UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Título do trabalho: Detecção de fadiga na operação ferroviária

Felipe Mendes

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Detecção de fadiga na operação ferroviária

Felipe Mendes

USP - São Carlos

2022

Felipe Mendes

Título do trabalho: Detecção de fadiga na operação ferroviária

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Diego Raphael Amancio

USP - São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

M538d

Mendes, Felipe
Detecção de fadiga na operação ferroviária / Felipe Mendes; orientador Diego Raphael Amancio. -- São Carlos, 2022.
59 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Fadiga. 2. Bibliotecas. 3. OpenCv. 4. Dlib. 5. Vídeos. I. Raphael Amancio, Diego, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

DEDICATÓRIA

A minha esposa por todo auxílio,
com muita paciência, compreensão
e apoio durante todo o MBA.

AGRADECIMENTOS

Primeiramente a Deus, que permitiu continuar adquirindo conhecimento ao longo de minha vida.

A minha esposa Júlia Montenegro de Oliveira por toda dedicação, empenho, compreensão e auxílio durante a execução de todo MBA.

Aos meus familiares, em especial a meu irmão Bruno Mendes que recomendou o curso e acompanhou algumas etapas mesmo que distante.

Ao meu orientador professor Dr. Diego Raphael Amancio por toda dedicação, suporte e auxílio.

A coordenação do curso que não abdicou esforço para garantir o melhor atendimento e auxílio possível, em especial para a professora Solange O. Rezende que foi um exemplo como pessoa e profissional.

A todos os professores e tutores que se dedicaram para passar o conhecimento adquirido durante sua carreira profissional.

A empresa Rumo Logística por todo apoio para aperfeiçoamento do conhecimento e proporcionar a possibilidade de estar atuando na ferrovia.

EPÍGRAFE

No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio-termo. Ou você faz uma coisa bem-feita ou não faz.

(Senna, 1990)

RESUMO

MENDES, F. **Detecção de fadiga na operação ferroviária** 2022. 59 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

Com o avanço da tecnologia, os operadores ferroviários ficam ociosos e consequentemente mais suscetíveis a fadiga. Para evitar que ocorram desvios operacionais, é necessário avaliar algoritmos que possuem como objetivo detecção de fadiga do maquinista durante sua rotina de trabalho, portanto, o algoritmo precisa atuar na análise de imagens de vídeos em tempo real. Não foi encontrado nenhuma aplicação de detecção de fadiga com imagem na operação ferroviária em busca de referências bibliográficas, porém algumas soluções similares foram analisadas e diversos algoritmos foram pesquisados para verificar qual a melhor solução já estudada até o momento para aplicação. As bibliotecas para tratamento de imagens estão avançadas e a pesquisa por aplicações simulares auxiliaram na decisão das bibliotecas escolhidas, visto que pontos positivos, negativos e resultados são sinalizados, contudo, as bibliotecas OpenCV e Dlib foram bem avaliadas e utilizadas nessa aplicação. Foi demonstrado o algoritmo de forma detalhada, as funções necessárias para detecção da face e pontos de interesse, relatado boas práticas com as dificuldades encontradas, para facilitar caso tenham interesse de replicar a funcionalidade. Para verificar a performance e robustez do algoritmo, foi realizado teste em situações adversas, com o objetivo de dificultar os assertos e mesmo com todas as dificuldades e variáveis inseridas, o algoritmo obteve em média 95,63% de assertividade com pessoas de diferentes sexos, sendo considerado que o resultado foi satisfatório para o problema proposto e possui escalabilidade para outras pessoas. Foi demonstrado os pontos negativos e dificuldades do algoritmo, assim como melhoria contínua na busca por novas bibliotecas que vão sendo desenvolvidas, assim como possibilidades para sequência desse estudo em trabalhos futuros.

Palavras-chave: fadiga; bibliotecas; OpenCV; Dlib; vídeos; imagens.

ABSTRACT

MENDES, F. **Fatigue detection in railway operation**. 2022. 59 f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

With the advancement of technology, railway operators are idle and consequently more susceptible to fatigue. To prevent operational deviations from occurring, it is necessary to evaluate algorithms that aim to detect the fatigue of the machinist during his work routine; therefore, the algorithm needs to act in real-time video image analysis. In my search of bibliographic references, there were no applications capable of detecting fatigue with images available in the railway operation; however, some similar solutions were analyzed, and several algorithms were researched to verify the best key already studied for application. The libraries for image processing are advanced, and the search for simulation applications helped decide the chosen libraries since positive and negative points, and results are signaled. However, the OpenCV and Dlib libraries were well evaluated and used in this application. The algorithm was demonstrated in detail, the functions necessary for detecting the face and points of interest, reporting good practices with the difficulties encountered, to facilitate if they are interested in replicating the functionality. To verify the performance and robustness of the algorithm, a test was carried out in adverse situations to make assertions difficult. Even with all the difficulties and variables inserted, the algorithm obtained an average of 95.63% assertiveness with people of different sexuality, considering that the result was satisfactory for the proposed problem and scalability for others. The negative points and difficulties of the algorithm were demonstrated, as well as continuous improvement in the search for new libraries that are in development and possibilities for following this study in future works.

Keywords: fatigue; libraries; OpenCV; Dlib; videos; images.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	–	Atuações Indevidas
CNN	–	*Convolutional Neural Network*
DWT	–	Transformada Wavelet Discreta
EAR	–	*Eye Aspect Ratio*
FA	–	Falhas de atuações
FDDB	–	Face Detection Data Set and Benchmark
GF	–	Filtro Gabor
HOG	–	*Histograms of Oriented Gradients*
LBP	–	*Local Binary Patterns*
LBPH	–	*Local Binary Patterns Histograms*
MB-LBP	–	*Multi-scale Block Local Binary*
MTCNN	–	*Multi-task Cascaded Convolutional Networks*
O-Net	–	Rede de Saída
PCA	–	Análise de Componentes Principais
PERCLOS	–	*Percentage Eye Closure*
P-Net	–	Redes de Propostas
QTD	–	Quantidade esperado de alertas
R-Net	–	Rede de Refinamento
SIFT	–	*Scale Invariant Feature Transform*
SVM	–	*Support Vector Machine*
TX	–	Taxa de Assertividade

SUMÁRIO

1 INTRODUÇÃO

Os seres humanos estão propícios a condição de fadiga durante diversos momentos da vida, essa sensação de estar cansado é um sintoma médico de desgaste que pode ser físico ou mental. É muito comum as pessoas sentirem fadiga após realizarem atividades físicas, situação de estresse prolongado, insônia e por terem diversos tipos de doenças (Dor, 2022).

A ferroviária depende de pessoas para execução das tarefas, onde os maquinistas são os condutores e responsáveis pela operação. A operação ferroviária é um sistema de transporte pelos quais circulam trens de carga e passageiros sobre os trilhos (Michaelis, 2022). Os trens são constituídos de locomotivas e vagões, onde a locomotiva é a máquina controlada pelo maquinista, sendo responsável por rebocar os vagões sobre os trilhos. Logo, os vagões são responsáveis pelo transporte da carga.

Nas locomotivas mais novas, os maquinistas podem utilizar a condução semiautônoma para obter uma boa performance de economia de combustível, conforme (Rumo, 2022), nesse caso sem interferência humana após atingir a velocidade de 12 km/h. Essa atuação gera grandes benefícios para a ferrovia, em contrapartida o maquinista fica mais ocioso e tende a ocorrer problema com fadiga.

Para identificar desvios com o maquinista, é necessário realizar análise de muitos vídeos, sendo inviável para humanos, visto que a quantidade de informação a ser analisada é grande devido a quantidade de locomotivas. Com isso, atualmente é impossível realizar verificação de forma rápida para identificação de desvio operacional.

Contudo, se utilizar um algoritmo para verificar o comportamento dos vídeos das câmeras de vigilância, é possível utilizar essa informação de melhor forma, sendo possível monitorar a ação humana que está propícia a erros por diversas variáveis, como a fadiga.

Para conseguir utilizar as imagens de forma eficiente e proativa, este trabalho tem como objetivo realizar o tratamento das imagens do vídeo para conseguir identificar condições de fadiga, sendo possível atuar de forma mais rápida ou até mesmo em tempo real no problema identificado.

Para detecção da anormalidade, possui a necessidade de atuação em tempo real. Alguns algoritmos foram pesquisados, como as bibliotecas OpenCV (Heesch, 2022) e Dlib (King, 2013), que se demonstram como fundamentais para desenvolvimento da solução.

O desafio é encontrar um algoritmo que atue de forma assertiva para a condição proposta. Caso o operador durma momentaneamente no comando da locomotiva, condições

não previstas de forma operacional podem acontecer, gerando parada do trem. Neste caso possui o custo do combustível para a retomada de velocidade do trem ou no pior caso, onde o trem pode passar da velocidade ou percurso permitido, ocasionar a parada por segurança, que dependendo da velocidade e local, possui o risco de descarrilamento por ser considerado uma frenagem em condição de emergência.

2 REVISÃO BIBLIOGRÁFICA

Com a evolução tecnológica e automação das máquinas em busca de melhor eficiência energética, é natural que as pessoas fiquem mais ociosas quando se trata de um sistema que necessita ser monitorado devido condições de anomalias, como falha no sistema e condições de emergência que estão relacionadas a vida humana.

Na busca da melhor metodologia para automação do sistema de vigilância, nada similar foi encontrado relacionado a mesma aplicação na ferrovia, apenas modelos de imagens voltadas para identificação de componentes com anomalias voltados para segurança ferroviária, como no caso de identificação de defeitos em rodas, conforme (Trilla et al., 2021).

Em diferentes aplicações e com objetivo em comum, foi encontrado alguns sistemas em automóveis de alto padrão e caminhões com o objetivo de evitar acidentes por fadiga do motorista.

Muitas das aplicações, mostram informações apenas superficiais, sem detalhar a metodologia do desenvolvimento, isso devido fazer parte de um produto e o conhecimento aplicado ser propriedade de uma empresa.

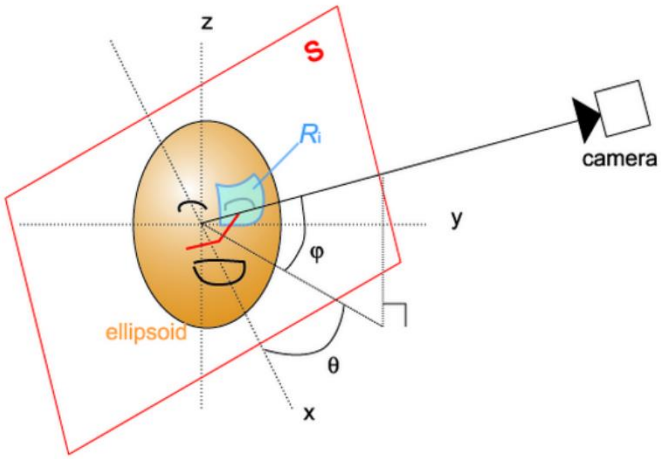
Possui diversos métodos para detecção de fadiga, conforme (Queiroz, 2011). O primeiro é por atividade cerebral, onde é monitorado a atividade cerebral, muscular e cardiovascular. O segundo é baseado no desempenho do motorista, onde no caso da ferrovia, já possui um sistema de vigilância que monitora e realiza a parada do trem caso o maquinista não tenha nenhuma interação com a locomotiva. O terceiro método é baseado na detecção por análise visual, utilizando técnicas de processamento de imagens.

O último método é o que se aproxima da proposta deste trabalho, funciona através de coleta de imagem obtidas por câmeras, onde utiliza a aparência do rosto, posicionamento da cabeça, atividade dos olhos e boca para refletir a ocorrência de sonolência.

Para realizar a detecção, na maioria dos estudos analisados, primeiramente é detectado a face do indivíduo e posteriormente aplicado um filtro para detecção de condições da face, como olhos e boca se movimentando de determinada maneira e tempo.

Uma das metodologias aplicadas cria uma elipse sobre a imagem com o objetivo de detectar a cabeça humana de diversas posições e com o objetivo de obter robustez e estabilidade na detecção, introduzindo recursos de imagem contra iluminação, foi utilizado mais duas elipses para ser possível estimar a orientação da face, uma dentro e outra fora da primeira elipse gerada, para obter de forma mais precisa a detecção da cabeça (Bunke et al., 2009), conforme Figura 1.

Figura 1 - Elipse para detecção de cabeça



Fonte: (Bunke et al., 2009).

A segunda etapa é detectar os componentes internos da face, como boca, olhos ou bochechas. Nessa fase, também são reconhecidos os componentes geométricos da face em um determinado ângulo de depressão. Para ser possível realizar esse processo, foi utilizado filtro *Gabor-Wavelets*.

A biblioteca OpenCV possui a capacidade de utilização do filtro *Gabor-Wavelets* (Filtering, 2014), (Heesch, 2022).

O reconhecimento facial também pode ser realizado com visão estéreo conforme (García-Rios et al., 2014), na ocasião se preocupou em garantir que as imagens sejam tratadas de forma tridimensionais e não bidimensionais, isso ocorre pois havia o receio de que uma imagem de alta resolução tivesse capacidade de enganar o sistema de reconhecimento facial. Neste caso são analisados três métodos diferentes para a extração de características: Análise de Componentes Principais (PCA), Filtro Gabor (GF) e Transformada Wavelet Discreta (DWT). O PCA foi utilizado para reconhecimento de padrões com redução de dimensionalidade, o que permite representar a imagem de forma compacta. O GF foi utilizado para caracterização da imagem de duas dimensões, sendo representado propriedades de orientação e frequência, determinada por quatro parâmetros, localização espacial x e y, frequência de sintonia e orientação espacial, dessa forma preserva o vetor característica da mesma dimensão. O DWT é utilizado para caracterizar as imagens com análise de espaço e frequência, realizando a decomposição da imagem por operação de filtragem, com compactação em níveis. Neste caso, todas as reduções de dimensionalidades foram aplicadas a classificador *Support Vector Machine* (SVM) com o objetivo de verificar em uma base de dados a identificação do indivíduo, processo um pouco diferente do proposto neste trabalho, porém com uma determinada

similaridade na verificação da face. O GF se comportou ligeiramente melhor para identificação, como resultado geral de classificação, foi obtido resultado de 99,5%, porém as imagens que foram tratadas estão padronizadas e sem variação de inclinação, conforme Figura 2, seria importante realizar teste e verificação de comportamento para outras condições.

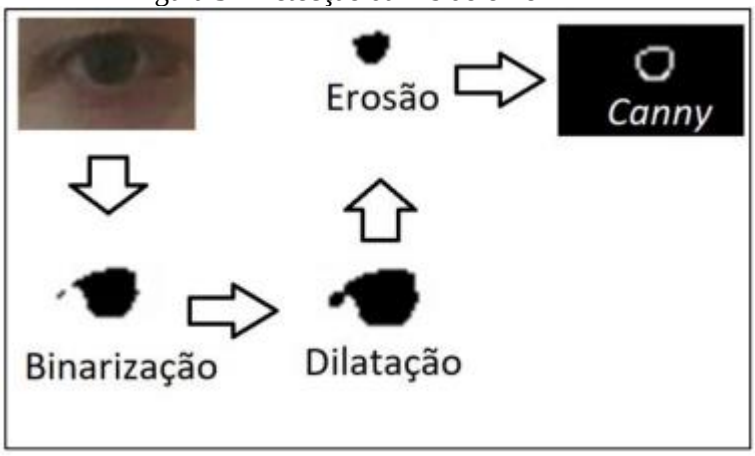
Figura 2 - Fotos da base de dados tratada

Fonte: (García-Rios et al., 2014).

O algoritmo de Viola-Jones para detecção de faces também foi utilizado em outro experimento, conforme (de Almeida Noronha et al., 2019), neste caso possui destaque na agilidade do tempo de detecção, o espaço em que a face se encontra é classificada em pontos interligados e selecionada para o restante da imagem não interferir no resultado. O algoritmo AdaBoost usando *Haar-like* foi adotado para sequência de análise dos olhos, neste caso também são retornados pontos interligados na região dos olhos que se deseja detectar. Como segunda opção, foi adotado o algoritmo de *landmarks*, neste caso foi selecionado 6 pontos da região dos olhos. O sistema de *landmarks* possui capacidade de detecção de fatores e expressões em tempo real. Para detecção de boca, também foi utilizado *landmarks*, informando que estudos de 2013 demonstraram dificuldade da biblioteca OpenCV com o movimento da cabeça, detectando outras partes do rosto.

Com os olhos detectados, é necessário implementar o método de transformada de Hough em círculo para detecção da íris do olho, sendo posteriormente aplicado o detector de borda Canny. Com cada posição dos pontos da borda da íris é possível definir o centro do círculo pela posição da matriz com mais incremento, caso não for encontrado, a íris não é detectada, conforme Figura 3. A desvantagem desse método é a interferência da luz do ambiente e o custo de processamento.

Figura 3 - Detecção da íris do olho

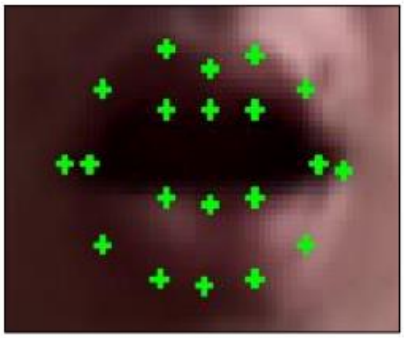


Fonte: (de Almeida Noronha et al., 2019).

Uma segunda técnica para detecção da íris foi o *Template Matching*, onde realiza a comparação de uma imagem salva com a detectada, a partir de determinado score o olho é considerado aberto ou fechado.

A terceira técnica de detecção foi o *landmarks* com aplicação de *threshold*, onde se a distância entre os pontos estiver com um determinado valor, é definido que o olho está fechado ou aberto. A mesma metodologia foi aplicada ao bocejo, onde se a distância entre os pontos estiver acima do *threshold* durante 1,5 segundo, caracteriza a ocorrência, conforme Figura 4.

Figura 4 - Pontos de landmarks da boca



Fonte: (de Almeida Noronha et al., 2019).

Com essas informações, é possível calcular a *percentage eye closure* (PERCLOS), que é baseado na quantidade de olhos abertos e a quantidade detectados. Caso o resultado for acima de 80%, 70% com bocejo constante em um minuto ou 16 piscadas lentas de 3 segundos por minuto, o condutor é classificado como fadigado e o sinal de alerta deve ser emitido.

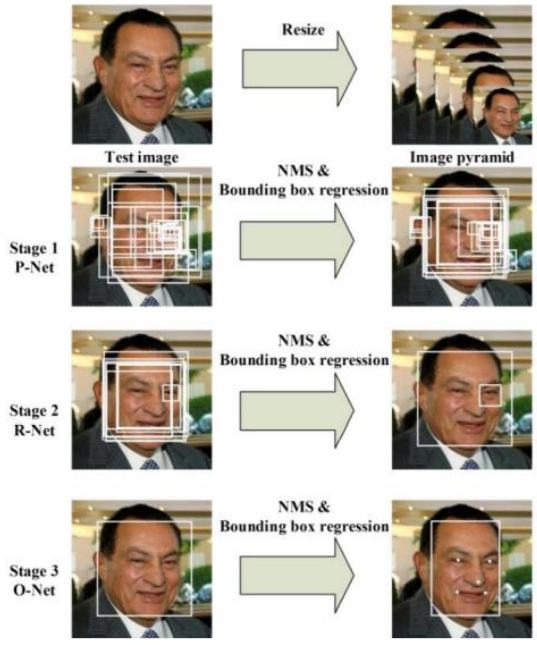
Dentre todos os testes, o landmarks foi o que obteve melhor resultado, 90% de precisão, a transformada de Hough obteve 29,5% e *Template Matching* 52,9%. Importante ressaltar que foi gerado dificuldade com os dois últimos modelos por serem mais custosos em questão de

processamento, isso por estar utilizando um hardware de baixo custo.

Com o passar do tempo, algoritmos baseados em redes neurais profundas começaram a surgir. Com isto, as dificuldades com iluminação, posicionamento do rosto passaram a ser superadas e com alto nível de precisão em aplicação no mundo real, conforme (Da & Paula, 2017). O detector de imagem facial Viola-Jones foi uma das principais evoluções neste quesito de aplicação do mundo real, devido resultado de precisão e velocidade, esse algoritmo é muito utilizado em câmeras digitais. Quatro novas melhorias foram tratadas como principais: *Scale Invariant Feature Transform* (SIFT), *Histograms of Oriented Gradients* (HOG), SVM e *Deep Learning*. Muitos conjuntos de dados com foco em reconhecimento facial como o LFW surgiram e biblioteca como OpenCV atingiu a disponibilidade de código de alta qualidade. Implementações em *deep learning* como HyperFace, DeepID3, DeepFace e FaceNet alcançaram ótimos resultados em LFW.

As *Convolutional Neural Network* (CNN) foram utilizadas para localização, podendo detectar objetos, faces, por exemplo. O *Multi-task Cascaded Convolutional Networks* (MTCNN) é uma metodologia para detecção de faces, também realiza tratativas de alinhamento. Para implementação é utilizado CNN em cascata e método de aprendizado composto por três etapas. Primeiramente é gerado janelas com CNN conhecida como Redes de Propostas (P-Net), posteriormente refina as janelas candidatas com a Rede de Refinamento (R-Net) e por último utiliza CNN mais complexa para gerar a caixa delimitadora com os pontos da face com a Rede de Saída (O-Net), conforme Figura 5.

Figura 5 - Demonstração da arquitetura da MTCNN



Fonte: (Da & Paula, 2017).

O MTCNN utiliza três funções de perda diferentes para realização do método de aprendizado multitarefa: classificação de face (perda de entropia cruzada), regressão de caixa delimitadora (perda euclidiana) e localização de marcos faciais com cinco pontos de referência (perda euclidiana).

Com o objetivo de obter metodologia de comparação dos classificadores, foram criados conjunto de dados como o Wider Face, composto por 32.203 imagens e 393.703 faces. Essas imagens estão divididas em 61 classes. Outros conjuntos de dados são Face Detection Data Set and Benchmark (FDDB) e IARPA Janus (IJB-A). O FDDB contém 2.845 imagens com 5.171 faces e o IJB-A é um conjunto de dados com 5.712 imagens e 2.085 vídeos de 500 assuntos diferentes.

No reconhecimento de face demonstrados por (Costa et al., 2021) foram utilizados os algoritmos Eigenface, Fisherface e *Local Binary Patterns Histograms* (LBPH) em conjunto com Haar Cascade, sendo todos da biblioteca OpenCV. Foi explorada CNN em conjunto com o algoritmo HOG, da biblioteca Dlib. Foram utilizados datasets de imagens de faces públicos, Caltech com 445 imagens, Georgia Tech com 750 imagens e Yale que possui 165 imagens.

Com o objetivo de obter os descritores das faces, para identificar os pontos dos rostos como sobrancelha, olhos, nariz e boca, foi necessário utilizado o arquivo dlib_face_recognition_resnet_model_v1.dat disponível na biblioteca do Dlib, constituído de uma rede ResNet com 29 camadas de convolução. Isso possibilitou encontrar os melhores resultados com a CNN, conseguindo acertar 100% dos dois primeiros datasets respectivamente e 93% do dataset Yale.

Após compilar os algoritmos, como foi utilizado diferentes modelos treinados em três datasets, é possível verificar que a CNN funcionou melhor na maioria dos casos, porém dependendo da base de dados, o comportamento pode ser diferente. Podem ser considerados outros fatores como grau de confiabilidade e tempo de execução, no último quesito, a CNN superou apenas o Eigenfaces, para os demais casos o tempo foi superior, chegando a 8,6 vezes maior.

Com o objetivo de desenvolver uma metodologia para detectar sonolência ao dirigir automóveis e implementar em aplicativo android (Jabbar et al., 2020), relata que uma Rede Bayesiana pode ser melhor do que PERCLOS. É proposto a utilização de MobileNet-SSD que gera uma precisão de 84% em média, o sistema pode trabalhar em tempo real pois é eficiente e econômico.

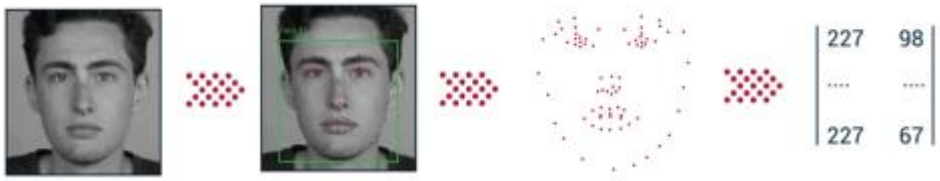
Também foi utilizado banco de dados YawDD e NTHU-DDD com vídeo clipes de bocejos, sendo mais robustos a mudança de posição e ângulo do rosto na câmera.

Algumas empresas estão tentando combinar metodologias para aprimorar o resultado, como aplicação de detecção de sonolência, com detecção de piscada, rastreamento ocular e monitoramento de sinais vitais. Para essa implementação são necessários vários sensores, como infravermelho, câmeras e monitores de frequência cardíaca em uma única plataforma.

Mesmo com todas as implementações, por se tratar de um dispositivo embarcado, é importante avaliar o custo de processamento, pois além de melhorar o algoritmo é importante manter o aplicativo com bom desempenho.

Como se trata de vídeos, é necessário extrair os frames, o que resulta em 600.000 imagens aproximadamente. Por ser treinamento de aprendizado profundo, é utilizado o Codebox como segunda etapa para aumentar a quantidade de dados de treinamento. A biblioteca Dlib foi utilizada para encontrar os pontos de referências faciais com 68 pontos nas coordenadas x,y, conforme Figura 6. Também foi utilizado o Haar Cascades da biblioteca OpenCV para aprimorar a atuação da biblioteca Dlib.

Figura 6 - Detecção de recursos faciais



Fonte: (Jabbar et al., 2020)

Um modelo baseado em CNN com cinco camadas utilizando a biblioteca Dlib Facial, gerou novas imagem com o Codebox da base de dados NTHU.

Experimentos foram realizados usando o modelo D2CNN-FLD, D2MLP-FLD, RCNN com VGG-16 e modelos AlexNet como benchmarks para comparação, sendo comparados em precisão, compressão, tempo de detecção e velocidade, conforme Tabela 1.

Tabela 1- Resultado experimental

Dataset	Category	D2MLP-FLD MLP-Model	D2CNN-FLD CNN Model	Faster RCNN (VGG-16)	Faster RCNN (AlexNet)
Accuracy (%)	Validation	80.9	83.3	90.5	82.8
Compression (MB)	Model size	0.1	0.075	547	236
	GPU Memory	44.1	38.9	3183	845
Drowsiness Detection time (ms)	Dell workstation NVIDIA Quadro P4000	41.6	38.45	-	-
	Samsung Galaxy S8 plus	154	142	-	-
Overall speed (fps)	Dell workstation NVIDIA Quadro P4000	7.4	6.87	9.1 (GTX-1080)	22 (GTX-1080)
	Samsung Galaxy S8 plus	267.56	234.25	-	-

Fonte: (Jabbar et al., 2020)

O modelo D2CNN-FLD se comportou de forma mais satisfatória em relação aos demais, atingiu precisão de 83,33%, onde o tamanho máximo do modelo ficou próximo de 75 KB, o que possibilita a implementação em qualquer sistema embarcado com alta performance e velocidade.

A biblioteca OpenCV é utilizado para visão computacional, possui código aberto e biblioteca de aprendizado de máquina contendo várias funções. O algoritmo está escrito em C e C++, o que possibilita processamento em tempo real devido sua velocidade. Também possui adaptações para ser utilizado linguagem de programação Python e possui diversas aplicações, inclusive entre essas funcionalidades estão dois métodos de detecção de faces: *Haar Feature-based Cascade Classifiers* e *Local Binary Patterns* (LBP). Conforme (Da & Paula, 2017)(Heesch, 2022d).

A OpenCV pode ser treinada ou utilizar classificadores já treinados.

Os *Haar Feature-based Cascade Classifiers* é um método baseado em aprendizado de máquina que permite reconhecimento de objetos (carros, aviões, pessoas e entre outros). No treinamento o algoritmo extrai características do tipo Haar, que são baseadas em *wavelets* de Haar, um conjunto de funções que codificam diferenças nas intensidades médias entre as regiões, recursos muito parecidos com kernels convolucionais, que ficam responsáveis por detectar os recursos específicos presentes na imagem. É selecionado as melhores caraterísticas utilizadas no classificador, para isso é utilizado o AdaBoost, onde os classificadores estão em cascata com características especificas do tipo Haar.

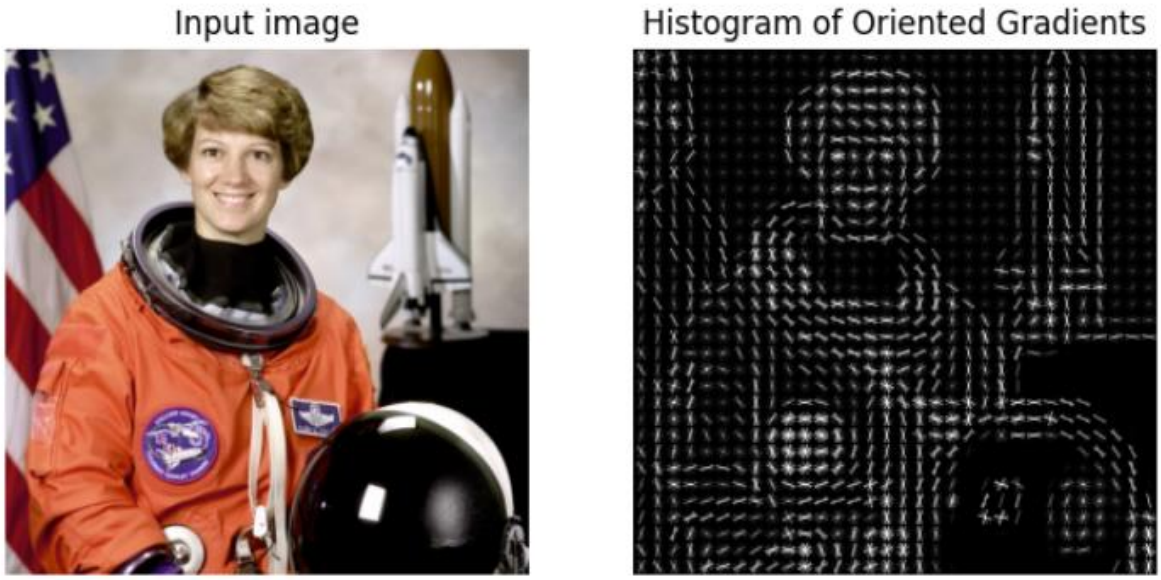
O LBP é um extrator de recursos que é utilizado para diversas aplicações, como detecção de face, reconhecimento de face, análise de expressão facial e entre outros. Funciona em 3×3, utilizando o valor do pixel central, portanto, basicamente as características são extraídas por comparação de um pixel com seus pixels vizinhos e limitando o valor de cada pixel, se o valor comparado é maior ou igual ao pixel selecionado, ele é limitado a 1, caso contrário é limitado a 0. Como possui 8 pixels vizinhos possíveis, é gerado 256 unidades de textura. É tratado com um algoritmo que não é sensível a mudança de intensidade luminosa, porém a mudança for brusca, terá influência no resultado.

Como o LBP possui dificuldade com reconhecimento facial mesmo com a evolução, foi necessário realizar uma associação criando o *Multi-scale Block Local Binary* (MB-LBP) que é utilizado no OpenCV, onde não possui o nível de pixel, sendo adotado o valor médio de cada região de 9x9. O AdaBoost também é aplicado para obter as melhores características do histograma da respectiva região e criar um classificador de face, portanto é realizado da mesma forma que Haar, utilizando o classificador de rosto em cascata, descartando as janelas que não

possuem face antes de ir para a próxima etapa, otimizando o algoritmo e evitando cálculos desnecessários.

A biblioteca Dlib é escrita em código aberto C++ que contém algoritmos e métodos de aprendizado de máquina. Entre as funções oferecidas, é necessário ter uma atenção especial com o algoritmo de detecção de imagem frontal, onde o principal algoritmo é baseado em HOG, sendo um descritor de recursos densos, extraindo os recursos para todos os locais sobre uma imagem ou região de interesse, sem computar nos pixels da vizinhança. O HOG é baseado na distribuição de gradientes de intensidade e direções de borda podem descrever a imagem, portanto pode identificar pessoas e distinguir objetos, conforme Figura 7. Conforme (Da & Paula, 2017),(Dlib C++ Library, n.d.).

Figura 7 - Histograma de Gradientes Orientados



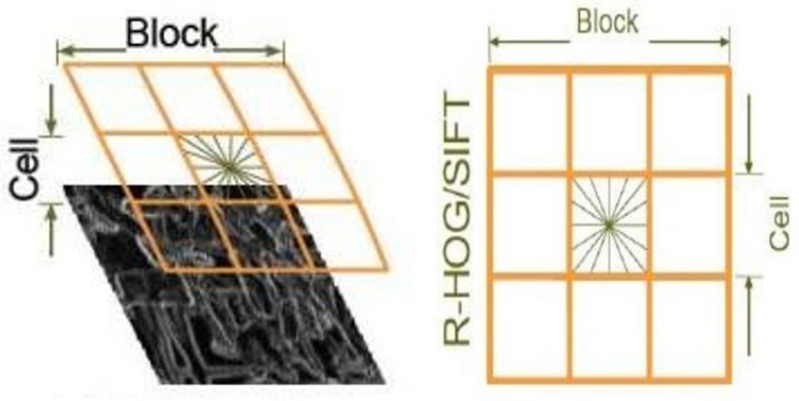
Fonte: (Image, 2011)

A sequência de utilização do HOG somente se fortaleceu quando começou a ser utilizado com o SIFT, que é invariável quanto a translação, escala e rotação. Transforma a imagem em um conjunto de vetores locais, onde após dividir a imagem em regiões, a convolução é calculada utilizando um kernel, como o Filtro Sobel. Dessa forma é possível criar o histograma de células que são combinados em apenas um histograma que representa a imagem, possibilitando o uso do HOG na detecção de face.

Para calcular o HOG, pode ser realizado a normalização da imagem, deve ser computado o gradiente da imagem, histogramas de gradiente, normalização entre blocos e achatamento em um vetor de recurso.

A imagem é dividida em blocos que possuem células, onde o histograma é compilado de acordo com os pixels dentro de cada célula, conforme Figura 8. O descritor é a concatenação desses histogramas, onde para maior precisão, podem ser normalizados calculando a medida de intensidade de uma região maior (bloco), gerando menor variação às mudanças na iluminação e sombreamento.

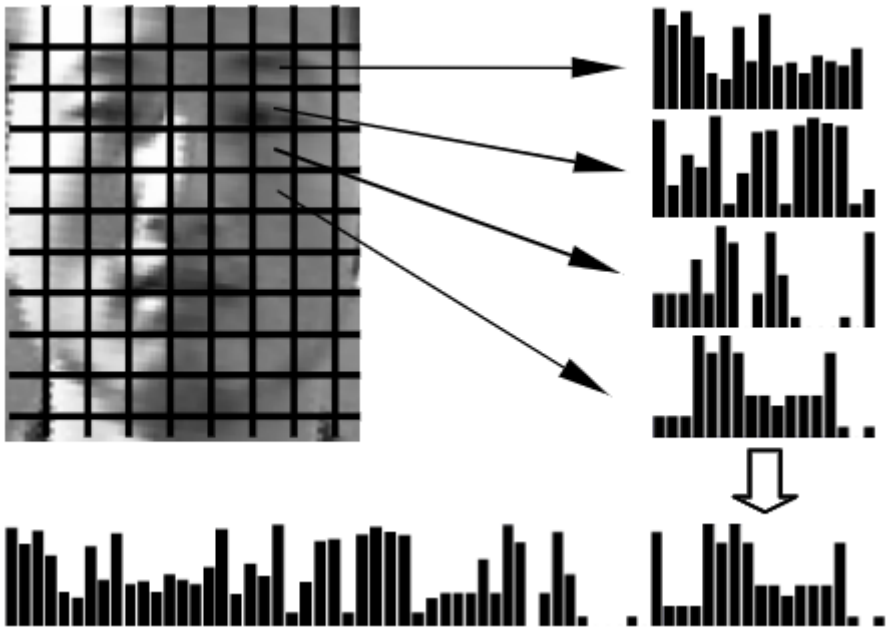
Figura 8 - Células e bloco do HOG



Fonte: (Asmaul Husna et al., 2020)

Para cada célula possui um histograma 1-D local de gradiente ou orientações de borda sobre todos os pixels da célula, onde a combinação desse histograma combinado em nível de célula formando a representação do histograma de orientação, conforme Figura 9.

Figura 9 - Divisão uniforme em células com histogramas 1-D, direções do gradiente acumulados e concatenados no histograma final



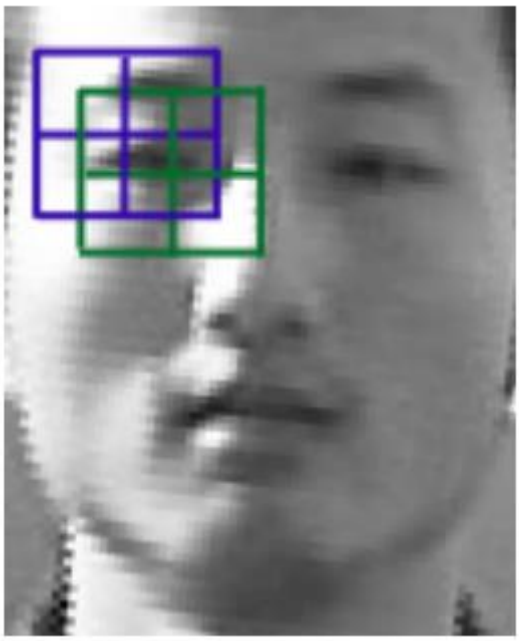
Fonte:(Chang et al., 2011)

Cada histograma de orientação divide a faixa de ângulo de gradiente e as magnitudes de gradiente dos pixels na célula.

Os blocos são normalizados novamente, com o mesmo objetivo inicial, é uma redundância para obtenção um resultado mais refinado.

Por fim, todos os descritores de todos os blocos de uma grade sobreposta, gera um vetor de recursos combinado para uso no classificador (Image, 2011), conforme Figura 10.

Figura 10 - Partes de duas grades diferentes que se sobrepõem



Fonte:(Chang et al., 2011)

Foi demonstrado que a biblioteca OpenCV e Dlib estão sendo utilizadas a muitos anos e que possui os melhores algoritmos, em conjunto com CNN se mostrou interessante em alguns trabalhos similares, mostrando precisão e performance, porém com custo de processamento mais elevado.

3 METODOLOGIA

Atualmente possuem três (3) câmeras instaladas na cabine de cada locomotiva, duas voltadas para dentro da cabine (uma de frente para o maquinista e outra para o auxiliar) e uma para frente da locomotiva, onde as imagens auxiliam na verificação de ocorrências de negligência operacional, vandalismo na cabine e movimento à frente da locomotiva.

Figura 11 - Imagem da câmera instalada na locomotiva

Fonte: De autoria própria

Como a ferrovia possui 288 locomotivas equipadas com câmeras e atualmente possui dificuldade para visualização das imagens, será analisado possibilidades de automatização do processo.

Analisar gravações de vídeo não é um processo eficiente, sendo reativo e não evita uma ocorrência ferroviária, é necessário implementar um algoritmo que possibilite a automação de análise das imagens das câmeras para identificação de condições de fadiga em tempo real. É importante que o algoritmo seja eficiente e robusto, porém não pode possuir custo de processamento elevado devido a necessidade da aplicação, possibilitando a utilização em Raspberry Pi ou *smartphones*.

Foram estudadas diversas bibliotecas em trabalhos similares com o objetivo de analisar as melhores opções para a necessidade, sendo que para o propósito da aplicação do estudo, ocorreu destaque para o OpenCV e Dlib, que são duas bibliotecas que se completam e processam as imagens com agilidade.

Com base nessas informações, será demonstrado o processo que foi realizado para desenvolvimento da detecção.

3.1 Base de dados

As informações dos vídeos das câmeras das locomotivas ficam armazenadas em um servidor de difícil acesso e o acesso as informações é realizada utilizando um software, se torna complexo a tratativas dos dados em larga escala, sendo necessário realizar o processo de forma manual, pois as locomotivas demoram para enviar os dados para o sistema e para estar disponível para download devido utilização operacional.

Como as tratativas precisam ser realizadas em tempo real, o único objetivo para obter os dados é para realização do treinamento do algoritmo para aprendizado.

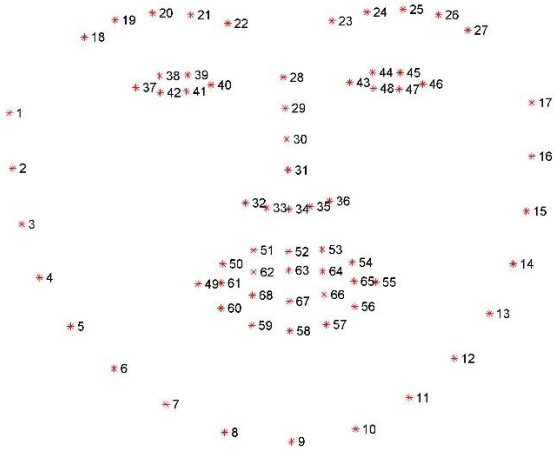
Devido as bibliotecas propostas serem bem evoluídas e possibilitarem a utilização de classificadores já treinados, se tornou viável a realização de teste de implementação para análise de resultados, isso ocorre pois seria necessária grande quantidade de informações para realização do treinamento e como possui variabilidade de maquinistas a depender do local de operação, as faces podem ser bem úteis para atingir o objetivo esperado e conseguir escalabilidade para outras pessoas.

O Dlib também permite treinar o próprio modelo utilizando as ferramentas de aprendizado de máquina da própria biblioteca, onde futuramente é possível realizar melhorias ou modificações do estudo realizado.

O banco de dados utilizado pode ser encontrado na documentação da biblioteca(King, 2013b), onde o modelo treinado *shape_predictor_68_face_landmark* foi utilizado na aplicação.

O conjunto de dados de referência de face iBUG 300-W (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013) foi utilizado pelo modelo treinado e é disponível para download, esse é composto por um total de 300 imagens que foram anotadas com o objetivo de detecção dos marcos faciais de imagens reais. Para o treinamento foram fornecidas 8336 imagens que foram anotadas usando metodologia semi-supervisionada, onde as anotações finais foram corrigidas manualmente por um especialista. A configuração de marco bem estabelecido com o objetivo de garantir precisão no resultado, onde foi utilizado 68 pontos da face, conforme Figura 12. Foi utilizada uma grande diversidade de informações no conjunto de dados, como mudança de local, em ambiente interno e ao ar livre, pessoas com óculos leitura e de sol, utilizando boné e outros utensílios na cabeça, pessoas de diversas regiões do mundo, com rosto altamente expressivos e poses difíceis. Todas as informações foram disponibilizadas publicamente e podem ser encontradas atualmente.

Figura 12 - Marcação dos 68 pontos utilizados nas anotações

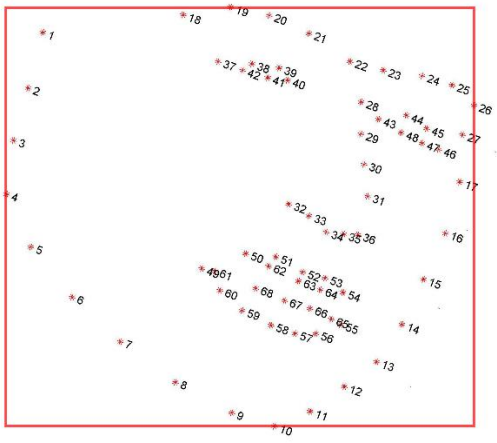


Fonte: (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013)

O conjunto de teste foi realizado com 300 imagens internas e 300 ao ar livre, com o objetivo principal de ter capacidade de lidar com situação adversas, como variação de poses, iluminação, expressões, fundo e qualidade da imagem.

Uma caixa delimitadora de identificação padronizada de detecção de rostos foi definida como calculada pelas anotações de referência, conforme Figura 13. A região facial do treino do detector é baseada nessa restrição.

Figura 13 - Região facial (caixa de delimitação) em que nosso detector facial foi treinado

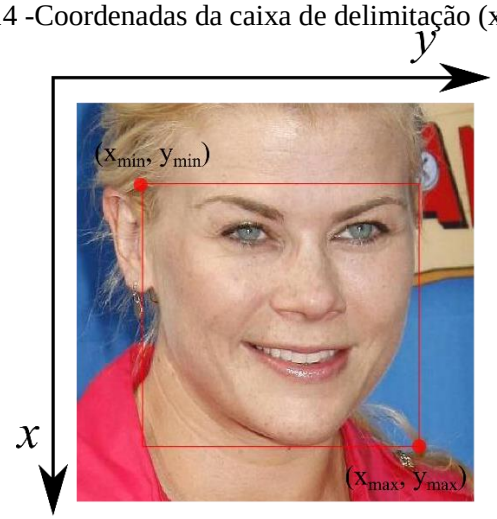


Fonte: (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013)

Como forma de padronização, a imagem de entrada deve ser RGB com extensão .png e a caixa de delimitação deve ser um vetor 4x1 [x_{min}, y_{min}, x_{max}, y_{max}], conforme Figura 14. O

binário de saída deve ser uma matriz 68x2 com os marcos detectados, sendo salva na extensão .pts.

Figura 14 -Coordenadas da caixa de delimitação (x=1, y=1)



Fonte: (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013)

3.2 Algoritmo

Para o desenvolvimento do algoritmo, foi utilizada a linguagem de programação Python, onde foi necessário a instalação das bibliotecas abaixo:

1) OpenCV
2) Dlib
3) Imutils
4) Numpy
5) Playsound

O OpenCv foi utilizado para inserir os pontos e contornos na imagem, transformar para tons de cinza e mostrar informações pertinentes no vídeo analisado. Uma informação importante é que as imagens são representadas por uma matriz na ordem BGR, sendo necessário a utilização da biblioteca NumPy.

O Dlib por sua vez, possui a opção de detecção de *landmarks* da face do rosto, que assume a forma de 68 pontos de referência que ficam localizados em diversas posições, como no canto da boca, sobrancelhas, olhos, nariz e toda lateral do face.

Esse algoritmo utiliza o recurso HOG combinado com classificador linear, onde foram treinados através do *datasets* de referência de face da iBUG, disponibilizado em http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2.

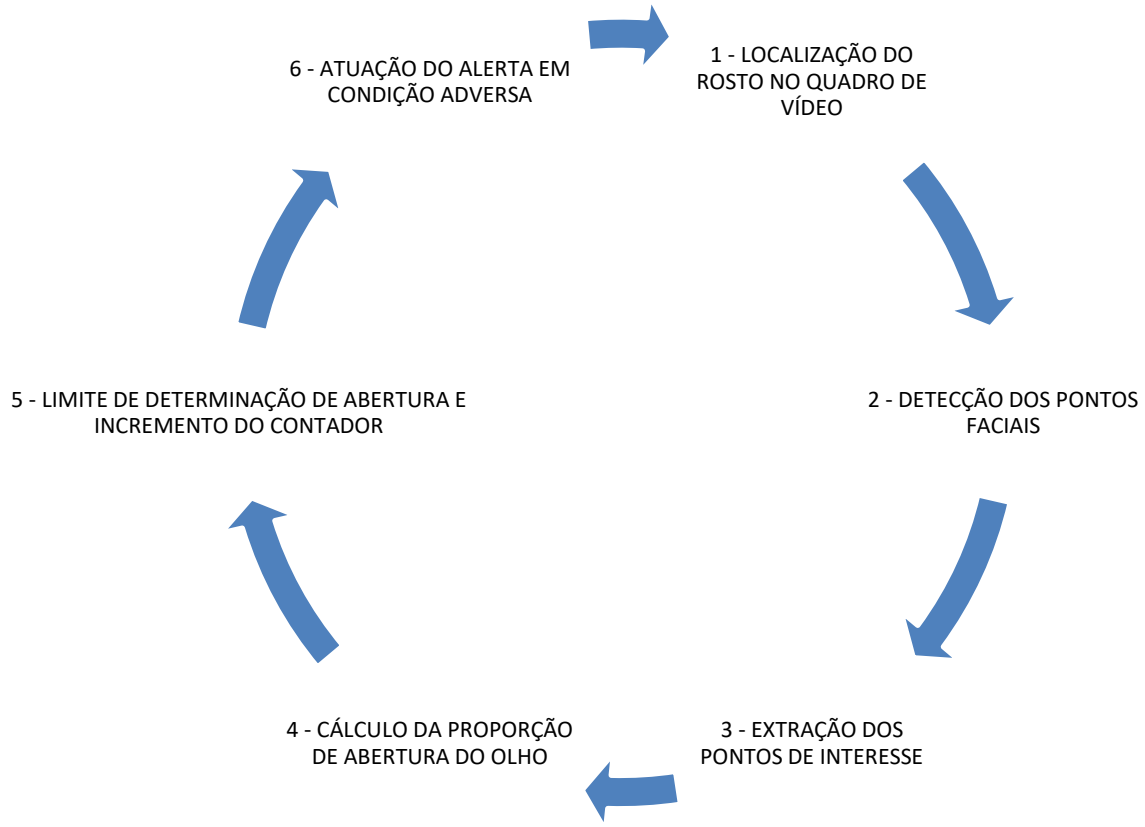
O Imutils foi utilizado, porém não é obrigatório, esse possui como objetivo facilitar a implementação de funções de processamento de imagens, como deslocamento, rotação, redimensionamento, personalização e exibição utilizando o OpenCV.

O Numpy é utilizado pelo OpenCV, pois é a forma como as coordenadas dos pontos de referência facial são convertidas, sendo possível separar os pontos de interesse.

O Playsound foi utilizado apenas para gerar o alarme sonoro no momento da ocorrência, porém também poderia ser utilizado outro método e arquivo.

Para implementação é necessário realizar as etapas demonstradas na Figura 15.

Figura 15 - Etapas de implementação



Fonte: De autoria própria

Antes de realizar a primeira etapa da implementação, é necessário configurar ou preparar o algoritmo para sua utilização.

A câmera que irá realizar a detecção, se for utilizar a webcam padrão geralmente está configurada como 0, sendo possível utilizar câmeras externas, basta configurar corretamente.

Com a entrada configurada, é necessário preparar a leitura do vídeo, possui duas formas *VideoCapture()* da biblioteca OpenCV (Heesch, 2021) ou *VideoStream()* da biblioteca Imutils que utiliza o *VideoCapture()* para realizar a função de forma simplificada, conforme

(Rosebrock, 2018, 2019). Possui a possibilidade de ter diferentes formas adicionais de acesso para diferentes hardwares, como no caso do Raspberry Pi, portanto sempre é recomendado a leitura do manual disponibilizado pelo fabricante.

Também é necessário carregar o detector frontal de face e preditor dos pontos de referência de rosto, onde no caso da biblioteca Dlib foi utilizado o *dlib.get_frontal_face_detector()* e o *dlib.shape_predictor()* respectivamente, conforme (King, 2013).

Para identificar os pontos de interesse foi utilizado *face_utils.FACIAL_LANDMARKS_IDXS[]* da biblioteca Imutils, onde ao digitar *right_eye* e *left_eye*, os pontos dos olhos direito e esquerdo são gerados em uma lista informando o valor inicial e final, conforme (Rosebrock, 2021). Por exemplo, para o olho direito, os pontos de interesse são do 37 a 42 e para o olho esquerdo são do 43 a 48, que são exatamente os pontos correspondentes mencionados na Figura 12.

Com essas variáveis preparadas, é possível iniciar a rotina de detecção, pois como está sendo realizado análise de vídeo, é necessário realizar ciclos (loop) de todos os quadros (frames).

3.2.1 Localização do rosto no quadro de vídeo

Dentro do ciclo (*loop*), deve ser iniciado a leitura da webcam com o comando *read()*, posteriormente é possível redimensionar a imagem com o comando *imutils.resize (frame, width=900)*, essa opção de dimensão pode ser modificada conforme a necessidade de aplicação.

Neste momento é importante transformar o frame para escala de cinza com o comando *cv2.cvtColor("frame", cv2.COLOR_BGR2GRAY)* do OpenCV.

Posteriormente é possível aplicar o detector *dlib.get_frontal_face_detector()* para o *frame* transformado para escala de cinza.

É importante ressaltar que, ao abrir o ciclo de leitura da webcam, é necessário inserir a rotina para fechar e realizar limpeza da biblioteca OpenCV. Neste caso foi utilizado os comandos cv2.destroyAllWindows() e stop().

Como essa rotina não possui uma condição de parada, não será possível finalizar a janela que está mostrando o vídeo, pois o *loop* não é encerrado, portanto é necessário inserir um comando para apertar uma tecla para parar o processo caso tenha interesse, por exemplo: *if cv2.waitKey(1) & 0xFF == ord("q"): break*, onde ao selecionar a tecla "q", a janela de vídeo é fechada.

3.2.2 Detecção dos pontos faciais

Com a face detectada em escala de cinza e os frames sendo lidos em um *array* do Numpy, deve ser realizado um *loop* para verificar cada detecção de face na escala de cinza.

Para cada detecção, é necessário segregar o ponto de interesse com treinamento por dados anotados com as coordenadas dos pontos, conforme informado anteriormente (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013). O comando *predictor()* e *face_utils.shape_to_np()* são necessários para identificar os 68 pontos da face, reduzindo o *array* para um tamanho de 68, que são exatamente os pontos desejados.

A visualização desses pontos pode ser facilitada utilizando o comando *cv2.circle()*, conforme (Heesch, 2022), que irá posicionar os 68 pontos que foram configurados na cor verde para cada uma das posições x e y informados.

3.2.3 Extração dos pontos de interesse

Com os 68 pontos definidos, basta realizar um novo *shape()*, nos valores iniciais desejados *right_eye e left_eye*, restando apenas um *array* de tamanho 6 para cada variável.

3.2.4 Cálculo da proporção de abertura do olho

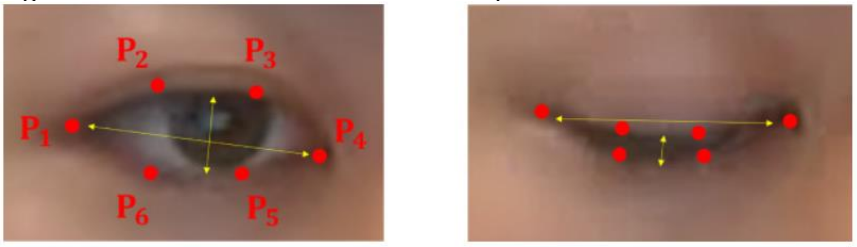
Com as coordenadas dos seis (6) pontos de cada olho, é possível calcular a distância euclidiana individualmente para obter o EAR (*eye aspect ratio*), conforme Equação 1 e Figura 16.

Equação 1 - Cálculo de proporção do olho

$$EAR = \frac{\left||P2 - P6|\right| + ||P3 - P5||}{2||P1 - P4||}$$

Fonte: (Soukupová, 2016)

Figura 16 - Olhos abertos e fechados com pontos de referência detectados



Fonte: (Kuwahara et al., 2022)

Como foi realizado cálculo para cada um dos olhos, é necessário calcular o EAR médio para a tomada de decisão entre olho fechado e aberto.

Este item é o indicador de tomada de decisão principal, visto que o valor de EAR alto, o olho está aberto e EAR baixo, o olho está fechado.

Para facilitar a visualização do valor do EAR médio, foi utilizado o comando *cv2.putText()* da biblioteca OpenCV para inserir o valor encontrado na tela, dessa forma fica mais fácil de identificar o valor encontrado e calibrar a atuação de alarmes.

Foi utilizado dois comandos para demonstrar a detecção dos pontos do olho no vídeo em tempo real, *cv2.convexHull()* e *cv2.drawContours()*, conforme (Heesch, 2022b, 2022c), um para identificar o contorno do olho e outro para desenhar no vídeo respectivamente, sendo inserido a cor vermelha para facilitar a visualização.

3.2.5 Limite de determinação de abertura e incremento do contador

O valor de EAR é a proporção mais esperada, com isso é necessário calibrar um limitador para que algo ocorra caso o olho atinja o limite por um determinado período. Neste caso foi realizado teste e constatado que a melhor assertividade é com 0,25, isso pois é um valor intermediário entre o olho fechado e aberto.

O EAR precisa atingir determinado limite para atuação, porém se não for configurado o valor de tempo, será gerado alarme até ao perceber uma piscada do olho. Para configurar esse tempo, é necessário tratar o vídeo em quantidade de frames, neste caso foi inserido 40 como valor para o contador.

Esses valores foram definidos com testes práticos, com o objetivo de atuação apenas quando realmente necessário e podem ser ajustados com facilidade no algoritmo caso necessário.

3.2.6 Atuação do alerta em condição adversa

Caso os limites do valor EAR e contagem de frame sejam atingidos, é necessário que ocorra sinalização da ocorrência de fadiga, logo, caso o valor esteja dentro do limite, não deve ser sinalizado e o valor de contagem de *frame* deve ser zerado. Foi utilizado dois tipos de alertas, o sonoro e visual (mensagem na tela).

O sinal sonoro foi gerado utilizando a biblioteca *playsound*, onde foi utilizado um áudio de sirene para alertar a ocorrência, assim que o olho volta à normalidade, a alerta é desligado.

A mensagem alerta de fadiga foi inserida na tela com o comando *cv2.putText()*, com acionamento e desligamento em conjunto com o sinal sonoro. Neste item possui um ponto de atenção, o texto não pode ficar na mesma posição do EAR, é necessário deslocar uma das mensagens para não criar sobreposição.

3.3 Outros preditores utilizados no DLIB

Na biblioteca Dlib, possuem outros preditores que podem ser verificados com o objetivo de melhorar o resultado.

Adotando a mesma metodologia informada anteriormente, foi realizado alteração apenas no preditor treinado para o *shape_predictor_68_face_landmarks_GTX* (King, 2021), porém ao realizar teste prático de detecção, foi verificado que o comportamento não foi satisfatório, perdendo o posicionamento dos 68 pontos da face diversas vezes.

O preditor foi treinado na mesma base de dados iBUG 300-W (C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, 2013) e como o modelo resultante é 4,4 vezes menor, a proposta é que o sistema fique mais rápido, suave e preciso.

Foi constatado sensibilidade do algoritmo quanto ao alinhamento da face e pontos de referências possuem dificuldade na detecção dos pontos de referência da face. O resultado encontrado demonstra rostos na posição frontal, conforme Figura 17.

Figura 17 - Comparação da detecção do GTX

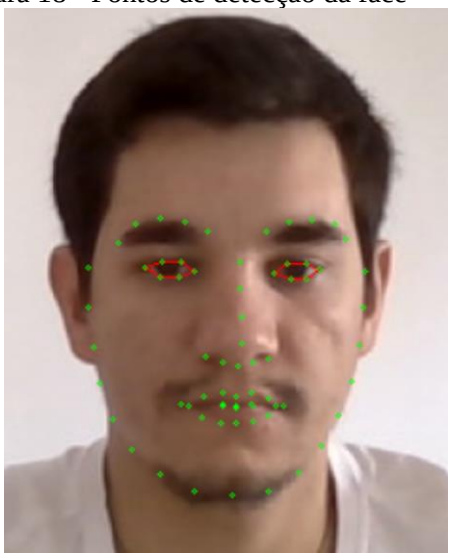
Fonte: (Casado & López, 2021)

4 RESULTADOS

Com o objetivo de avaliar o algoritmo, foi realizado alguns testes práticos para verificar o comportamento das etapas que foram implementadas na Figura 15, tentando identificar fragilidades em diferentes situações, que podem ocorrer durante a detecção.

Inicialmente foi avaliado se a detecção da face está ocorrendo da maneira correta com a extração dos 68 pontos de interesse, os pontos foram definidos na cor verde. Além dos pontos de interesse, também foi avaliado a região de interesse do olho para cálculo do EAR e definido com uma linha na cor vermelha. Todos os pontos funcionaram corretamente, conforme Figura 18.

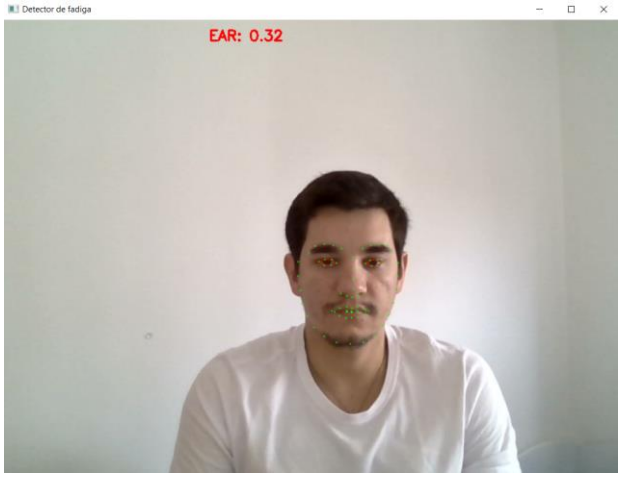
Figura 18 - Pontos de detecção da face



Fonte: De autoria própria

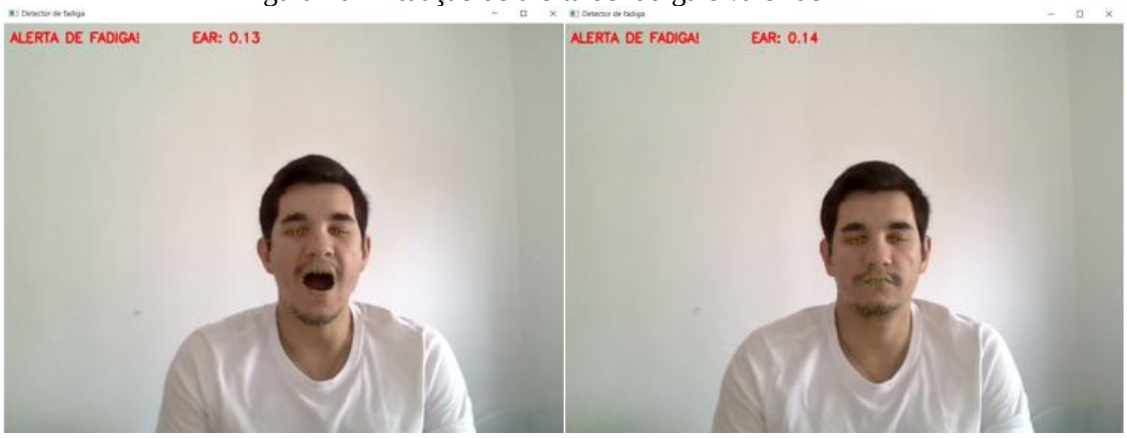
Com todos os pontos funcionando corretamente, foi verificado o valor do EAR e atuação do alarme de detecção, conforme Figura 19 e Figura 20.

Figura 19 - Sem alerta de fadiga e valor do EAR



Fonte: De autoria própria

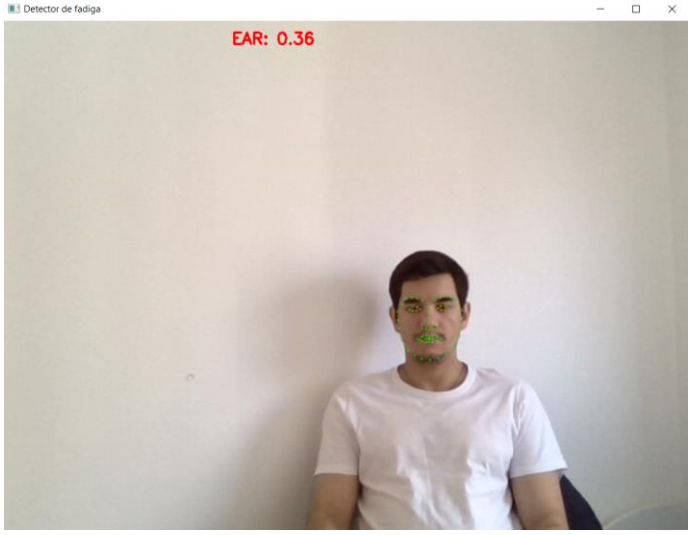
Figura 20 - Atuação do alerta de fadiga e valor do EAR



Fonte: De autoria própria

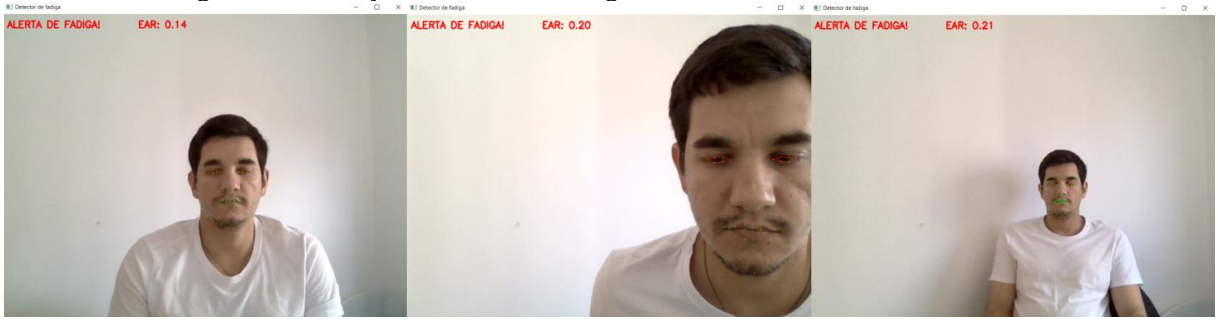
O fator distância também foi avaliado, neste caso, para identificação da face e atuação do detector, onde a pessoa pode ficar muito próxima ou distante da câmera. Em ambas as comparações o resultado foi satisfatório e assertivo na detecção de fadiga, conforme Figura 21 e Figura 22.

Figura 21 - Detecção do rosto com distância maior



Fonte: De autoria própria

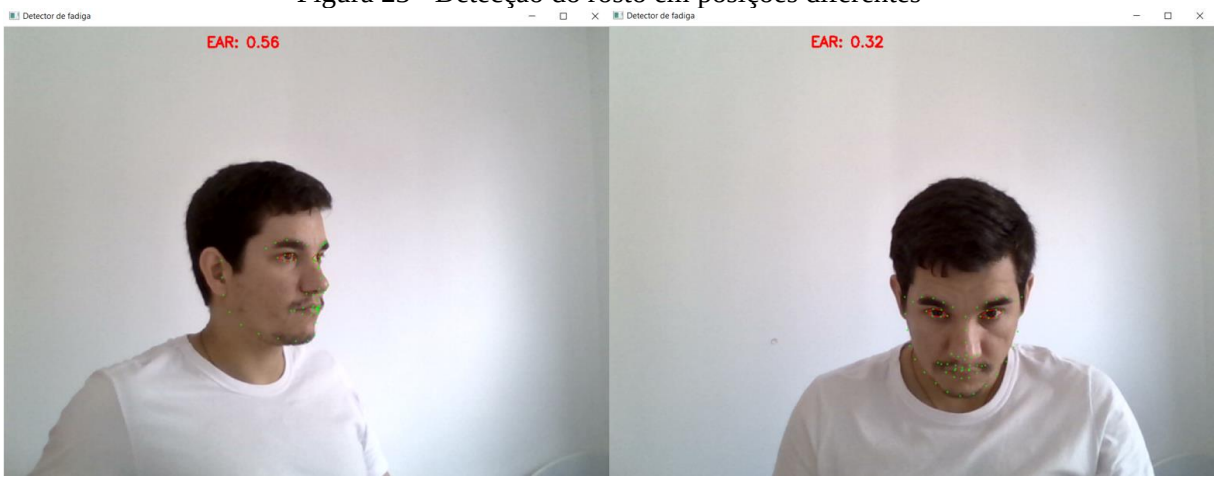
Figura 22 - Atuação do alerta de fadiga em distâncias diferentes da câmera



Fonte: De autoria própria

Até o momento, todos os testes foram realizados com a identificação frontal, porém é necessário avaliar a inclinação do rosto para verificar o comportamento do sistema. Em diversas posições diferentes a detecção ocorreu corretamente e obteve comportamento satisfatório, conforme Figura 23.

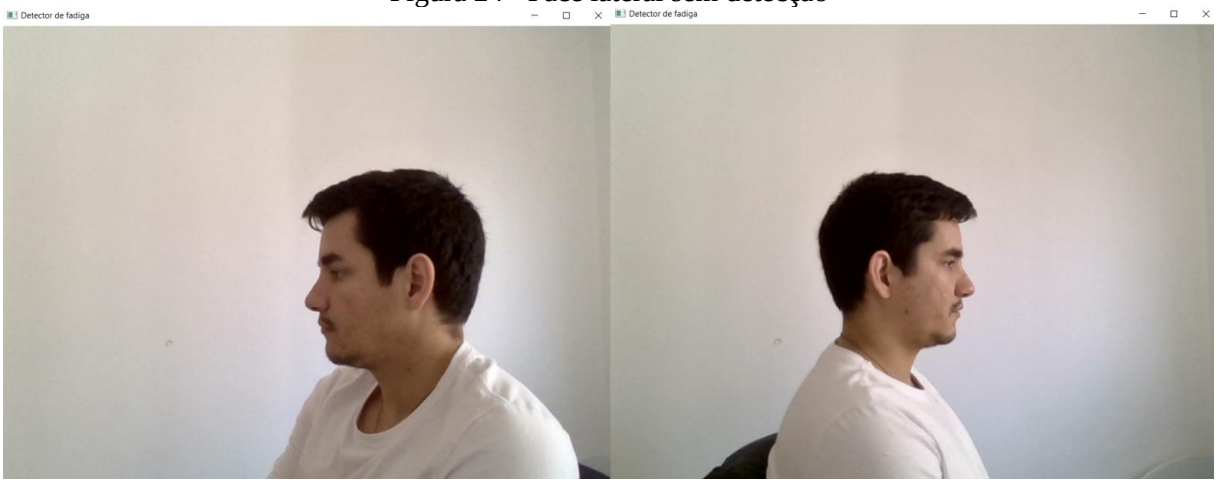
Figura 23 - Detecção do rosto em posições diferentes



Fonte: De autoria própria

O único ponto negativo durante a detecção, é caso o rosto esteja totalmente lateral, dessa forma o sistema perde a funcionalidade por limitação do algoritmo, porém em nenhuma situação normal de utilização, essa situação irá ocorrer, conforme Figura 24.

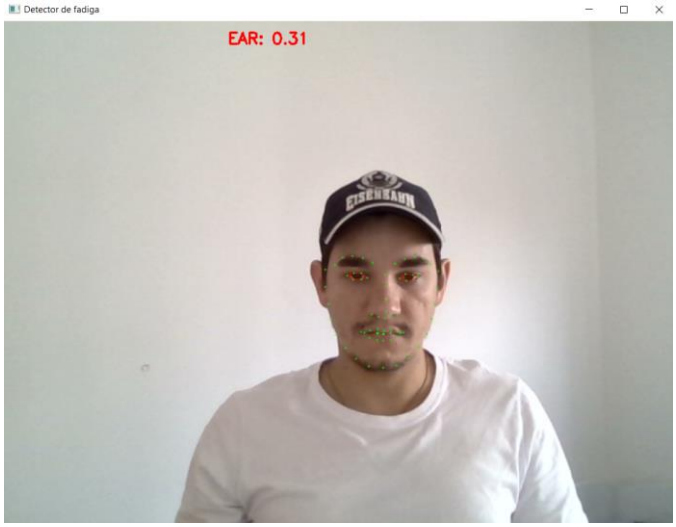
Figura 24 - Face lateral sem detecção



Fonte: De autoria própria

Foi inserido objetos como boné para verificar a influência na detecção da face, neste caso nada foi alterado no comportamento do sistema, conforme Figura 25.

Figura 25 - Detecção da face utilizando boné

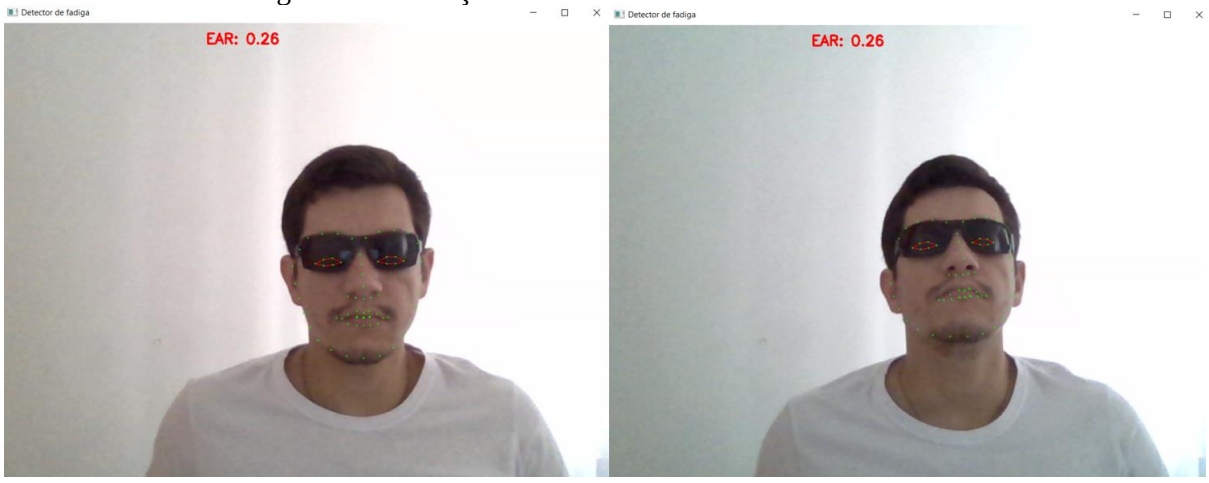


Fonte: De autoria própria

Desafiando a capacidade de detecção, foi inserido um óculo escuro para dificultar o funcionamento e muita luminosidade, o sistema ficou mais sensível ao inserir os pontos de detecção, conforme Figura 26 e Figura 27. O algoritmo continuou realizando o alerta de fadiga, conforme Figura 28.

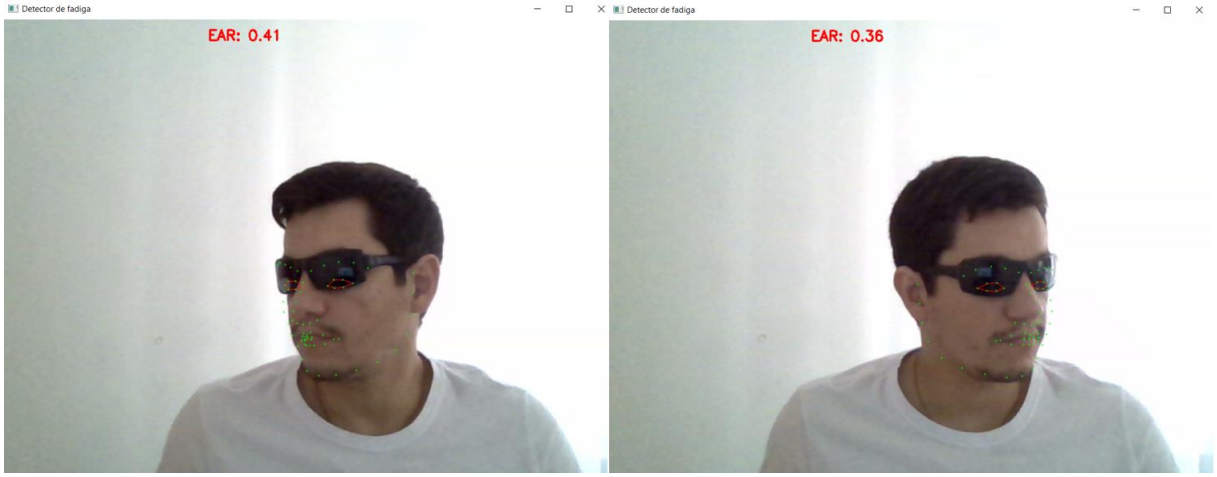
Com o objetivo de tentar enganar o sistema, foi inserido um animal de estimação, o sistema foi capaz de realizar a distinção do que deve ser identificado e atua corretamente, conforme Figura 29 .

Figura 26 - Detecção da face frontal utilizando óculos escuro



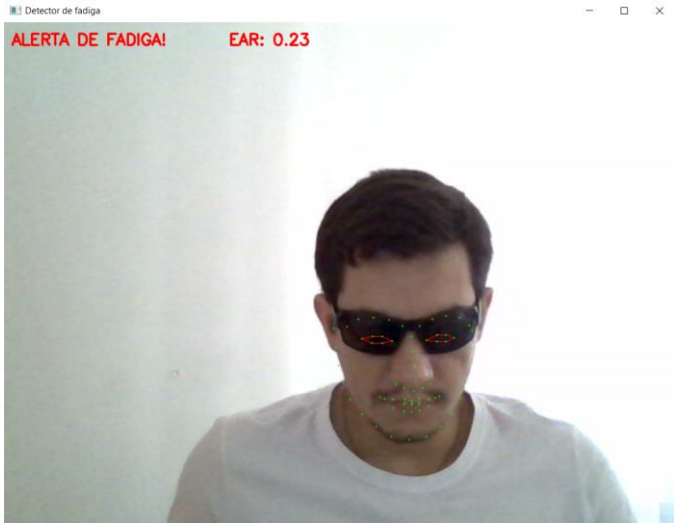
Fonte: De autoria própria

Figura 27 - Detecção da face rotacionada utilizando óculos escuro



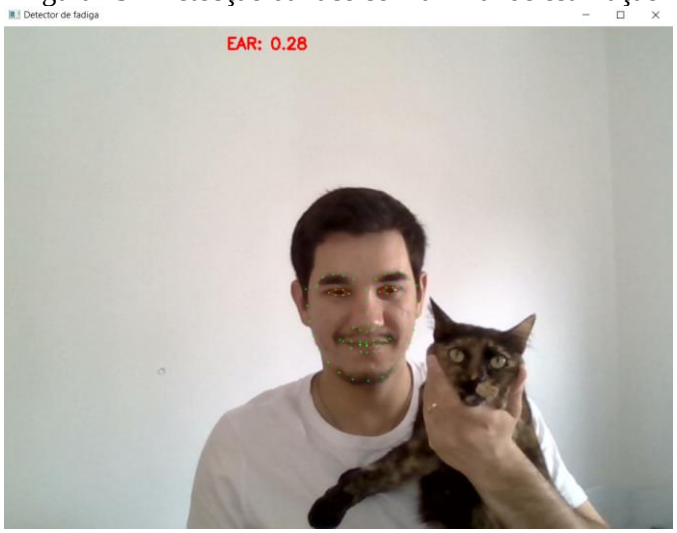
Fonte: De autoria própria

Figura 28 - Alerta de fadiga com óculos escuro



Fonte: De autoria própria

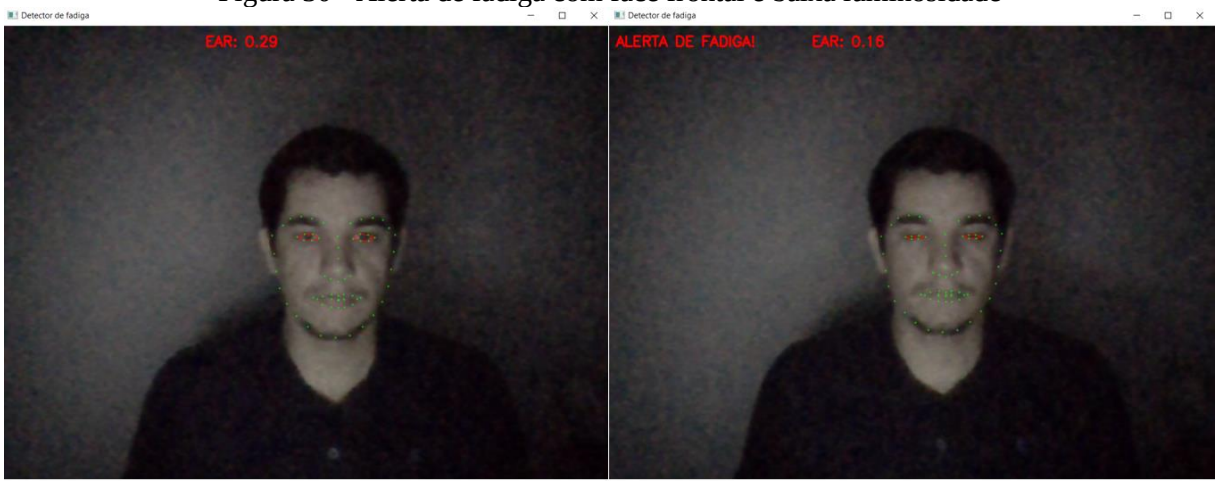
Figura 29 - Detecção da face com animal de estimação



Fonte: De autoria própria

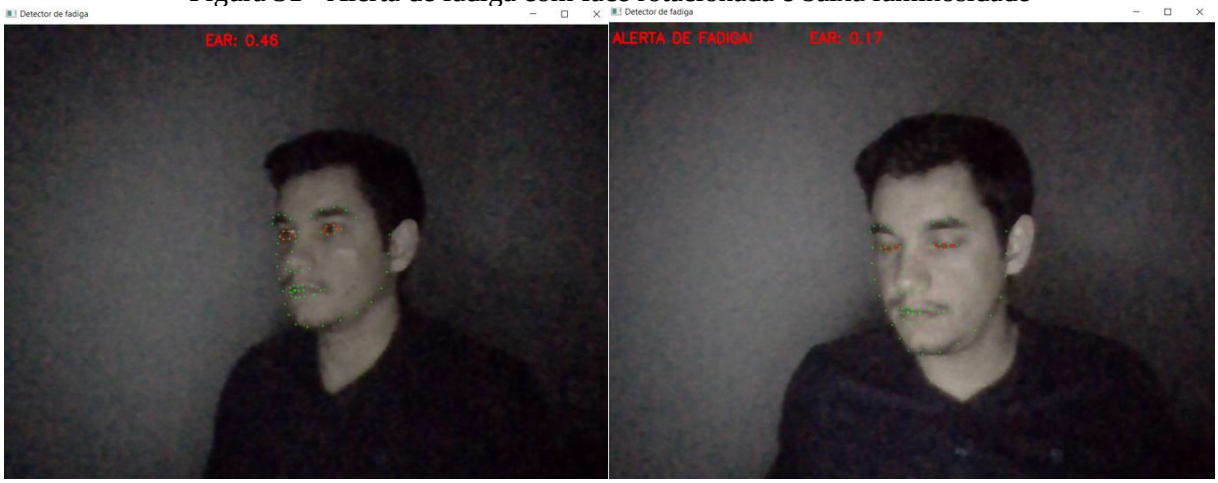
Mantendo a lógica de avaliação, também foi verificado o comportamento em ambiente praticamente sem luminosidade, alterando o rosto em posicionamento frontal e rotacionado, modificando o posicionamento com relação a câmera, mais próximo e distante. Em todos os casos, o comportamento foi muito similar ao que possui luminosidade natural, o resultado foi satisfatório, sendo possível verificar a atuação do alerta de fadiga em todas as ocasiões, conforme Figura 30, Figura 31 e Figura 32.

Figura 30 - Alerta de fadiga com face frontal e baixa luminosidade



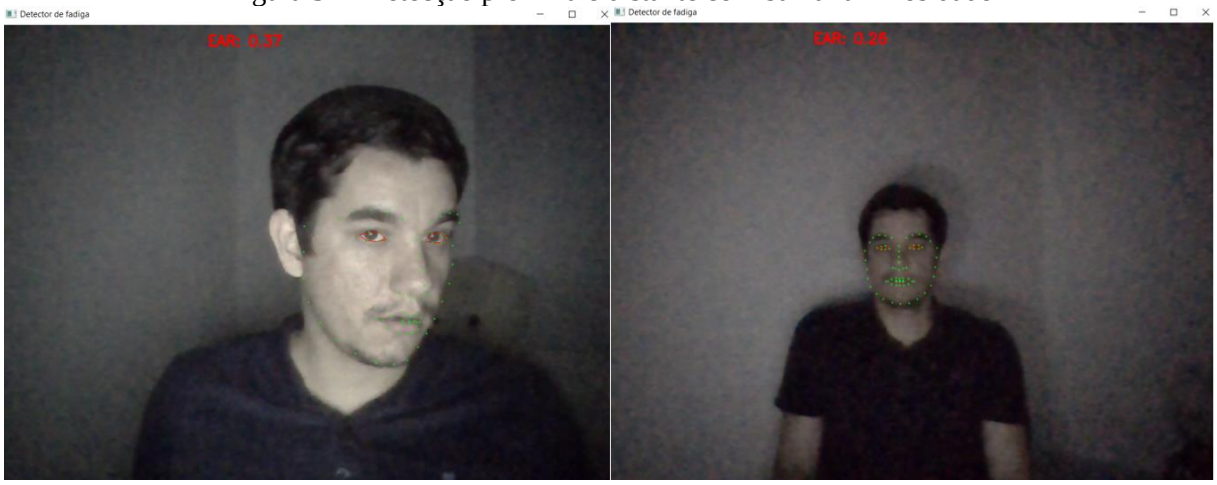
Fonte: De autoria própria

Figura 31 - Alerta de fadiga com face rotacionada e baixa luminosidade



Fonte: De autoria própria

Figura 32 - Detecção próxima e distante com baixa luminosidade



Fonte: De autoria própria

Para manter a sequência de avaliação, foi realizado o mesmo processo com uma pessoa do sexo feminino e o algoritmo se comportou conforme era esperado, as detecções ocorreram adequadamente, conforme Figura 33 e Figura 34. Interessante ressaltar que a mulher utilizava óculo de grau e não ocorreu interferência no funcionamento do algoritmo.

Figura 33 - Pontos de detecção da face

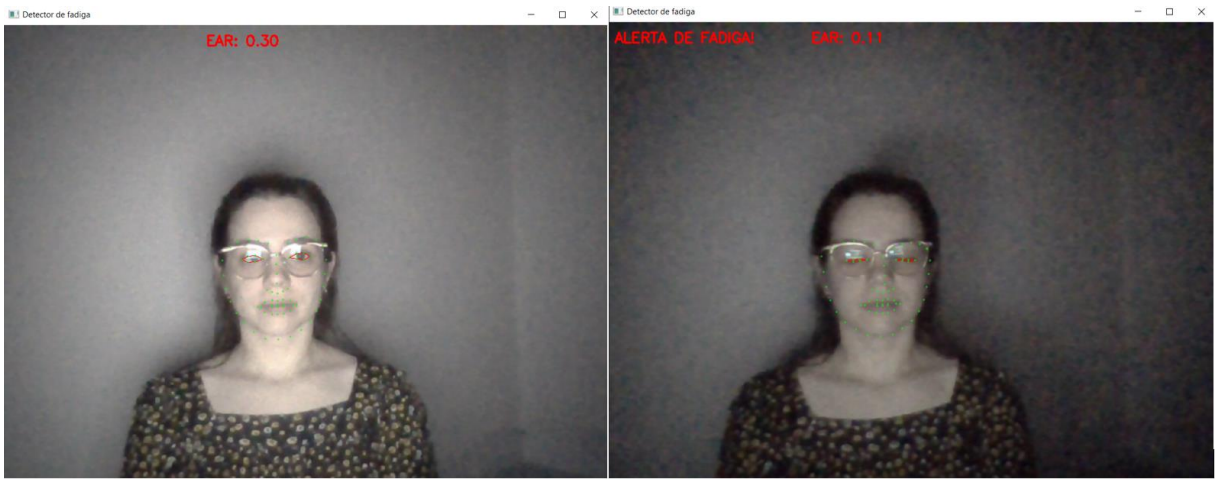
Fonte: De autoria própria

Figura 34 - Alerta de fadiga detectado

Fonte: De autoria própria

Com baixa luminosidade, o comportamento do algoritmo não mudou o comportamento, obtendo performance muito parecida com o encontrado em ambiente luminoso, detecção funcionando perfeitamente, conforme Figura *35*.

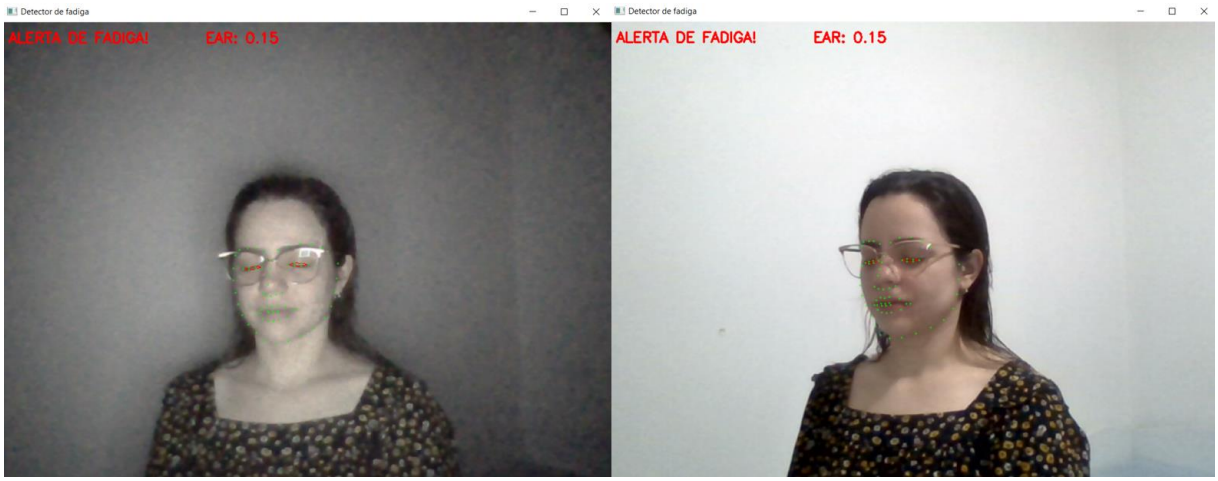
Figura 35 - Detecção de fadiga com baixa luminosidade



Fonte: De autoria própria

Todos os demais testes foram realizados, seguindo a mesma lógica de detecção realizada para o sexo masculino, com o objetivo de identificar anormalidades. O comportamento do detector de fadiga obteve comportamento muito similar quando comparado as pessoas do sexo masculino e feminino, conforme pode ser verificado na Figura 36 e Figura 37.

Figura 36 – Detecção de fadiga com face lateral



Fonte: De autoria própria

Figura 37 - Detecção de fadiga com a face distante

Fonte: De autoria própria

Com diversos testes práticos realizados e comportamento satisfatório do algoritmo, é necessário mensurar a taxa de assertividade durante funcionamento em condição adversas, para isso será gerado um vídeo com aproximadamente 8.000 frames.

Será realizado uma sequência de atuações, onde será anotado atuações indevidas e falhas de atuações de um total esperado, representando a taxa de assertividade do sistema, conforme fórmula da Equação 2:

Equação 2 - Cálculo de assertividade do sistema

$$TX\ (\%) = 1 - \left(\frac{AI + FA}{QTD}\right) x\ 100$$

Onde,

TX é a taxa de assertividade do sistema em porcentagem, define o quanto o sistema acertou de detecção;

AI são as atuações indevidas representados por alertas de fadigas que não deveriam ocorrer e o sistema detectou de forma incorreta;

FA são as falhas de atuações com alertas de fadigas que deveriam ser sinalizados e o sistema não detectou;

QTD é a quantidade esperada de alertas de fadiga.

Seguindo a mesma sequência e lógica de situação adversas, foi realizado o mesmo padrão de teste e mesma sequência de tendência de fadiga, simulando situações com pouca luminosidade, muita claridade, inclinação do rosto em diversas posições, com óculos, com boné, entre outros. Essas posições demonstraram o comportamento para diferentes pessoas.

Tabela 2 - Resultado encontrado com homem e mulher

Pessoa	TX (%)	AI	FA	QTD
Homem	95	0	4	80
Mulher	96,25	0	3	80
Total	**95,63**	0	7	160

Fonte: De autoria própria

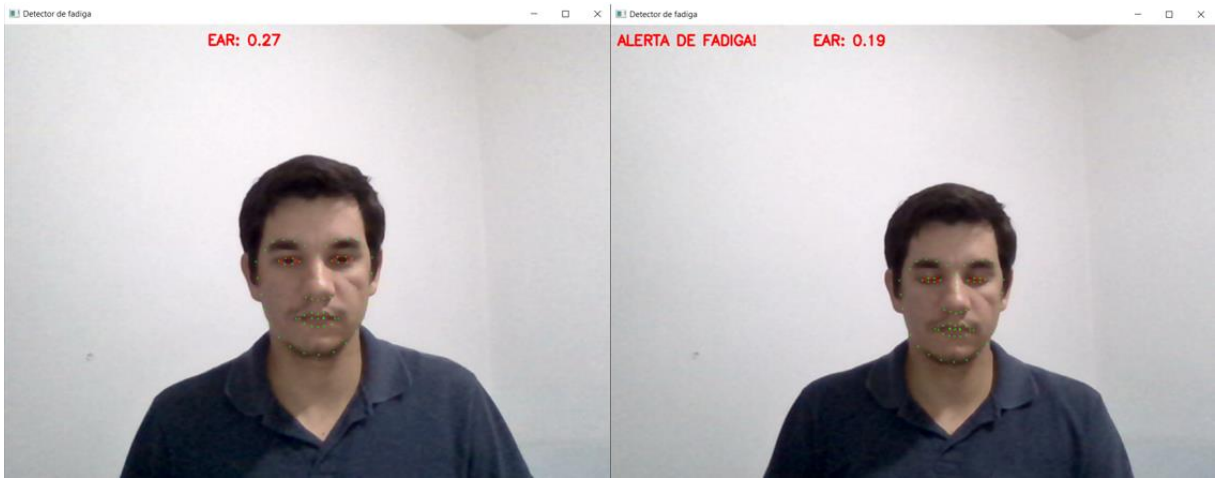
Como pode ser verificado na Tabela 2, o resultado em mulher foi melhor do que no homem, porém com assertividade muito similar. Vale ressaltar que em nenhuma ocasião correu AI, portanto mostra a boa detecção do algoritmo.

As FA ocorreram com o rosto virado lateralmente, em situação noturna em sua maioria, apenas um caso ocorreu falha com óculos, situação que a mulher estava utilizando. Foi possível notar que a referência de linhas que sinalizam o olho se altera, modificando o valor de abertura e deixando de atuar corretamente.

O mesmo padrão de comparação foi realizado com outros preditores, como no caso do *shape_predictor_68_face_landmarks_GTX*, *o* comportamento foi insatisfatório pois se tornou sensível a luminosidade, onde o FA foram frequentes.

Como o rosto próximo e ambiente controlado, o rosto e os 68 pontos foram detectados corretamente, conforme Figura 38.

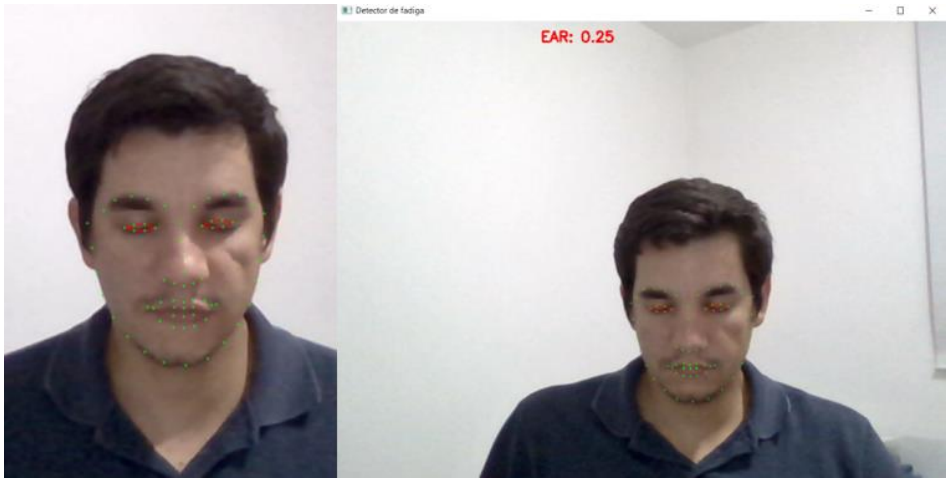
Figura 38 - Detecção de fadiga do predito GTX



Fonte: De autoria própria

Mesmo em situação favorável, foi possível constatar que em algumas ocasiões o algoritmo não se comportou como o esperado, na Figura *39* é possível verificar os olhos totalmente fechados e o algoritmo mostrando o posicionamento aberto.

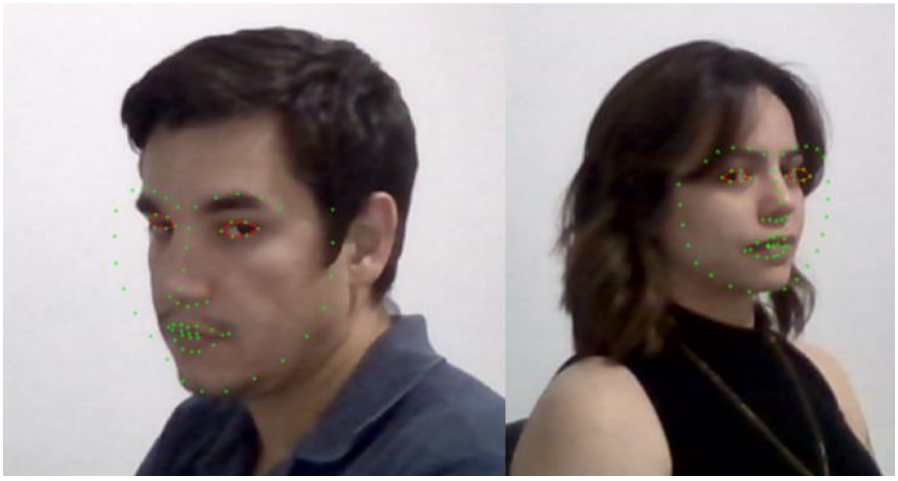
Figura 39 - Preditor GTX perdendo os pontos dos olhos



Fonte: De autoria própria

Ao movimentar o rosto lateralmente, a detecção dos pontos também perdeu a referência, mesmo em ambiente claro, conforme Figura 40.

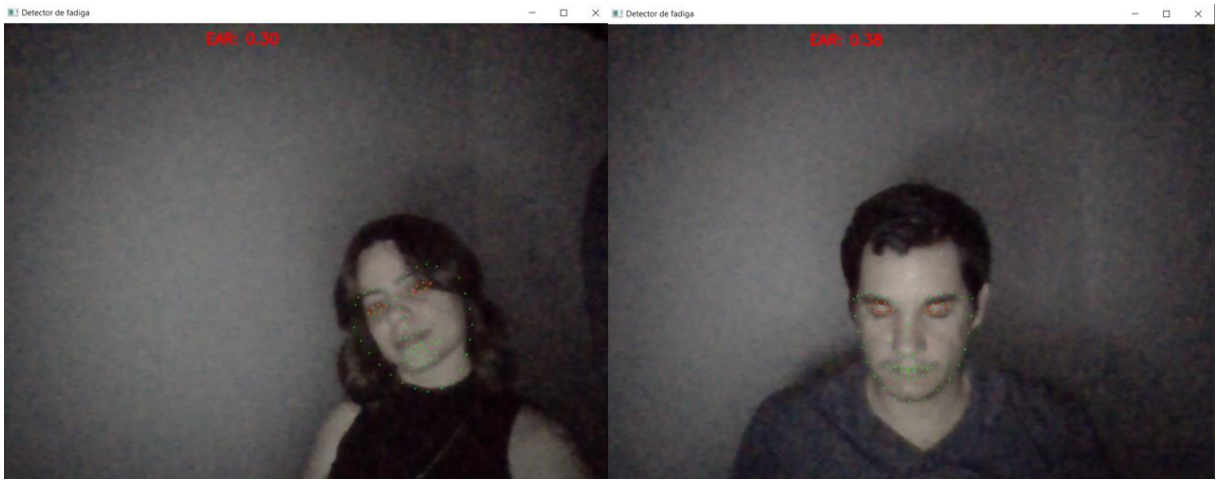
Figura 40 - GTX com perda de referência da face lateral



Fonte: De autoria própria

Ao verificar em ambiente noturno, o resultado encontrado foi pior, onde muitos pontos foram perdidos e FA encontrados, conforme Figura 41.

Figura 41 - Falha de detecção do GTX com baixa luminosidade



Fonte: De autoria própria

Apenas para comparação, foi realizado a mesma sequência e lógica de situação adversas, mantendo o padrão de teste e sequência de tendência de fadiga, onde o resultado e falha de atuações foi significativo, muitas FA, conforme Tabela *3*, principalmente com baixa luminosidade e rosto inclinado para qualquer posição. O algoritmo se comportou melhor com a face de frente.

Tabela 3 - Resultado encontrado com homem e mulher do GTX

Pessoa	**TX (%)**	**AI**	**FA**	**QTD**
Homem	53,75	0	43	80
Mulher	55	0	44	80
Total	**54,37**	0	87	160

Fonte: De autoria própria

Com base nessas informações, o preditor *shape_predictor_68_face_landmarks_GTX*, foi considerado ineficiente e de baixa performance, quando comparado com o *shape_predictor_68_face_landmarks.*

5 CONCLUSÃO

A utilização de imagens para detecção de fadiga é essencial para identificação do desvio, sendo importante ressaltar que para a aplicação proposta, é necessário análise das imagens dos vídeos em tempo real para detecção.

Como o objetivo é atuar de forma eficiente e proativa, foi realizado o trabalho de pesquisa de diversos algoritmos e utilizado as melhores práticas para a definição da metodologia aplicada neste trabalho.

Possui diversas bibliotecas disponíveis para tratamento de imagens, sendo que não foi encontrada aplicação direta de fadiga na ferrovia, porém possui trabalhos similares com objetivos de detecção em outras áreas de aplicações.

Com base no trabalho de pesquisa, as bibliotecas OpenCV e Dlib se demonstraram com sendo as mais completas, foram fundamentais para a tratativa do problema e conforme demostrado, atenderam a necessidade com agilidade, precisão e aplicabilidade.

Os testes foram realizados em diversas situações, inserindo variáveis não controladas, como luminosidade, óculos, boné, mudança no posicionamento e entre outros. O algoritmo atingiu uma taxa de assertividade de 95,63%, demonstrando ser robusto para aplicação e atendendo a necessidade proposta.

A fragilidade do algoritmo está no posicionamento totalmente lateral, não sendo possível realizar a detecção do rosto e como consequência, impossibilitando a detecção de fadiga. Na aplicação proposta, esse problema não irá impactar o processo, visto que possui 3 câmeras que coletam informação.

Como sugestão de trabalhos futuros, a sequência de busca de novas bibliotecas com o objetivo de melhoria contínua. Também é possível aprimorar o código de forma que realize cálculo de bocejos e associar com a abertura do olho, alguns transdutores também podem ser adicionados para detecção de frequência cardíaca. Como sequência desse trabalho, também é interessante desenvolver um hardware robusto que suporte o ambiente operacional agressivo da ferrovia.

O objetivo proposto foi atendido, sendo possível detectar fadiga em tempo real e com boa assertividade, independente do sexo do (a) maquinista, o que demostra escalabilidade.

REFERÊNCIAS

Asmaul Husna, R., Achmad, A., Ilham, A. A., Zainuddin, Z., & Jaya, A. K. (2020). Early Childhood Gymnastic Motion Recognition System Using Image Processing Technology. *Hasanuddin University*, 1–5. https://doi.org/10.1109/ICT49546.2020.9239482

Bunke, H., Villanueva, J. J., Sánchez, G., & Otazu, X. (2009). Progress in Computer Vision and Image Analysis. *World Scientific*, *73*. https://doi.org/10.1142/7003

C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. (2013). *300 Faces In-the-Wild Challenge (300-W)*. Ibug. https://ibug.doc.ic.ac.uk/resources/300-W/

Casado, C. Á., & López, M. B. (2021). *Real-time Face Alignment · GitLab*. Gitlab.Com. https://gitlab.com/visualhealth/vhpapers/real-time-facealignment

Chang, S., Ding, X., & Fang, C. (2011). Histogram of the Oriented Gradient for Face Recognition *. *Tsinghua Science and Technology*, *16*(2).

Costa, L. J. da, Sousa, T. L. de, Silva, F. A. da, Almeida, L. L. de, Pereira, D. R., Artero, A. O., & Piteri, M. A. (2021). Análise de Métodos de Detecção e Reconhecimento de Faces Utilizando Visão Computacional e Algoritmos de Aprendizagem de Máquina. *Colloquium Exactarum*, *13*(2), 01–11. https://doi.org/10.5747/ce.2021.v13.n2.e354

Da, T., & Paula, S. (2017). *Contributions in Face Detection with Deep Neural Networks*.

de Almeida Noronha, F., Luiz de Almeida, L., Assis da Silva, F., Pandur Albuquerque Cabral, F., & Augusto Siscoutto, R. (2019). Detecção de Fadiga a Partir da Análise de Imagens Faciais. *Colloquium Exactarum*, *11*(2), 34–45. https://doi.org/10.5747/ce.2019.v11.n2.e273

Dor, R. (2022). *Fadiga - Rede D'Or*. Rede Dor. https://www.rededorsaoluiz.com.br/sintomas/fadiga

Filtering, I. (2014). Image Filtering. *Opencv*, *44*, 1–6. https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html

García-Rios, E., Escamilla-Hernández, E., Nakano-Miyatake, M., & Pérez-Meana, H. (2014). Sistema de reconocimiento de rostros usando visión estéreo. *Informacion Tecnologica*, *25*(6), 117–130. https://doi.org/10.4067/S0718-07642014000600015

Heesch, D. van. (2021). *OpenCV VideoCapture Class Reference*. Docs.Opencv.Org. https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html#a57c0e81e83e60f36c83027dc2a188e80

Heesch, D. van. (2022a). *Drawing Functions in OpenCV*. Opencv. https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html

Heesch, D. van. (2022b). *OpenCV: Bibliography*. Opencv. https://docs.opencv.org/3.4/d0/de3/citelist.html#CITEREF_Wiskott97

Heesch, D. van. (2022c). *OpenCV Contours*. Opencv. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

Heesch, D. van. (2022d). *OpenCV Convex Hull*. Opencv. https://docs.opencv.org/3.4/d7/d1d/tutorial_hull.html

Heesch, D. van. (2022e). *OpenCV: reconhecimento facial com OpenCV*. Opencv. https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html

Image, S. (2011). *Histograma de gradientes orientados — skimage v0.20.0.dev0 docs*. Scikit-Image.Org. https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html

Jabbar, R., Shinoy, M., Kharbeche, M., Al-Khalifa, K., Krichen, M., & Barkaoui, K. (2020). Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application. *Arxiv*. http://arxiv.org/abs/2002.03728

King, D. E. (2013a). *Classes dlib documentation*. Dlib.Net. http://dlib.net/python/index.html

King, D. E. (2013b). *dlib C++ Library*. Dlib.Net. http://dlib.net/

King, D. E. (2021). *shape_predictor_68_face_landmarks_GTX.dat.bz2*. Dlib Models. https://github.com/davisking/dlib-models/blob/master/shape_predictor_68_face_landmarks_GTX.dat.bz2

Kuwahara, A., Nishikawa, K., Hirakawa, R., Kawano, H., & Nakatoh, Y. (2022). Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping(EARM). *Cognitive Robotics*, *2*, 50–59. https://doi.org/10.1016/J.COGR.2022.01.003

Michaelis. (2022). *Ferrovia*. Michaelis. https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/ferrovia

Queiroz, K. L. de. (2011). Sistema baseado em vídeo para detecção de sonolência em motoristas. *UnB*. https://repositorio.unb.br/handle/10482/10704

Rosebrock, A. (2018). *Imutils webcamvideostream*. Github.Com/PyImageSearch. https://github.com/PyImageSearch/imutils/blob/master/imutils/video/webcamvideostream.py

Rosebrock, A. (2019). *Imutils videostream*. Github.Com/PyImageSearch. https://github.com/PyImageSearch/imutils/blob/master/imutils/video/videostream.py

Rosebrock, A. (2021). *Imutils helpers*. Github.Com/PyImageSearch. https://github.com/PyImageSearch/imutils/blob/master/imutils/face_utils/helpers.py

Rumo. (2022). *Modelo de Negócio - Rumo*. Rumolog. http://ri.rumolog.com/sobre-a-rumo/modelo-de-negocio/

Senna, A. (1990). *Ayrton Senna*. Pensador. https://www.pensador.com/frase/MTYxNjQx/

Soukupová, T. (2016). Real-Time Eye Blink Detection using Facial Landmarks. *Czech Technical University in Prague*, 8.

Trilla, A., Bob-Manuel, J., Lamoureux, B., & Vilasis-Cardona, X. (2021). Integrated Multiple-Defect Detection and Evaluation of Rail Wheel Tread Images using Convolutional Neural Networks. *International Journal of Prognostics and Health Management*, *12*(1). https://doi.org/10.36001/IJPHM.2021.V12I1.2906