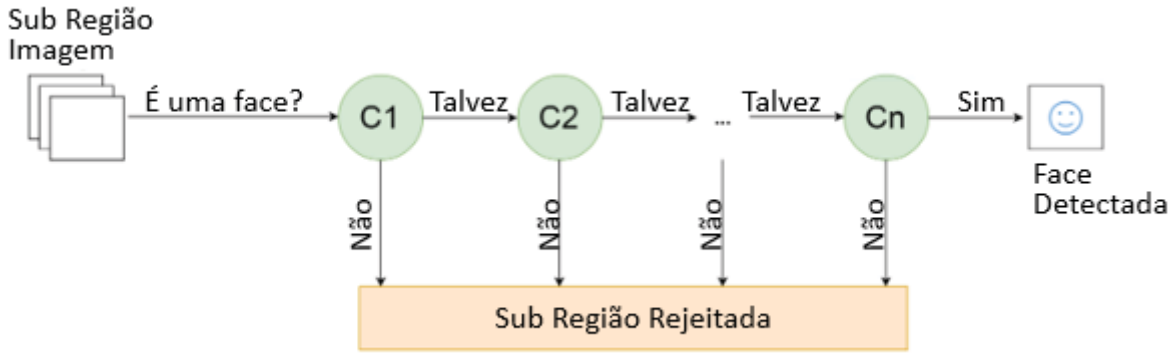
Esta cascata é formada por estágios, formados por classificadores fortes, que por sua vez contêm um conjunto de classificadores fracos. Um classificador forte é formado por classificadores fracos e um valor de limiar. Um classificador fraco é formado de uma ou mais características, e seus respectivos pesos. Os classificadores são denominados fracos por não haver garantias de que sua classificação seja ótima. (RODRIGUES, 2012)

Os pesos e o limiar são resultados de um processo de treinamento. Neste processo de treinamento é empregado o algoritmo AdaBoost, o qual escolhe as características que melhor identificam o objeto em questão. (RODRIGUES, 2012)

Este processo de treinamento consiste em sucessivas iterações, e, ao final de cada uma, os pesos de cada classificador fraco são recalculados, penalizando por falsa detecção e reforçando por sucesso, este processo também é conhecido por boosting. A condição de parada do treinamento é atingida ao alcançar um determinado limite de tempo, quantidade de iterações ou por convergência dos pesos calculados. O teste de todas as possíveis características é a parte mais custosa em relação ao tempo de todo processo. (RODRIGUES, 2012)

Um estágio, por sua vez, é formado por uma coleção ordenada de classificadores fortes, que são testados sequencialmente, em tempo de detecção, e descartam o processamento dos demais estágios, caso uma janela não obedeça ao seu limiar definido no treinamento. A saída de um estágio é a entrada do posterior, que usa de classificadores mais complexos. Esta sequência é demonstrada na Figura 14. (RODRIGUES, 2012)

Figura 14 – Classificador em cascata



Fonte: Próprio Autor

Já a cascata é a concatenação, em uma sequência bem definida e ordenada, de todos os estágios. É desenvolvida de forma que a grande parte dos resultados negativos seja obtida nos primeiros estágios - que são projetados para utilizar testes mais simples, e assim, descartar áreas que não contenham o objeto tão cedo quanto possível, reduzindo a necessidade de processamento mais complexo. Estes estágios buscam ocorrências de "não-objetos", que são mais fáceis de detectar que objetos. A configuração dos estágios iniciais tem como objetivo direto a maior eficiência do detector. (RODRIGUES, 2012)

A execução do detector aqui descrito, é feita pela seleção de um tamanho de janela, que então desliza na imagem e analisa cada fragmento da imagem delimitada pela janela deslizante. Para cada uma delas, executa toda a cascata - esta execução pode ser interrompida, caso um estágio rejeite a janela atual. O passo, em pixels, definido para determinar a próxima janela é importante. Ele pode aumentar o desempenho do classificador, quando aumentado. Mas, como efeito negativo, pode saltar objetos se definido demasiadamente grande. É recomendada uma análise do objeto em função da imagem a ser processada. (RODRIGUES, 2012)

3.2 DETECÇÃO DE PONTOS DE REFERÊNCIA FACIAIS

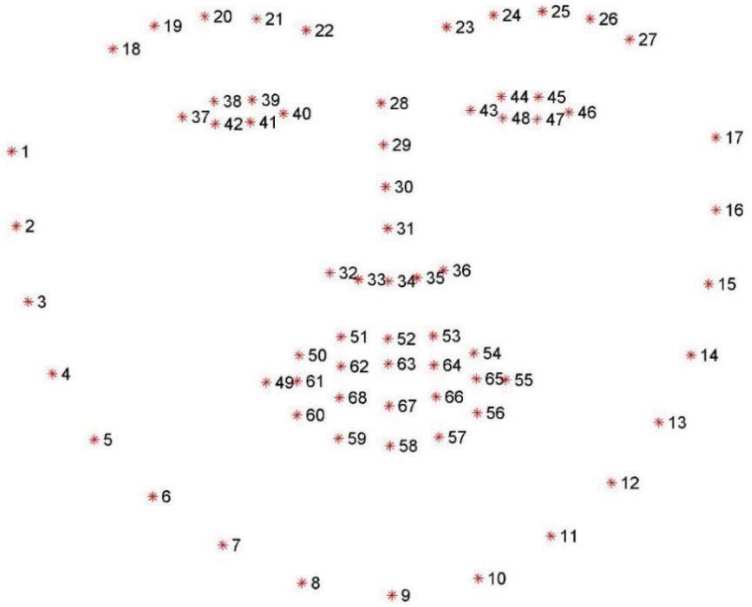
A detecção de pontos de referência faciais ou simplesmente marcos facias (*Facial Landmarks*) é identificada como uma técnica que busca localizar e identificar pontos característicos (como olhos, boca, ponta do nariz) com um alto índice de precisão.

Afim de popularizar essa técnica da análise foi implementada na biblioteca DLIB, que é uma biblioteca muito utilizada em projeto envolvendo computação visual e processamento digital de imagens, um detector de pontos de referência faciais que é utilizado para estimar a localização de 68 coordenadas(x, y) que mapeiam as estruturas faciais no rosto de uma pessoa como visto na Figura 15.

Essas 68 coordenadas foram obtidas a partir do treinamento de um modelo de predição de formas chamado *iBUG300-W*, os dados são armazenados em vetor que pode ser utilizado de forma específica apenas escolhendo o intervalo necessário. O vetor está caracterizado da seguinte forma:

- Olho direito: [36, 41];
- Olho esquerdo: [42, 37];
- Boca: [48, 68];
- Nariz: [27, 35].

Figura 15 – Facial Landmarks



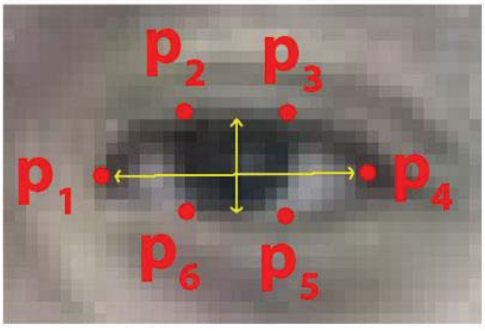
Fonte: https://medium.com/brasil-ai/mapeamento-facial-landmarks-com-dlib-python-3a200bb35b87

3.3 EYE ASPECT RATIO

O piscar dos olhos é um movimento natural de abertura e fechamento de forma rápida de um olho humano. Esse padrão de piscada é diferente em cada pessoa, a variação pode estar relacionada com a velocidade em que ocorrem os movimentos assim alterando a intensidade a qual esse movimento é aplicado. Geralmente, o tempo de uma piscada varia entre 100 a 400 ms. (CECH; SOUKUPOVA, 2016)

A relação de aspecto do olho (EAR – *Eye aspect ratio)* é um método de análise de características fisiológicas do ser humano que é aplicada utilizando os Facials Landmarks, esses pontos faciais são importantes para reconhecer características importantes. A Figura 16 ilustra os pontos de interesse concentrados nos olhos. (FERREIRA; MARQUEZ; JUNIOR, 2019)

Figura 16 – Pontos de referência olho aberto



Fonte: Adaptado de https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

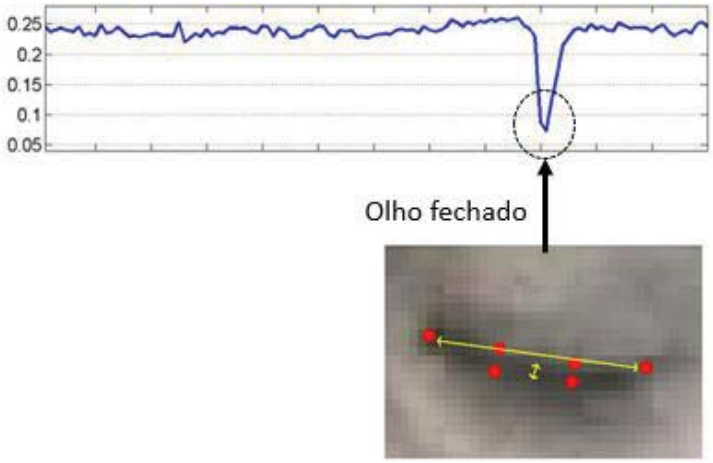
A medida do EAR é calculada a partir da dos pontos destacados na Figura 16, ou, seja é calculada levando em consideração a altura e a largura do olho. Os P1 a P6 coordenadas ilustradas em um plano 2D, elas são representadas pelos intervalos [36, 41] e [42, 47] do vetor dos Facial Landmarks que são calculados pela expressão abaixo:
(FERREIRA; MARQUEZ; JUNIOR, 2019)

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|} \tag{9}$$

O numerador dessa equação calcula a distância entre os pontos de referência verticais dos olhos enquanto o denominador calcula a distância entre os pontos horizontais dos olhos, ponderando adequadamente o denominador, pois há apenas um conjunto de pontos horizontais, mas dois conjuntos de pontos verticais. (CECH; SOUKUPOVA, 2016)

O valor do EAR é constante quando um olho permanece em um estado e se altera a partir do momento em que o olho pisca. Esse valor pode variar de pessoa para pessoa, sendo mais indicada uma parametrização de qual limiar usar para cada indivíduo. Pois a medida EAR varia de acordo com à proporção que cada olho possui, geralmente esse valor para o olho aberto varia de 0,25 a 0,32. Para valores abaixo deve-se considerar que o indivíduo está entrando em estado de sonolência, a Figura 17 demonstra esse comportamento.
(FERREIRA; MARQUEZ; JUNIOR, 2019)

Figura 17 – Gráfico do EAR olho fechado.



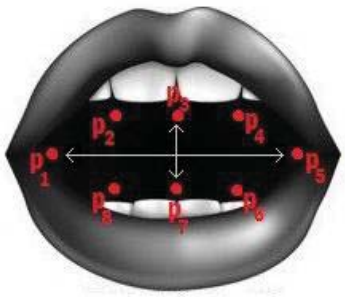
Fonte: Adaptado de https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pd

3.4 MOUTH ASPECT RATIO

A relação de aspecto da boca (MAR – *mouth aspect ratio*) é mais uma técnica para detectar sinais fisiológicos em seres humanos. Ela utiliza dos pontos de referência na face para calcular o a relação de abertura de boca. Esses dados podem ser de grande utilidade para eventual detecção de sinais de sonolência.

As coordenadas dos pontos na região da boca podem ser visualizadas na Figura 18.

Figura 18 – Ponto de referência na região da boca



Fonte: Adaptado de https://towardsdatascience.com/mouse-control-facial-movements-hci-app-c16b0494a971

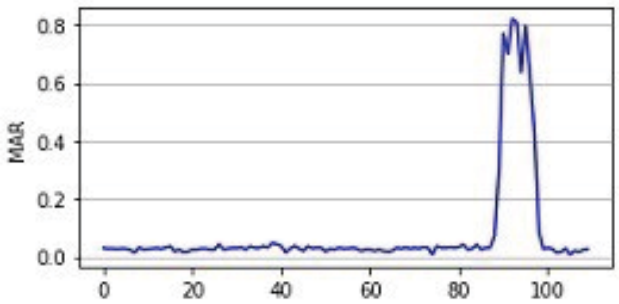
Para o cálculo do MAR é utilizada a seguinte equação:

$$\mathbf{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{3\|p_1 - p_5\|} \tag{10}$$

De forma análoga a ao cálculo do EAR, o numerador dessa equação é responsável pelo cálculo da distância entre os pontos de referência verticais da boca, enquanto o denominador realiza o cálculo dos pontos de referência horizontais da boca, outro aspecto do denominador é que existem um ajuste para compensar o menor número de referência se comparado com o numerador. A medida do MAR varia de acordo com à proporção que cada boca possui, normalmente esse valor encontrasse entre 0,4 e 0,6. Para valores maiores considerasse que o indivíduo está bocejando e consequentemente apresentando sinais de sonolência.

A Figura 19 demonstra um gráfico da relação de aspecto da boca

Figura 19 – Gráfico da relação de aspecto da boca (MAR)



Fonte: Adaptado de https://towardsdatascience.com/mouse-control-facial-movements-hci-app-c16b0494a971

3.5 RASPBERRY PI

O Raspberry pi é um computador de placa única (SBC - *Single Board Computer*), que possui um tamanho reduzido e um baixo custo. Roda um sistema operacional completo (conta com editor de texto, capacidade de visualizar imagens ou mesmo reproduzir vídeos) baseado em uma versão customizada do Linux, o Raspian (MONK, 2013).

As características técnicas do dispositivo estão descritas no Quadro 1.

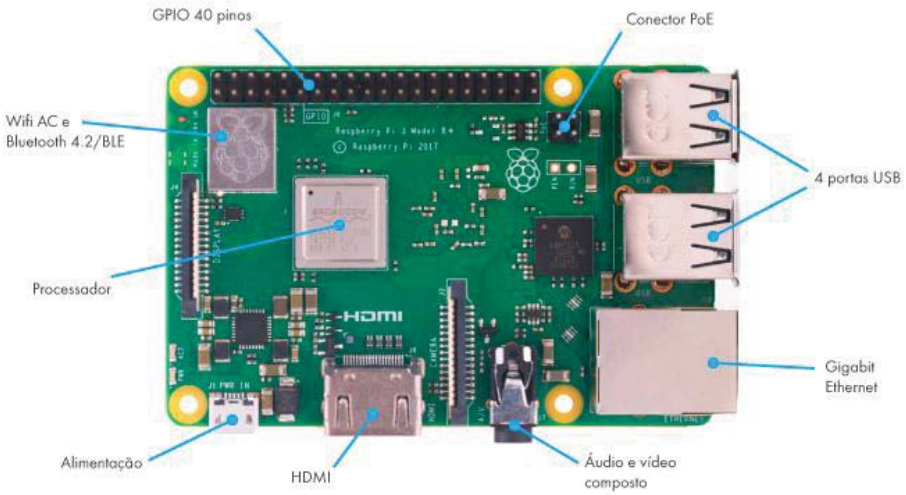
Quadro 1 - Especificações técnicas do Raspberry Pi 3 model B+:

SoC	Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
Gpu	Broadcom Videocore-IV
Ram	1GB LPDDR2 SDRAM
Wifi	2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
Bluetooth	Bluetooth 4.2, Bluetooth Low Energy (BLE)
Armazenamento	Cartão MicroSD
GPIO	40 Pinos
HDMI	1x
Conector Ethernet	1x
Conector Áudio e Vídeo	1x
Interface serial para Câmera (CSI)	1x
Interface serial para Display (DSI)	1x
USB	4x
Tensão	5v
Corrente de Alimentação	2,5A
Dimensões	82mm x 56mm x 19.5mm
Peso	50g

Fonte: Adaptado de https://www.raspberrypi.org/magpi/raspberry-pi-3bplus-specs-benchmarks/

A estrutura física que compõem a placa Raspberry Pi 3 Model B+, como as portas USB, conector CSI e DSI, os pinos de GPIO e a porta de alimentação da placa podem ser vistos na Figura 20.

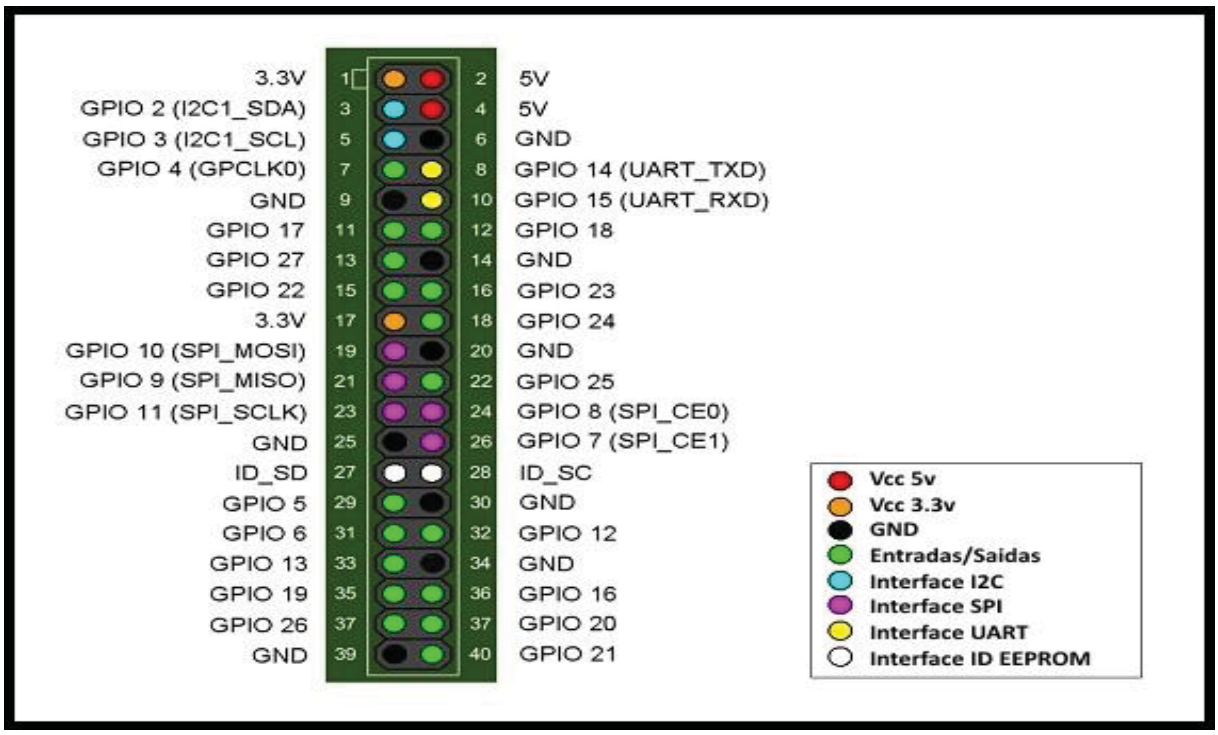
Figura 20 – Raspberry Pi 3 Model B+



Fonte: https://www.filipeflop.com/blog/lancamento-raspberry-pi-3-model-b-plus/

A Figura 21 mostra a função de cada um dos pinos da GPIO.

Figura 21 – Raspberry Pi 3 Model B+ GPIO



Fonte: https://www.filipeflop.com/blog/tutorial-raspberry-pi-linux/

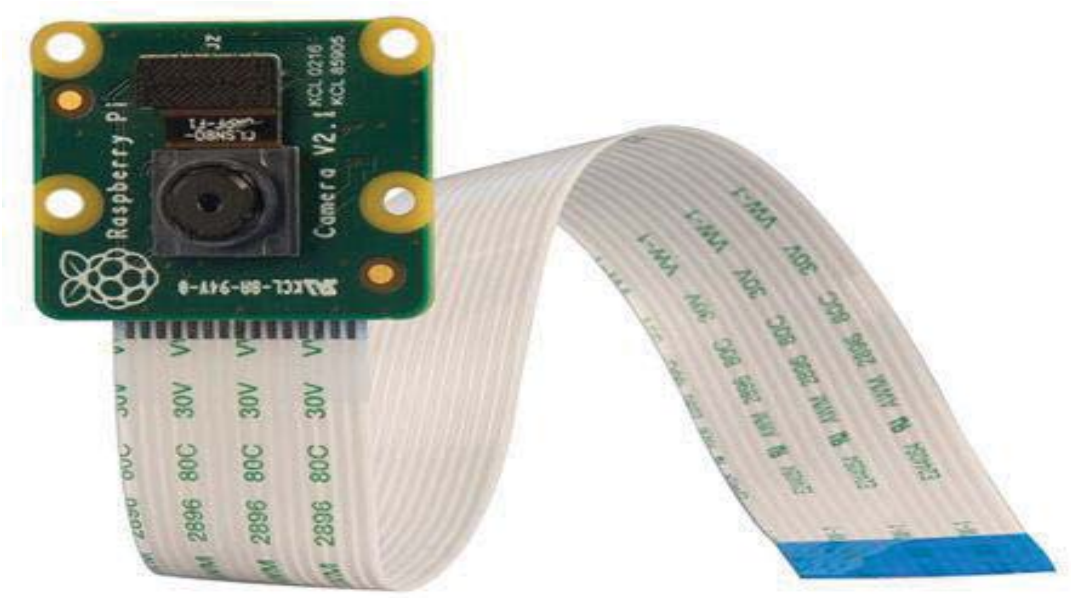
3.6 RASPBERRY PI CÂMERA

O Módulo de Câmera V2, é o segundo módulo desenvolvido para a plataforma Raspberry Pi, ele possui um sensor IMX219 de 8 megapixels da Sony. Ele pode ser usado para capturar vídeos em alta definição e fotos estáticas, sendo que existem aplicações que utilizam este dispositivo para realizar captura de vídeos em câmera lenta ou mesmo realizar time lapse.

Este módulo possui características possuiu menor latência se comparado com câmeras USB, possui filtros *Anti-aliasing* para evitar possíveis interferência cromáticas, utiliza um sensor de imagem do tipo CMOS. Pode capturar vídeos na resolução 1080p em até 30FPS, podendo chegar até 90 FPS na resolução 640x480 pixels. Sua conexão com o Raspberry ocorre por meio da porta CSI.

A Figura 22 mostra o módulo de câmera V2.

Figura 22 – Modulo de Câmera V2 8MP



Fonte: https://www.raspberrypi-spy.co.uk/2016/04/8-megapixel-raspberry-pi-camera-module-v2/

No quadro 2 são apresentadas as características do modulo de câmera Raspberry Pi câmera V2.

Quadro 2: Características Técnicas do Módulo V2

Resolução	8 Megapixels
Modos de vídeo	1080p30, 720p60 e 640 x 480p60/90
Sensor	Sony IMX219
Resolução do Sensor	3280 x 2464 pixels
Área Sensor	3.68 x 2.6 mm (4.6 mm diagonal)
Tamanho do Pixels	1.12µm x 1.12 µm
Abertura	1/4"
Comprimento Focal	3.04 mm
Campo de visão horizontal	62.2 graus
Campo de Visão Vertical	48.8 graus
Raio Focal	2.0

Fonte: https://www.raspberrypi.org/documentation/hardware/camera/README.md

3.7 ALERTAS

No sistema de alertas foram implementados avisos sonoros e visuais. Para os avisos sonoros foi conectado um Buzzer em um dos pinos de GPIO do Raspberry Pi. Já os alertas visuais foram utilizados leds, para demostrar o status do sistema.

3.8 FONTE DE ALIMENTAÇÃO

Para a alimentação do microcomputador Raspberry Pi 3 e dos demais periféricos como a câmera, os leds, o buzzer foi utilizada uma fonte de alimentação de 5 V e 3 A. Esses requisitos de alimentação seguem as recomendações do fabricante e conseguem suprir as necessidades do projeto.

3.9 PYTHON

O Python é uma linguagem de programação de alto nível, orientada a objetos. Sua sintaxe é clara e concisa, além disso o Python é considerado uma linguagem livre, ou seja, pode ser empregado para o desenvolvimento de qualquer aplicação.

O Python possui muitas funções similares a linguagem C, porém apresenta um tempo de execução do programa superior ao um programa escrito em C. A linguagem de programação Python está presente no Raspberry Pi por meio do compilador chamado IDLE, possuindo duas versões o Python 2 e o Python 3, a diferença entre ambas as versões está relacionada a sintaxe.

3.10 BIBLIOTECAS DE PYTHON

Em Python existem inúmeras bibliotecas que auxiliam e simplificam o trabalho de programação. A seguir, foram indicadas as principais biblioteca open source utilizadas no projeto.

3.10.1 SciPy

SciPy é uma biblioteca open source de computação científica. Nela contém ferramentas para estatística, otimização, processamento de sinais e imagens, funções especiais como bessel, por exemplo. Seu funcionamento ocorre em conjunto com a biblioteca NumPy. (SCIPY, 2019)

3.10.2 NumPy

O NumPy é o pacote básico da linguagem Python que permite trabalhar com arranjos, vetores e matrizes de N dimensões, de uma forma comparável e com uma sintaxe semelhante ao software proprietário Matlab, mas com muito mais eficiência, e com toda a expressividade da linguagem. (PYSCIENCE, 2019)

3.10.3 Imutils

Uma série de funções de conveniência para tornar as funções básicas de processamento de imagens, como rotação, redimensionamento, esqueletização e exibição de imagens do Matplotlib mais fáceis com o OpenCV e o Python 3.

3.10.4 GPIO

Com a biblioteca GPIO ou gpiozero é possível realizar configurações e comandos nos 40 pinos de GPIO presentes no Raspberry Pi 3. Esses pinos podem ser setas como entradas e saídas, além de ser possível se comunicar com outros dispositivos utilizando comunicação como I2C, SPI ou UART.

3.10.5 OpenCV

O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de software de visão computacional e aprendizado de máquina em código aberto. Ela foi desenvolvida para fornecer uma infraestrutura comum para aplicações de visão computacional e para acelerar o uso da percepção da máquina nos produtos comerciais. (OPENCV, 2018)

A biblioteca tem mais de 2500 algoritmos otimizados, o que inclui um conjunto abrangente de algoritmos de visão computacional e de aprendizado de máquina clássicos e de última geração. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmera, rastrear objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos 3D a partir de câmeras estéreo, unir imagens para produzir uma resolução alta imagem de uma cena inteira, encontrar imagens semelhantes de um banco de dados de imagem, remover olhos vermelhos de imagens tiradas usando flash, acompanhar movimentos dos olhos, reconhecer paisagens e estabelecer marcadores para sobrepô-lo com realidade aumentada.
(OPENCV, 2018)

Juntamente com empresas bem estabelecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda e Toyota que empregam a biblioteca, existem muitas startups como a Applied Minds, VideoSurf e Zeitera, que fazem amplo uso do OpenCV. (OPENCV, 2018)

Os usos implantados da OpenCV abrangem desde unir imagens de ruas, detectar intrusões em vídeo de vigilância em Israel, monitorar equipamentos de minas na China, ajudar robôs a navegar e pegar objetos na Willow Garage, detectar acidentes com afogamentos em piscinas na Europa, executar arte interativa em Espanha e Nova York, verificando pistas em busca de destroços na Turquia, inspecionando rótulos de produtos de fábricas em todo o mundo para uma rápida detecção de rostos no Japão. (OPENCV, 2018)

A biblioteca OPENCV é estruturada da seguinte forma:

- **cv** — Módulo das principais funcionalidades e algoritmos de Visão Computacional do OpenCV;
- **cvaux** — Módulo com algoritmos de Visão, ainda está em fase experimental;
- **cvcam** — Módulo com os algoritmos para acessar as informações geradas por câmeras;
- **cxcore** — Módulo de Estrutura de Dados e Álgebra Linear;
- **highgui** — Módulo de Controle de Interface e dispositivos de entrada;

3.10.6 Dlib

O DLIB é uma biblioteca multiplataforma de código aberto que foi desenvolvida na linguagem C++, possui muitos algoritmos e ferramentas de aprendizado de máquina e de processamento digital de imagens. É muito utilizada em softwares de detecção e reconhecimento facial.

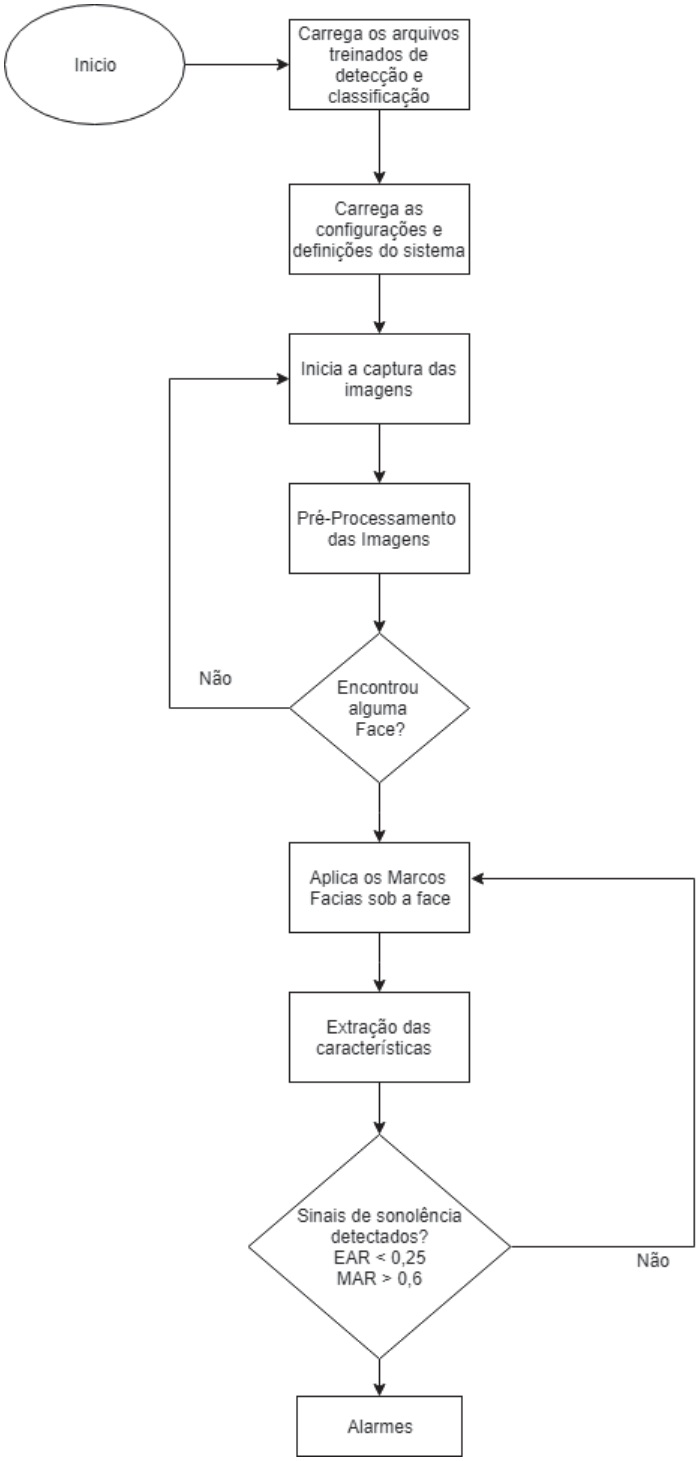
Possui componentes de software para trabalhar com:

- Redes;
- Threads;
- Interface gráfica para usuários;
- Estrutura de dados;
- Álgebra linear;
- Aprendizado de Máquina;
- Processamento digital de imagens.

4 DESENVOLVIMENTO DO PROJETO

Neste capítulo, será explicado como foi realizado o desenvolvimento do trabalho. O fluxograma do código pode ser visualizado na Figura 23.

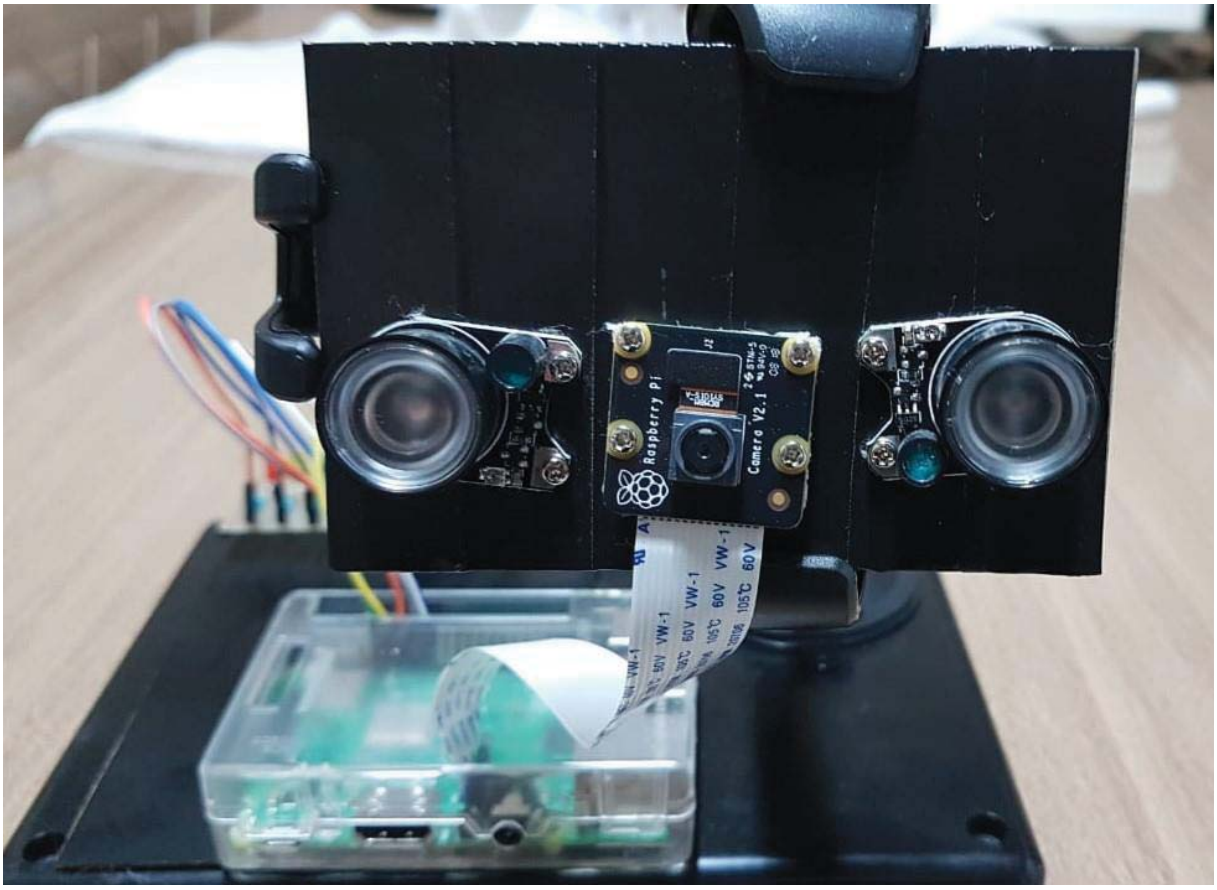
Figura 23 – Fluxograma do software



Fonte Próprio Autor

O projeto foi desenvolvido utilizando a microcomputador Raspberry Pi 3 model B+ e o módulo de câmera V2 Noir, esse modelo de câmera não possui filtros infravermelhos o que possibilita a utilização de leds IR para captura de imagens noturnas. Ambos os dispositivos foram montados em um suporte juntamente, com os leds e o buzzer, que compõem o sistema de alertas empregado neste projeto. A Figura 24 ilustra a montagem do protótipo desenvolvido para validar o projeto.

Figura 24 – Protótipo do Hardware



Fonte Próprio Autor

Para iniciar o desenvolvimento do software no Raspberry Pi deve-se primeiramente instalar o sistema operacional, o Raspian. Em seguida, é essencial a instalação e configuração de todas as bibliotecas necessárias como o OpenCV para o processamento de imagens, DLIB para os pontos de referência faciais, Imutils para a captura das imagens além de bibliotecas auxiliares como gpiozero, NumPy. O apêndice A demonstra as mudanças realizadas na biblioteca Imutils para implementação no projeto. Além disso, deve-se realizar a transferência dos arquivos treinados de detecção e classificação, que serão utilizados no processo de detecção da face e na obtenção dos Facial Landmarks.

Com o processo de importação das bibliotecas concluído, é realizada a definição das equações utilizadas para a obtenção dos valores e parâmetros, a partir da biblioteca Imutils, para o cálculo do EAR, MAR e dos demais pontos de interesse detectados na face humana. Nesta etapa ainda é realizada a configuração dos valores de referência, como os limites para cada parâmetro além da quantidade de repetições dele, que serão utilizados pelo sistema no processo de análise para identificar ou não sinais de sonolência.

Uma vez que essa etapa de configuração foi concluída inicia-se a captura em loop do vídeo, o sistema irá coletar os frames que foram capturados pela câmera e que se encontram em outro núcleo do processador, e vai redimensiona-lo para que as etapas posteriores de processamento de imagens possam ocorrer sem uma grande demanda por parte do hardware.

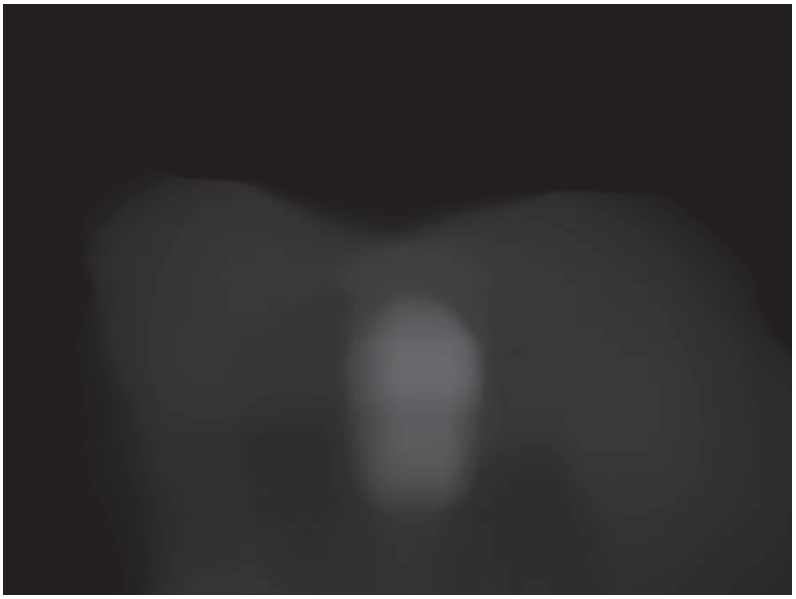
É realizada uma adequação nas imagens capturadas, onde paralelamente ocorre a conversão da imagem original (capturada em formato BGR) para escala de cinza e para o espaço de cor L.A.B, que será utilizada para corrigir possíveis interferência que a iluminação, na forma de um gradiente, possa ter no sistema. Essa conversão é demonstrada na Figura 25.

Figura 25 – Processo de conversão: A) Imagem original. B) Imagem em escala de cinza. C) Imagem no espaço L.A.B

Fonte Próprio Autor

No processo de conversão de BGR para L.A.B é isolado o canal L, que é o indicativo de luminosidade, após essa separação é aplicado um filtro de mediana que serve para obter as informações da iluminação gradiente presentes na imagem, conforme a Figura 26.

Figura 26 – Filtro de mediana na imagem L.A.B



Fonte Próprio Autor

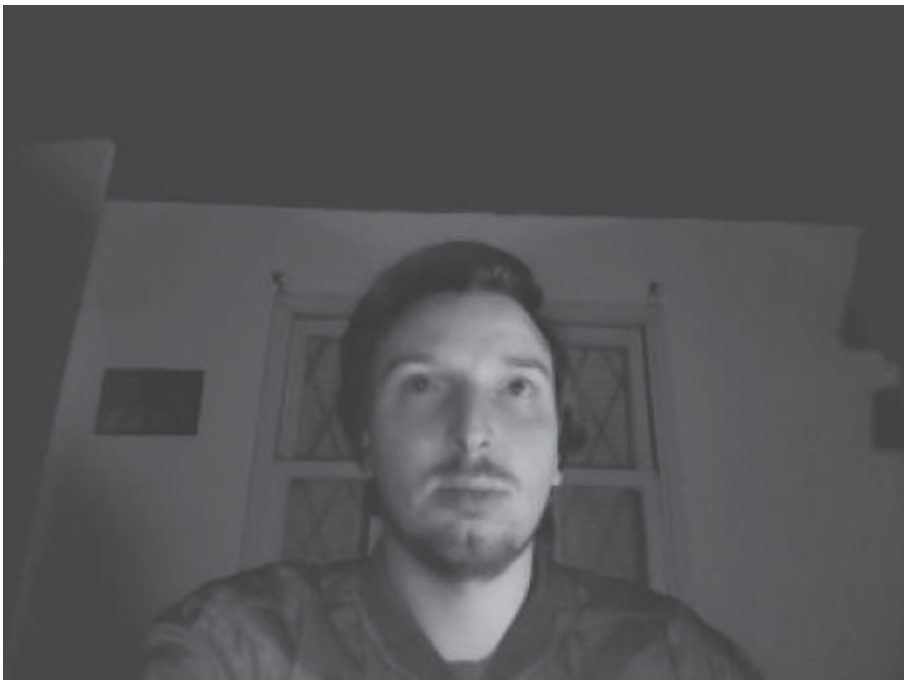
Com base nos dados obtidos pelo filtro de mediana é criado um canal L na imagem convertida para o formato L.A.B. Porém, esse novo canal tem a sua iluminação invertida, essa situação é ilustrada na Figura 27.

Figura 27 – Iluminação invertida

Fonte Próprio Autor

A partir das informações coletadas nesse novo canal L com iluminação invertida, é realizada a combinação deste recém-criado objeto com a imagem original convertida para escala de cinza. Essa combinação resulta em uma nova imagem, sem os efeitos da iluminação gradiente, que será utilizada nas demais etapas do sistema de processamento de imagens. Essa nova forma pode ser visualizada na Figura 28.

Figura 28 – Nova imagem gerada



Fonte Próprio Autor

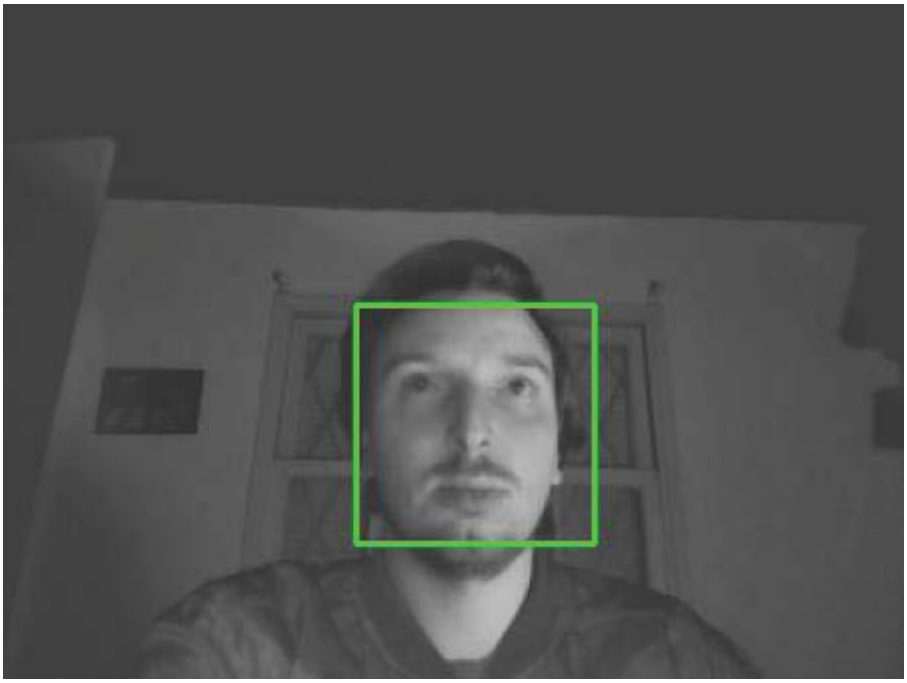
Após adequar as imagens coletadas é iniciado o processo de detecção facial, para isso é empregado o algoritmo proposto por Viola e Jones e que está implementado na biblioteca OpenCV por meio da função cv2.CascadeClassifier. Essa função possui 6 parâmetros de entrada e tem como retorna na saída uma sequência de retângulos, e que cada um desses retângulos representa um rosto. As entradas são:

- *Image*: A imagem onde será realizada a busca;
- *Cascade*: O classificador do tipo Haar Cascade, na forma de arquivo xml, que será lido com auxílio do OpenCV;
- *Storage*: Espaço alocado na memória para armazenar a sequência de resultados originado do processo de detecção do objeto
- *Scale* fator: Taxa pela qual a janela de busca da função será aumentada para percorrer a imagem novamente;

- *Min neighors*: Faz o agrupamento de retângulos vizinhos, da região de detecção, esse parâmetro é utilizado para mesclar retângulos de características similares;
- *Min Size*: Valor em pixels da menor janela de busca, sua largura e altura. Como parâmetro de saída temos os dados do vértice superior direito do retângulo.

Os valores que serão utilizados no detector devem levar em conta a quantidade de faces detectadas em uma imagem, neste caso será detectada apenas uma face. Outra característica é que o retângulo, utilizado na localização da face, deve ser maior que o rosto real pois a sua área será utilizada como delimitadora para as outras do sistema. A Figura 29 ilustra a detecção de faces.

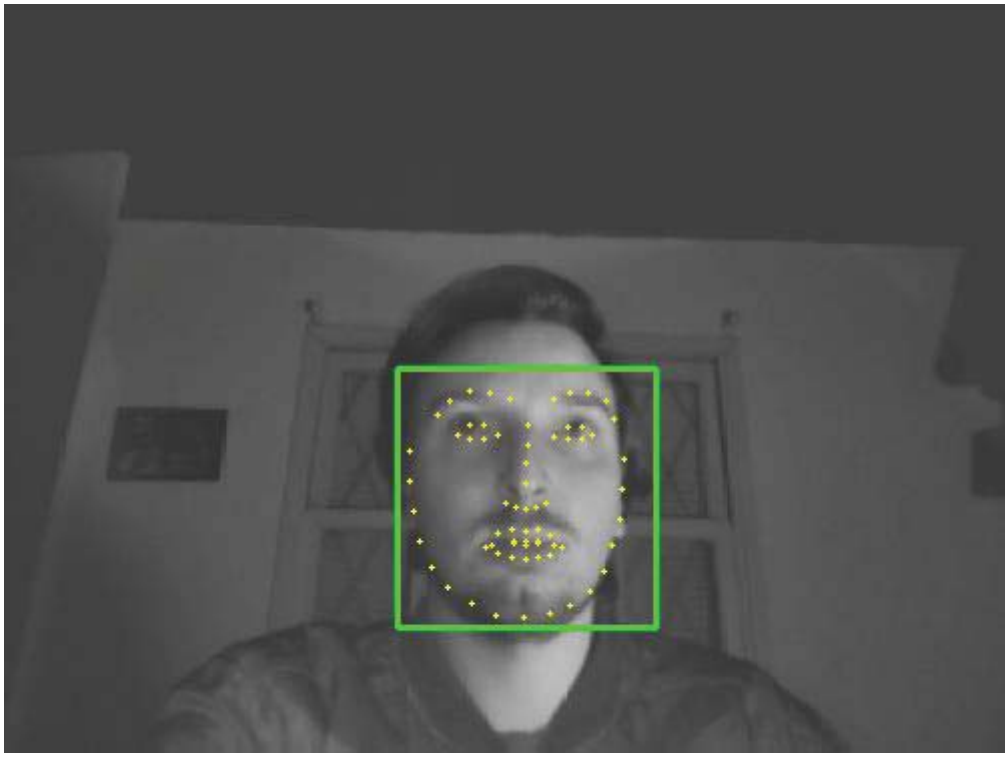
Figura 29 – Face detectada



Fonte Próprio Autor

Com a área de atuação já delimitada, é iniciado um loop para extrair as coordenadas, da altura e da largura. Com esses parâmetros é possível determinar os 68 pontos de referência faciais e converter as coordenadas (x, y) em uma matriz, utilizando a biblioteca Numpy. Os 68 pontos de referência estão definidos na Figura 30.

Figura 30 – Face com facial Landmarks



Fonte Próprio Autor

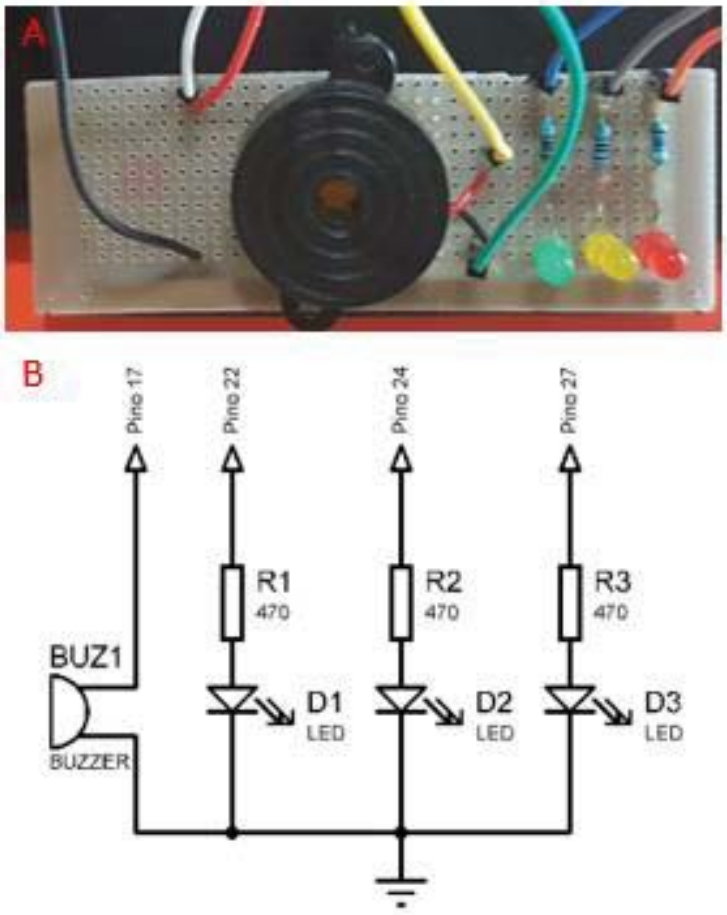
A partir dos dados armazenados na matriz serão realizadas as extrações dos valores referentes a áreas especificas da face. No projeto essa região são os olhos, a boca, o nariz e o queixo. Cada um dos valores encontrados nessas regiões está vinculado a um índice, e este é utilizado para separar os pontos horizontais e verticais que compõem essa região e assim calcular os parâmetros EAR e MAR.

A partir dos parâmetros encontrados e dos contadores que foram previamente configurados, o software irá analisar se o indivíduo está demonstrando sinais de sonolência ou não. Caso for identificado que existem sinais de sonolência, um alerta sonoro e visual notificara o usuário sobre essa situação. As condições para que o sistema de alarme entre em operação são:

- Parâmetro ear < ear_limite ,
- Paramtro mar > mar_limite
- Parametro dist < dist_limite

A Figura 31 ilustra o sistema de alerta que foi elaborado.

Figura 31 – Sistema de alertas: A) Placa montada. B) Esquematico



Fonte Próprio Autor

5 RESULTADOS E DISCUSSÕES

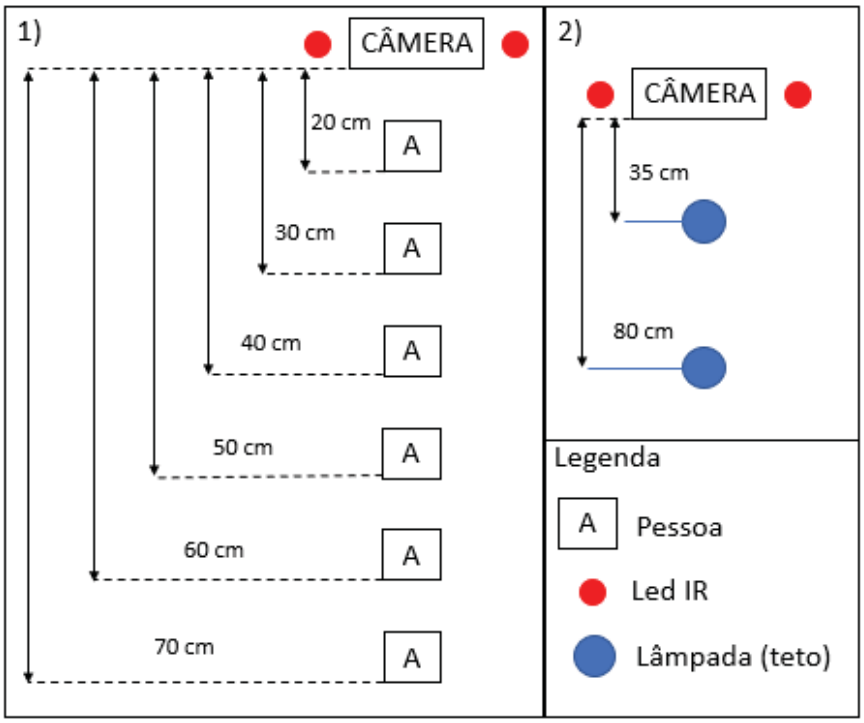
Para validar a eficiência do projeto, realizaram-se testes em diferentes situações: com variação na iluminação, com alterações nas distâncias de posicionamento e em diferentes condições (como a câmera sofrer influência de vibrações). As características e os resultados de cada teste estão descritos nos itens a seguir.

5.1 CONFIGURAÇÕES E LOCAIS UTILIZADOS

Para os testes foram escolhidos dois locais distintos, o primeiro é um ambiente interno onde pode controlar com mais facilidade questões como vibração, distância e iluminação de forma mais eficiente. O segundo local escolhido foi o interior de um veículo, esse ambiente serve para validar os resultados encontrados nos testes realizados no ambiente interno.

Os testes foram executados em diferentes distâncias e condições de iluminação. A Figura 32 ilustra as diferentes configurações de distância e iluminação que foram empregadas nos testes.

Figura 32 – Configurações dos testes: 1) distancias. 2) Iluminação



Fonte Próprio Autor

5.2 OS TESTES

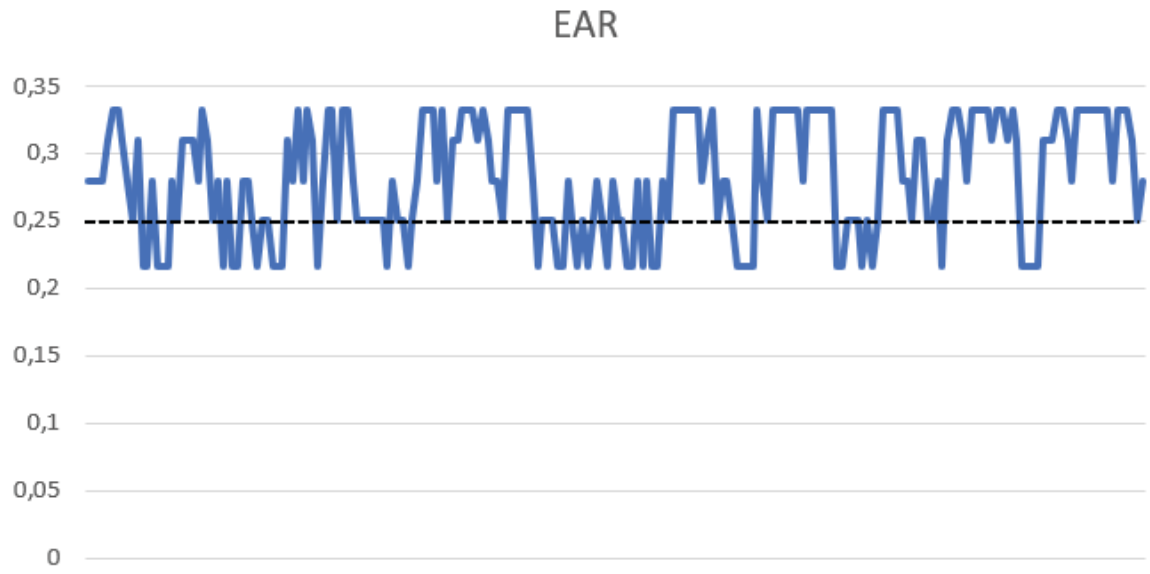
A partir das configurações e características propostas foram realizados os testes, neles foram observados pontos de acerto e de erro que levaram a conclusão do projeto.

O módulo de câmera escolhido, em conjunto com os leds IR, mostrou-se muito eficaz, pois, possibilitou a correta captura das imagens do rosto independente das condições de iluminação e de uma variação na distância entre o objeto e a câmera.

O sistema teve seu correto funcionamento observado em distancias que variam entre 20 a 60cm de distância (entre a câmera e o objeto), pois, nessas distancias foi possível realizar os processos de captura, detecção e extração das características faciais de forma precisa. Para demostrar o funcionamento, foram alguns dos testes realizados. O primeiro desses testes foi realizado a uma distância de 60 cm, com a iluminação dos leds IR. Foram analisados 220 frames

A Figura 33 ilustras o gráfico do EAR, ou seja, o comportamento dos olhos, o valor de 0,25 representa o limiar entre os olhos abertos ou fechado. Caso o valor seja menor que 0,25 considerasse que o indivíduo está com os olhos fechados.

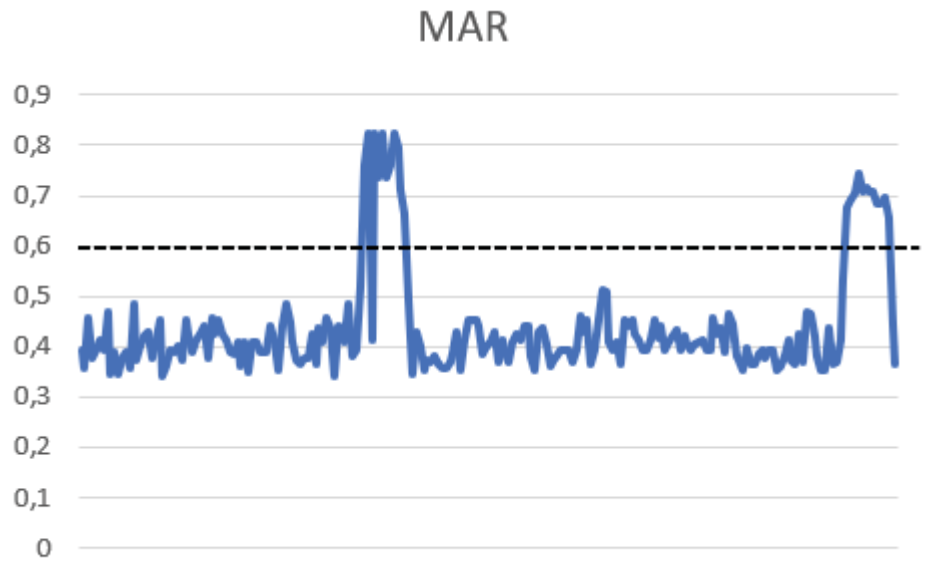
Figura 33 – Gráfico do EAR obtido



Fonte Próprio Autor

Na Figura 34 observa-se o gráfico do MAR, que caracteriza a relação entre abertura e fechamento da boca, foi proposto um valor de 0,6 como limite. Caso o valor obtido seja maior que 0,6 a boca será considerada como aberta.

Figura 34 – Gráfico do MAR obtido



Fonte Próprio Autor

Também foi realizada a análise da quantidade de falsos positivos, falsos negativos, e de verdadeiros positivos que foram encontrados durante esse teste, essas informações estão ilustradas no Quadro 3.

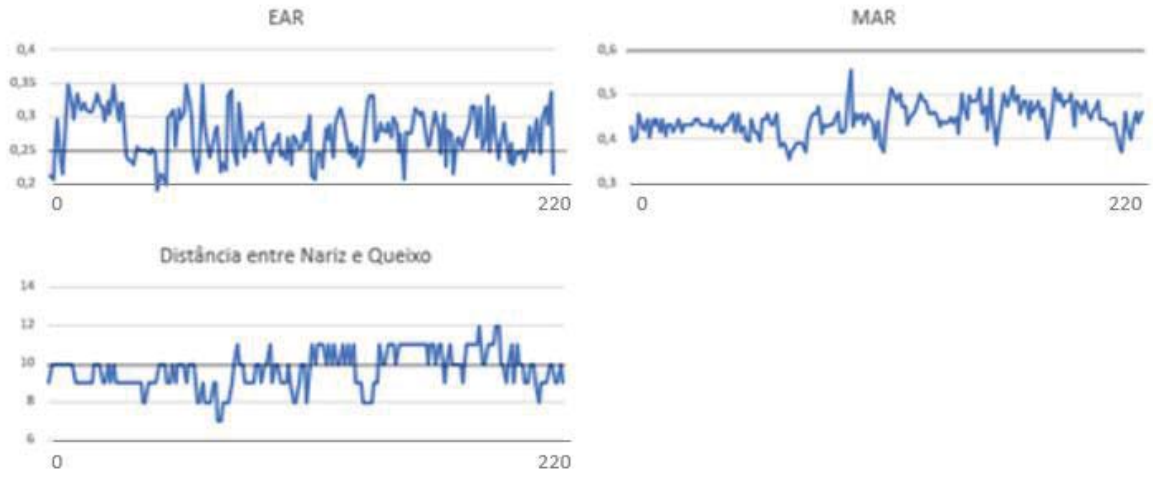
Quadro 3 – Teste de validação com distância 60 cm

Teste 1			
Distância: 60 cm		Iluminação: Led IR	
Falso Positivo	Falso Negativo	Verdadeiro Positivo	Total
4	8	208	220

Fonte Próprio Autor

O segundo teste ilustrado, também foi realizado no ambiente interno, sendo que neste teste a distância utilizada entre a câmera e o usuário foi de 50 cm, onde a iluminação era composta pelas lâmpadas de teto além dos leds IR. Nesta aplicação foi inserido um novo parâmetro a ser analisado, ele possibilita saber se o usuário está olhando para baixo. Para validação desse novo cenário o valor de EAR é 0,25, o valor MAR utilizado foi 0,6 e esse novo parâmetro, que foi denominado de distância, teve seu valor definido em 10. A Figura 35 ilustra esse novo teste realizado, onde 220 frames foram analisados.

Figura 35 – Resultado do teste com distância 50 cm.



Fonte Próprio Autor

Para esse teste também foi realizada a análise da quantidade de falsos positivos, falsos negativos, e de verdadeiros positivos que foram encontrados, esses dados obtidos estão descritos no Quadro 4.

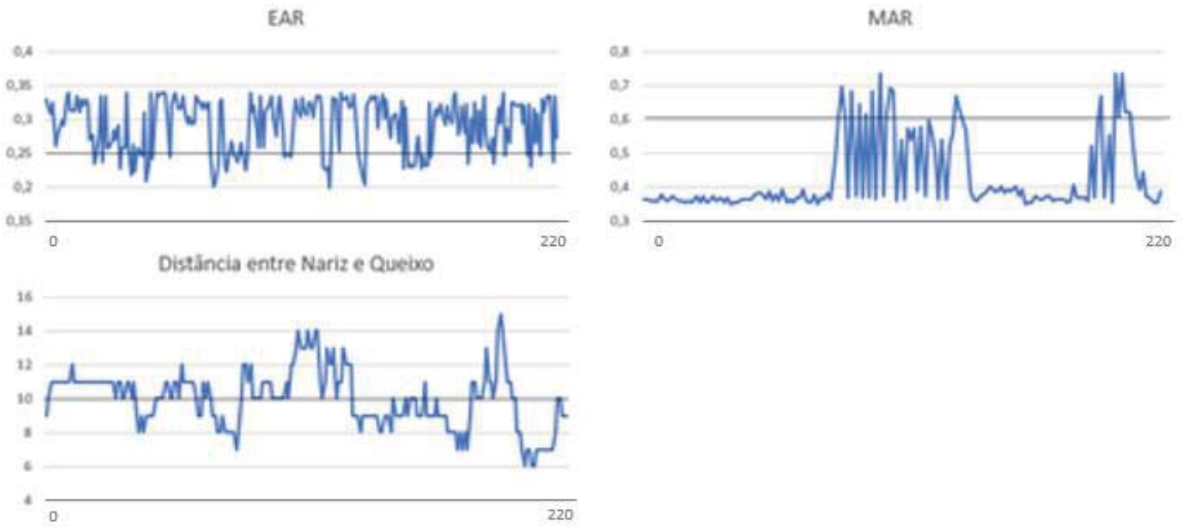
Quadro 4 – Teste de validação com distância 50 cm

Teste 2			
Distância: 50 cm		Iluminação: Led IR + Lâmpadas	
Falso Positivo	Falso Negativo	Verdadeiro Positivo	Total
4	4	212	220

Fonte Próprio Autor

Além dos testes realizados no ambiente interno, o sistema teve suas funcionalidades validadas dentro de um veículo em movimento. Para validar essa condição a câmera foi instalada a uma distância de 50 cm do usuário, todos os valores dos parâmetros permaneceram os mesmos dos testes anteriores. O resultado desse teste está representado na Figura 36, onde novamente foram analisados 220 frames para a construção dos gráficos.

Figura 36 – Resultado do teste realizado com veículo em movimento.



Fonte Próprio Autor

A análise da quantidade de falsos positivos, falsos negativos e verdadeiros positivos pode ser visualizada no Quadro 5.

Quadro 5 – Teste com veículo em movimento

Teste 3			
Distância: 50 cm		Iluminação: Led IR	
Falso Positivo	Falso Negativo	Verdadeiro Positivo	Total
8	12	200	220

Fonte Próprio Autor

5.3 AS ADVERSIDADES

Uma das dificuldades encontradas foi funcionamento em distancias superiores a 70 cm. Nessas condições ocorrem inúmeros problemas com a detecção da face, o sistema não consegue detectar o rosto devido a maior área de captura que acarreta num maior nível de ruído na imagem, causado por influências da iluminação ou mesmo pela presença de outros objetos, que não foram treinados na imagem. Como o processo de detecção funciona bem até uma distância de 60 cm, optou-se por não buscar uma solução para esse problema.

Outra adversidade encontrada no decorrer do desenvolvimento, foi a utilização de óculos, tanto de grau quanto óculos de sol, nessa questão foi observado que existe uma área, na região dos olhos, que possui uma grande quantidade de reflexo, proveniente do módulo de câmera que captura as imagens, e esse reflexo impossibilita que seja possível realizar a detecção da face e dos olhos, essa situação é mostrada na Figura 37.

Para tentar solucionar esse problema, foi tentado utilizar um outro arquivo de detecção, próprio para óculos, treinar um novo arquivo de classificadores Haar, alterar o ângulo da câmera e modificar a iluminação, porém, nenhum desses métodos empregados funcionou. Todavia, por se tratar de uma das questões mais complexas na área de processamento de imagens optou-se por deixar essa questão em aberto visto que as demais condições do software funcionavam bem.

Figura 37 – Detecção usando óculos. A) Imagem com reflexo sem a detecção. B) Imagem com reflexo com a detecção

Fonte Próprio Autor

6 CONCLUSÕES FINAIS

A proposta de um sistema para detecção de sonolência foi atingida. Embora houve inúmeras dificuldades, seguiu-se o planejamento inicial, e, em um aspecto geral o sistema funcionou de maneira satisfatória nas condições a qual o software foi submetido.

Durante o desenvolvimento do projeto, houveram inúmeras dificuldades, que agregaram para o resultado final. O primeiro desafio foi estudar e implementar um sistema de detecção e extração de características faciais. Depois foi necessário aprender uma nova linguagem de programação, que foi o Python, a utilização dessa linguagem trouxe inúmeras vantagem para o trabalho com processamento de imagens, já que possui diferenças bibliotecas que facilitam um pouco o trabalho. Mesmo como o auxílio dessas ferramentas, foi um trabalho árduo realizar a implementação de todas as etapas necessárias para o projeto, já que o aprendizado necessário para desenvolver essas etapas não é algo estudado nas cadeiras do curso de Engenharia Elétrica.

Outro ponto de dificuldade foi a implementação do software no microcomputador Raspberry Pi. Como esse era um sistema novo, com inúmeros recursos e funcionalidades se torna um desafio conseguir aprender todas essa característica em um curto espaço de tempo. Além disso, foi necessário aprender a utilizar o módulo de câmera compatível com o hardware, a escolha da Raspberry Pi câmera foi fundamental, devido a sua compatibilidade nativa com o microcomputador o que diminui o tempo necessário para sua completa utilização.

A maior dificuldade encontrada durante o desenvolvimento, foi quando o usuário estava utilizando óculos, essa característica faz com que o sistema não funcione da maneira correta e como consequência não traz resultados satisfatórios. Foram realizadas diversas tentativas para solucionar esse problema, porém nenhuma trouxe o resultado esperado, então por se tratar de um aspecto muito complexo e também pelo pouco tempo disponível optou-se por deixar essa questão em aberto para possíveis outros trabalhos na área.

Como sugestão para possível continuidade desse trabalho está a implementação dele em um hardware dedicado para o processamento de imagens e inteligência artificial como por exemplo o Nvidia Jetson Nano. Outra sugestão, seria a aplicação de redes neurais artificiais para o auxílio nos processos de detecção e extração de características.

Apesar dos problemas citados, o software se comporta de forma satisfatória dentro das condições especificadas nos testes, sendo ainda possível aplicar o código para outras tarefas como, por exemplo, detecção de pessoas em um sistema de segurança ou mesmo para detectar expressões faciais.

REFERÊNCIAS

AZEVEDO, Eduardo ; LETA, Fabiana R.; CONCI, Aura . **Computação Gráfica** : Teoria e Pratica. Rio de Janeirao: Elsevier, 2008.

BOSCH. **Driver drowsiness detection**.: virtual book, 2018.
Disponível em:
<https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/driver-drowsiness-detection/ >.
Acesso em 06 de maio de 2018

CECH, Jan; Soukupova, TEREZA. Real-Time Eye Blink Detection using Facial Landmarks, 21st Computer Vision Winter Workshop, 2016.

DANTAS, Tiago. **Fases do Sono**; virtual book, 2018.
Disponível em <https://brasilescola.uol.com.br/curiosidades/fases-sono.htm>.
Acesso em 07 de maio de 2018.

FABIEN, Mael. A Guide to Face Detection: virtual book, 2019.
Disponível em: < https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1>. Acesso em 15 de abril de 2019

FERREIRA, Julian Manayra da Silva; MARQUES, Ricardo Costa da Silva; JUNIOR, Geraldo Braz. Deteccção de Sonolência ao Volante Utilizando Facial Landmarks e Eye Aspect Ratio, 1Universidade Federal do Maranhão (UFMA).

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Education, Inc., 2011.

KAISER, Adrien. **What is Computer Vision.:** virtual book, 2017.
Disponível em: <https://hayo.io/computer-vision/>. Acesso em 06 de maio de 2018.

MONK, Simon. **Programando o Raspberry Pi:** Primeiros Passos com Python: Virtual Books, 2013. Disponível em:<https://s3.novatec.com.br/capitulos/capitulo-9788575223574.pdf>. Acesso em: 08 de maio de 2018.

NUMPY. Disponível em: <http://www.numpy.org/>. Acesso em: 02 abril 2019.

OPENCV. **About**: virtual book, 2018
Disponível em: <https://opencv.org/about.html>. Acesso em 22 de maio de 2018

RASPBERRY PI FUNDATION, **Raspberry Pi Camera Module v2:** virtual book, 2016.
Disponível em: <https://www.raspberrypi.org/products/camera-module-v2/>.
Acesso em 07 de maio de 2018

RODRIGUES, Matheus Bezerra Estrela. Estudo da aplicação do algoritmo Viola-Jones à detecção de pneus com vistas ao reconhecimento de automóveis. 2012. 69 p. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina, Campina Grande, PB, 2012.

SAHAYADHAS, Arun, KENNETH Sundaraj. **Detecting Driver Drowsiness Based on Sensors: A Review**: Virtual Book, 2012,
Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/>.
Acesso em 05 de maio de 2018

SOLOMON, Chris; BRECKON, Toby. Fundamentos de Processamento Digital de Imagens - Uma Abordagem Prática com Exemplos em Matlab. LTC, 07/2013.

TERRA. **É mesmo necessário dormir 8 horas por noite**: virtual book, 2016.
Disponível em:
<https://www.terra.com.br/noticias/e-mesmo-necessario-dormir-oito-horas-por-noite,5e3c80e3b395088505dc886c41f8dcd5hxfopmh1.html>.
Acesso em 07 de maio de 2018.

THOMSEN, Adilson. **Raspberry Pi Model B**+: virtual book, 2018.
Disponível em: <https://www.filipeflop.com/blog/lancamento-raspberry-pi-3-model-b-plus/>.
Acesso em 08 de maio de 2018

VENTURA, felipe. **Estas Animações mostram como funciona os sensores da sua câmera**: virtual book, 2015.
Disponível em: <https://gizmodo.uol.com.br/video-sensor-cmos-ccd/ >.
Acesso em 07 de maio de 2018.

APÊNDICE A – SCRIPT DA BIBLIOTECA IMUTILS ALTERADA

```
from picamera.array import PiRGBArray
from picamera import PiCamera
from threading import Thread
import cv2

class PiVideoStream:
        def __init__(self, resolution=(320, 240), framerate=30):
                # Inicializa câmera e o stream
                self.camera = PiCamera()
                self.camera.resolution = resolution
                self.camera.framerate = framerate
    self.camera.videostabilization = True
                self.rawCapture = PiRGBArray(self.camera, size=resolution)
                self.stream = self.camera.capture_continuous(self.rawCapture,
                        format="bgr", use_video_port=True)

                # Inicializa o frame e a flag, se o threa for interrompido
                self.frame = None
                self.stopped = False

        def start(self):
                # Inicializa o Thread para ler os frames do video
                t = Thread(target=self.update, args=())
                t.daemon = True
                t.start()
                return self

        def update(self):
    #Loop aguardando uma interrupção no segmento
                for f in self.stream:
                        # Pega o frame atual, e espera por um novo quadro
                        self.frame = f.array
```

```
                self.rawCapture.truncate(0)

                # Se a variavel indicar que parou a captura, realiza:
                if self.stopped:
                        self.stream.close()
                        self.rawCapture.close()
                        self.camera.close()
                        return

def read(self):
        # Retorna o quadro mais recente
        return self.frame

def stop(self):
        # Indica se thread parou
        self.stopped = True
```