INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Coordenação do Curso de Engenharia Elétrica

Mateus Gomes Correia

Detecção de semáforo utilizando sistemas embarcados e técnicas de aprendizagem de máquina

João Pessoa
2022

Mateus Gomes Correia

Detecção de semáforo utilizando sistemas embarcados e técnicas de aprendizagem de máquina

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como parte dos requisitos para a obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Patric Lacouth da Silva

João Pessoa
2022

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha do IFPB, *campus* João Pessoa

C824d Correia, Mateus Gomes.

Detecção de semáforo utilizando sistemas embarcados e técnicas de aprendizagem de máquina / Mateus Gomes Correia. – 2022.

48 f. : il.

TCC (Graduação – Curso Superior em Engenharia Elétrica) - Instituto Federal de Educação da Paraíba / Coordenação do Curso Superior de Engenharia Elétrica, 2022.

Orientação : Profº Patric Lacouth da Silva.

1.Imagem digital. 2. Visão computacional. 3. Semáforo. 4. Linguagem de programação. 5. Árvore de decisão. I. Título.

CDU 004.932:656.056.4(043)

Lucrecia Camilo de Lima
Bibliotecária - CRB 15/13

Mateus Gomes Correia

Detecção de semáforo utilizando sistemas embarcados e técnicas de aprendizagem de máquina

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como parte dos requisitos para a obtenção do grau de Engenheiro Eletricista.

BANCA EXAMINADORA

Prof. Patric Lacouth da Silva, D.Sc. – IFPB
Orientador

Prof. Lincoln Machado de Araujo, D.Sc. – IFPB
Membro da Banca

Prof. Erik Farias da Silva, D.Sc. – IFPB
Membro da Banca

João Pessoa, 04 de Agosto de 2022.

SERVIÇO PÚBLICO FEDERAL
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA ELÉTRICA

ATA 74/2022 - CCSBEE/UA3/UA/DDE/DG/JP/REITORIA/IFPB

Coordenação do Curso Superior de Bacharelado
em Engenharia Elétrica
CCSBEE-JP

ATA DE APRESENTAÇÃO PÚBLICA E AVALIAÇÃO DE TRABALHO DE CONCLUSÃO DE CURSO

ATA Nº: (Nº / ANO)	**269/2022**

Às quinze horas e trinta minutos do dia quatro db mês de agosto do ano de dois mil e vinte e dois, de modo virtual, foi realizada a Apresentação Pública e Avaliação do Trabalho de Conclusão de Curso intitulado **"DETECÇÃO DE SEMÁFORO UTILIZANDO SISTEMAS EMBARCADOS E TÉCNICAS DE APRENDIZAGEM DE MÁQUINA"**, do aluno **MATEUS GOMES CORREIA**, requisito obrigatório para conclusão do CURSO DE BACHARELADO EM ENGENHARIA ELÉTRICA, com os membros da Banca Examinadora **Patric Lacouth da Silva, DR.** (Orientador, IFPB), **Erik Farias da Silva, DR.** (Examinador, IFPB) e **Lincoln Machado de Araújo, DR.** (Examinador, IFPB). Após a apresentação e as considerações da Banca Examinadora, o trabalho foi considerado **APROVADO**, com nota **100** sendo esta composta pela média aritmética das seguintes avaliações parciais:

Texto:	Apresentação:	Defesa oral:
100	100	100

Eu, **Patric Lacouth da Silva, DR.** (Orientador, IFPB), lavrei a presente Ata, que segue assinada por mim e pelos demais membros da Banca Examinadora.

Observações:

Documento assinado eletronicamente por:

- **Patric Lacouth da Silva, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 15/08/2022 17:20:39.
- **Erik Farias da Silva, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 15/08/2022 17:23:02.
- **Lincoln Machado de Araújo, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 16/08/2022 12:25:17.

Este documento foi emitido pelo SUAP em 15/08/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 325984
Código de Autenticação: 3cc78399b1

Agradecimentos

Aos meus pais Zenaldo e Silvana, meu irmão André e madrinha Vitória que me ensinaram os verdadeiros caminhos, amor, apoio e sempre deram todo suporte na minha jornada.

Ao meu orientador, Prof. Patric Lacouth, que após uma longa jornada de trabalho na execução desta monografia, sempre deu suporte, incentivo e confiança. Suas contribuições e ensinamentos foram fundamentais para o enriquecimento do trabalho e a concretização dos objetivos traçados em 2021.

Aos meus amigos Carlos Henrique, Matheus Lucas e Ruan Castro, companheiros durante anos na formação acadêmica, superando vários desafios e crescendo como profissional e pessoa a cada semestre.

À todos os professores e amigos construídos no meu período de graduação no IFPB, pelo incentivo, exemplo e ensinamentos que me ajudaram a ter competência na realização deste trabalho.

RESUMO

Os veículos autônomos devem ser capazes de perceber semáforos e reconhecer seus estados presentes, além de seguir seu caminho corretamente. O trabalho propõe a interpretação de imagens por uma câmera e executa a tomada de decisão para o veículo autônomo construído, com pouco poder computacional e embarcado com Raspberry Pi. Utilizando visão computacional para seguir a trajetória da pista e reconhecimento de semáforo com o algoritmo *Haar Cascade* para a identificação e o algoritmo de Árvore de Decisão para a classificação do sinal. No final dos experimentos, o modelo construído foi capaz de classificar corretamente os sinais do semáforo com 80% de precisão.

***Palavras-chave**: Visão Computacional, Detecção de Semáforo, Árvore de Decisão, Haar Cascade*

ABSTRACT

Autonomous vehicles must be able to perceive traffic lights and recognize their present states, in addition to following their path correctly. The work proposes the interpretation of images by a camera and performs the decision making for the autonomous vehicle built, with little computational power and embedded with Raspberry Pi. Using computer vision to follow the track trajectory and traffic light recognition with the Haar Cascade algorithm for the identification and the Decision Tree algorithm for the signal classification. At the end of the experiments, the built model was able to correctly classify the semaphore signals with 80% accuracy.

Keywords: *Computer Vision, Traffic Light Detection, Decision Tree, Haar Cascade*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGV	*Veículo Guiado Automatizado*
AMR	*Robô Móvel Autônomo*
CC	*Corrente Contínua*
EVA	*Acetato-Vinilo de Etileno*
HSV	*Hue, Saturation e Value*
IFPB	*Instituto Federal de Educação, Ciência e Tecnologia da Paraiba*
LED	*Diodo Emissor de Luz*
MDF	*Fibras de Média Densidade*
OpenCV	*Open Source Computer Vision Library*
RGB	*Red, Green e Blue*
RMA	*Robô Móvel Autônomo*
RPM	*Rotações Por Minuto*
WEB	*World Wide Web*
XML	*Extensible Markup Language*
pixel	*Picture Element*

SUMÁRIO

1 INTRODUÇÃO

A segurança no trânsito é uma preocupação constante e demanda um esforço global em torno dela. No Brasil, a taxa de mortalidade por lesões de trânsito é de 15,2 óbitos por 100 mil habitantes, (SIDRA, 2019). Uma das medidas para organizar o trânsito e evitar acidentes é a utilização de semáforos. Entretanto, segundo (SENATRAN, 2022), em 2021 foram registrados 2444725 avanços ao sinal vermelho do semáforo no Brasil. Um comportamento que gera muitos acidentes e dependem da tomada de decisão e reação humana, o que em vários casos não são efetuados com sucesso.

A visão computacional é um tema que está em franca ascensão. Ela é ponto-chave para o desenvolvimento de diversas tecnologias, principalmente na área de automação de veículos. Possuindo inúmeras aplicações: reconhecimento facial, inspeção de linhas de produção, inspeção de elementos na área de saúde, entre muitas outras. A semelhança com o modo de percepção humana do ambiente a torna uma ferramenta extremamente poderosa. Este é um sentido que já é bastante dominado pela raça humana, pois podemos facilmente distinguir dois objetos distintos apenas ao olhar para eles, e classificá-los em categorias, (COELHO, 2011).

Utilizado no trabalho, o OpenCV, uma ferramenta de programação desenvolvida para aplicações de visão computacional (BRADSKI; KAEHLER, 2008). A qual possui módulos para processamento de imagens e algoritmos de reconhecimento de objetos. Ajuda e agiliza no desenvolvimento das aplicações experimentadas neste trabalho.

Junto com o protótipo de um carro, em tamanho reduzido, e uma pista de teste com semáforo, é aplicado a visão computacional para a tomada de decisão do veículo. Sendo assim, neste trabalho é construído um robô móvel autônomo (RMA). Os RMAs podem reconhecer o ambiente em que estão inseridos e através da leitura de sensores e câmeras, podem fazer uma análise e planejar melhor suas ações, (HEINEN, 2002).

Os capítulos deste trabalho estão organizados da seguinte forma: no Capítulo 2, é apresentado a fundamentação teórica, o embasamento teórico para o desenvolvimento da pesquisa. No Capítulo 3, é mostrado os materiais e metodologia aplicada. No Capítulo 4, são mostrados os resultados obtidos. Por fim, no Capítulo 5, é apresentado a conclusão do trabalho.

2 Fundamentação teórica

Usar o computador para examinar uma imagem é transformar os dados de uma câmera fotográfica ou de vídeo em uma decisão ou uma nova representação. Todas essas transformações são feitas para alcançar algum objetivo específico. Os dados de entrada podem incluir algumas informações contextuais como "a câmera está montada em um carro". A decisão pode ser "há um semáforo nesta cena" ou "há um sinal vermelho à frente". Uma nova representação pode significar transformar uma imagem colorida em uma imagem em tons de cinza ou recortar o objeto principal da imagem, (BRADSKI, 2017).

Neste capítulo, são descritos os embasamentos teóricos utilizados para o desenvolvimento do projeto.

2.1 Imagens

A palavra imagem provem do latim, *imāgo*, que significa retrato, semelhança ou representação. É uma figura ou representação visual de algo ou alguém. Ou seja, uma pintura, um retrato, um desenho, uma fotografia ou um vídeo são imagens.

As imagens podem tentar representar a realidade e possuir uma função simbólica, como o caso de sinais de trânsito, bandeiras e sinais associados à comunicação visual. Os seus significados possuem uma grande interferência em seu contexto cultural. É o caso de semáforos, mesmo que em quase todos os países as cores de sinalização sejam sempre vermelho ("pare"), amarelo ("atenção") e verde ("siga"), há no Japão alguns semáforos que trocam a cor verde pela cor azul.

Uma imagem contém uma imensa quantidade de informações e um observador humano interpreta globalmente e qualitativamente. Do ponto de vista da computação, uma imagem é um conjunto de pontos que convergem num plano, e de forma abstrata é um suporte para realizar trocas de informações. Na Tabela 1, mostra a comparação entre a visão humana e a visão computacional, em relação a flexibilidade, habilidade e sensibilidade, adaptado de (NETO, 1999).

2.1.1 Imagem Digital

Uma imagem pode ser definida como uma função bidimensional, f(x, y), onde x e y são coordenadas espaciais (planas), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. Quando x, y e os valores de intensidade de f são todos quantidades finitas e discretas, chamamos a

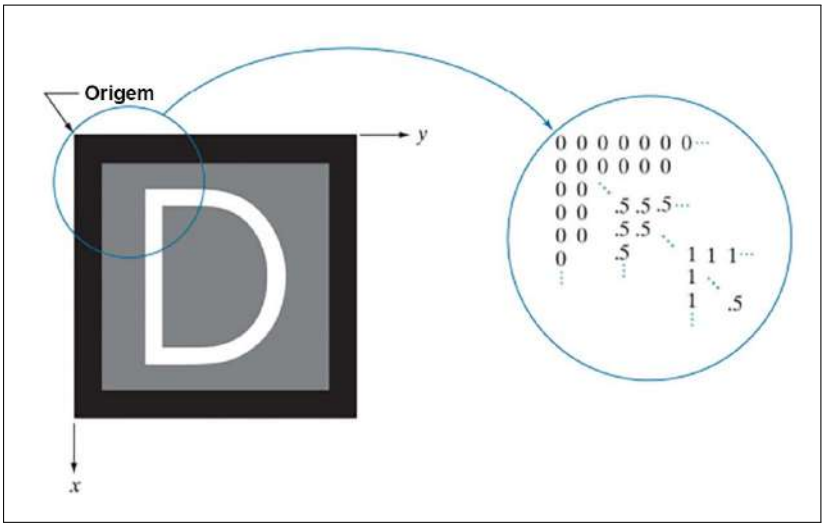
	Visão humana	**Visão Computacional**
Flexibilidade	Extremamente flexível, capaz de se adaptar a diferentes tarefas e condições de trabalho.	Normalmente inflexível, apresenta bom desempenho somente na tarefa para a qual foi projetado.
Habilidade	Pode estabelecer estimativas relativamente precisas em assuntos subjetivos.	Pode efetuar medições exatas, baseadas em contagem de pixels e, portanto, dependentes da resolução da imagem digitalizada.
Sensibilidade	Capaz de se adaptar a diferentes condições de luminosidade, características físicas da superfície do objeto e distância ao objeto. Limitado na distinção de muitos níveis diferentes de cinza, simultaneamente.	Sensível ao nível e padrão de iluminação, bem como à distância em relação ao objeto e suas características físicas. Pode trabalhar com centenas de tons de cinza, conforme projeto do digitalizador.

Tabela 1 – Comparação entre a visão humano e a visão computacional

imagem de imagem digital. Representando uma cena por meio um conjunto de elementos discretos, cada um com uma localização e valor, chamados de *pixels*, (RICHARD, 2018).

Suponha uma imagem digital, f(x, y), contendo M linhas e N colunas, onde (x, y) são coordenadas discretas. Assim, por exemplo, o valor da imagem digital na origem é f(0, 0), e seu valor nas próximas coordenadas ao longo da primeira linha é f(0, 1). A notação (0, 1) é usada para denotar a segunda amostra ao longo da primeira linha. Em geral, o valor de uma imagem digital em qualquer coordenada (x, y) é denotado por f(x, y), onde x e y são valores inteiros.

Figura 1 – Representação da matriz de intensidade 2D de uma imagem



FONTE: Adaptado de (RICHARD, 2018)

Na Figura 1, mostra um exemplo de f(x, y) em uma imagem e a representação de sua matriz, como é observada pelo computador. A intensidade de cada ponto na tela é proporcional ao valor de f naquele ponto.

2.1.2 Representação RGB e HSV

O RGB vem das iniciais das cores em inglês *red*, *green* e *blue* (vermelho, verde e azul). Utilizado nos monitores de computadores, câmeras digitais, televisões, entre outros equipamentos. Com a mistura dessas três cores é possível atingir outras, em uma escala de 0 a 255 para cada cor, como a cor branca quando as três cores possuem o valor máximo (255) e a cor preta quando estão com o valor mínimo (0).

Na Figura 2, o primeiro exemplo, observa-se a soma das cores primárias, RGB, produzindo cores secundárias da luz (magenta, ciano e amarelo). Misturar as três primárias, ou uma secundária com sua cor primária oposta, nas intensidades certas, produz luz branca. Já no segundo exemplo, observa-se a subtração das cores primárias (magenta, ciano e amarelo), gerando as secundárias, RGB. Uma combinação adequada dessas três cores primárias, ou uma secundário com seu primário oposto, produz preto.

Embora seja possível adivinhar a combinação de valores para algumas cores, como amarelo (quantidade iguais de vermelho e verde), cores menos puras são muito difíceis de calcular nesse modelo de cores. Por exemplo, encontrar a combinação para um roxo escuro não é tão simples, pois os humanos não pensam nas cores como misturas de luzes vermelhas, verdes e azuis.

O espaço de cores HSV das iniciais em inglês *hue*, *saturation* e *value* (matiz, saturação, luminosidade), é um modelo de cor cilíndrica que remapeia as cores primárias RGB em dimensões que são mais fáceis de entender para os humanos. Através das suas grandezas quantitativas especificáveis e independentes. Dependendo dos valores dados a matiz, saturação e luminosidade obtém cores diferentes, (ALMEIDA, 2012).

A Figura 3 mostra o espaço de cores HSV. A matiz especifica o ângulo da cor no círculo de cores RGB, com o valor que varia entre 0° à 360°. A saturação controla a pureza da cor, é a medida de quão diferente uma cor aparece de um cinza da mesma leveza e o seu valor varia entre 0 à 1,0. A luminosidade controla o brilho da cor, uma cor com 0 de luminosidade é preta, com 0,5 é a cor mais pura possível e com 1 é branca.

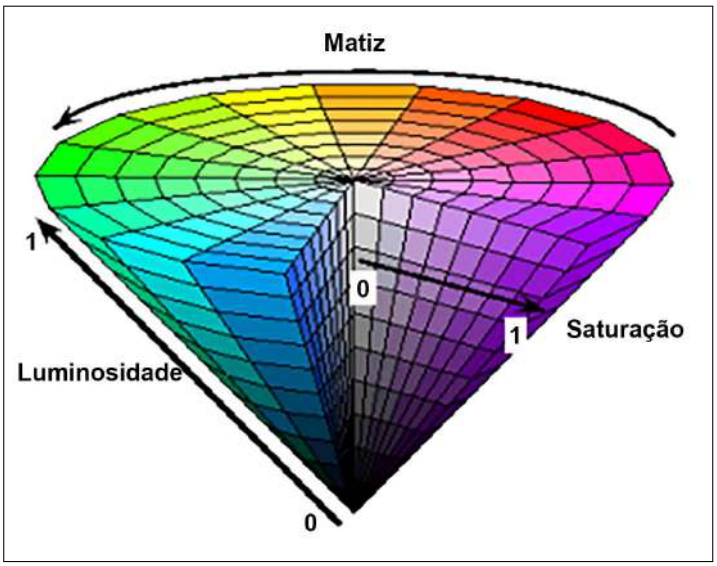
2.2 Manipulação de Imagens

As imagens carregam no seu interior uma determinada informação, proporcionando várias aplicações através dela. Ao fazer uma análise computacional de uma imagem, está na realidade em busca de informações quantitativas que representam um fenômeno a ser estudado. E para extrair mais facilmente a informação presente é possível transformar

Figura 2 – Cores primárias e secundárias da luz e pigmentos

FONTE: Adaptado de (RICHARD, 2018)

Figura 3 – Espaço de cores HSV



FONTE: Adaptado de (JATAKIA, 2017)

a imagem e enfatizar o objeto de interesse contido na imagem. A seguir são mostradas algumas transformações.

2.2.1 Transformação RGB Para Escala de Cinza

A escala de cinza contém apenas tons de cinza e sem cor, possuindo apenas um valor em cada *pixel*. Enquanto a escala RGB possui três valores em cada *pixel*. O único valor no *pixel* da escala de cinza refere-se a luminância, descrita também como brilho ou intensidade. Com uma intensidade zero tem-se o preto e com a intensidade total tem-se o branco.

Para converter uma imagem colorida em preto e branco é preciso remover todas as informações de cores, deixando apenas a luminância de cada *pixel*. Portanto, fazendo uma combinação da luminância do vermelho, verde e azul é possível gerar um único valor. Existem várias maneiras de fazer isso. Uma opção é calcular a média de todos os valores de luminância para cada *pixel*. Na Equação 2.1, mostra a relação utilizada neste trabalho.

$$RGB\ para\ Escala\ de\ Cinza: \ Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{2.1}$$

2.2.2 Transformação RGB Para HSV

O espaço de cores HSV foi projetado como uma deformação de um cubo de cores RGB. Em outras palavras, o HSV é uma coordenada diferente do sistema RGB. Representa geometricamente um cone hexagonal, como mostrado na Figura 3. No formato HSV, um pixel é codificado em três valores que estão mais descorrelacionados do que no formato RGB, facilitando o processamento das imagens em condições de luminosidade diferentes.

Abaixo segue as Equações 2.2, 2.3 e 2.4 para a transformação do RGB para o HSV utilizadas no (OPENCV, 2022b).

$$V \leftarrow max(R, G, B) \tag{2.2}$$

$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & se\ V \neq 0 \\ 0 & caso\ contrário \end{cases} \tag{2.3}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V - min(R,G,B)) & se\ V = R \\ 120 + 60(G-B)/(V - min(R,G,B)) & se\ V = G \\ 240 + 60(G-B)/(V - min(R,G,B)) & se\ V = B \end{cases} \tag{2.4}$$

2.2.3 Suavização de Imagem

Ao processar uma imagem, muitas vezes o interesse é identificar objetos representados dentro dela para que possa ser realizado análises. Um conceito importante associado à identificação de objetos em uma imagem é a das arestas: linhas que representam uma transição de um grupo de *pixels* semelhantes na imagem para outro grupo diferente. Um exemplo de aresta são os *pixels* que representam os limites de um objeto em uma imagem, onde o fundo da imagem termina e o objeto começa.

Quando uma imagem é desfocada, é feito uma transição de cor de um lado de uma borda na imagem para outro de modo suave, em vez de repentina. Reduzindo e distorcendo detalhes que podem caracterizar como ruídos no processamento das imagens. Como por exemplo, um filtro passa-baixa, que remove detalhes em altas frequências.

Para isso, utiliza-se um kernel, que consiste em uma pequena matriz que é combinada com a imagem usando o método de convolução. O kernel pode possuir diferentes tamanhos, formas e conteúdos, produzindo efeitos diferentes. Por exemplo, o kernel Gaussiano que segue a seguinte distribuição gaussiana, mostrada na Equação 2.5.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.5}$$

A largura e a altura do kernel devem ser um número ímpar, para que o *pixel* que está sendo trabalhado esteja sempre em seu centro. É calculada uma média dos valores de cor dos *pixels* ao redor dele, ponderada pelos valores no kernel. Na Equação 2.6 é mostrado como é calculado os valores do kernel ao definir a largura e altura da imagem, e o tamanho do kernel. Um kernel com um tamanho maior borrará a imagem mais do que um kernel menor.

$$K = \frac{1}{ksize \cdot largura \cdot ksize \cdot altura} \begin{bmatrix} 1 & 1 & 1 & ... & 1 & 1 \\ 1 & 1 & 1 & ... & 1 & 1 \\ ... & & & & & \\ 1 & 1 & 1 & ... & 1 & 1 \end{bmatrix} \tag{2.6}$$

2.2.4 Redimensionamento de Imagem

Redimensionar uma imagem significa alterar as dimensões dela, seja apenas largura, apenas altura ou ambas. Além disso, a proporção da imagem original pode ser preservada na imagem redimensionada. Quando uma imagem é reduzida em tamanho, qualquer informação de *pixel* desnecessária será descartada pelo método de redimensionamento escolhido. Quando uma imagem é ampliada, deve-se criar e adicionar novas informações de *pixel*, normalmente resultando em uma imagem muito suave e desfocada.

A operação de escala pode ser representada por sua matriz de transformação, Equação 2.7. No qual, s_x e s_y são os fatores de escala nos eixos X e Y.

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \tag{2.7}$$

2.3 Reconhecimento de Padrões e Classificadores

Enxergar é um processo fisiológico altamente complexo e que envolve inúmeras estruturas do nosso organismo. A visão não se resume à captação da luz refletida por objetos, mas abrange uma série de processos de identificação e interpretação dessa luz pelo cérebro. Realizar esses mesmos processos em uma máquina, exige por antecedência uma compreensão filosófica do mundo ou dos conhecimentos humanos. Por isso, o reconhecimento de imagens por uma máquina é dependente do sistema no qual ele está associado, não existindo uma solução única e abrangente para todos os problemas.

A identificação de objetos por um computador é baseado nas tecnologias de *machine learning* e *deep learning*. Os algoritmos analisam e aprendem com cada *pixel* da imagem, passando a reconhecê-las e agrupá-las por meio valores numéricos. Possibilitando a comparação de imagens semelhantes.

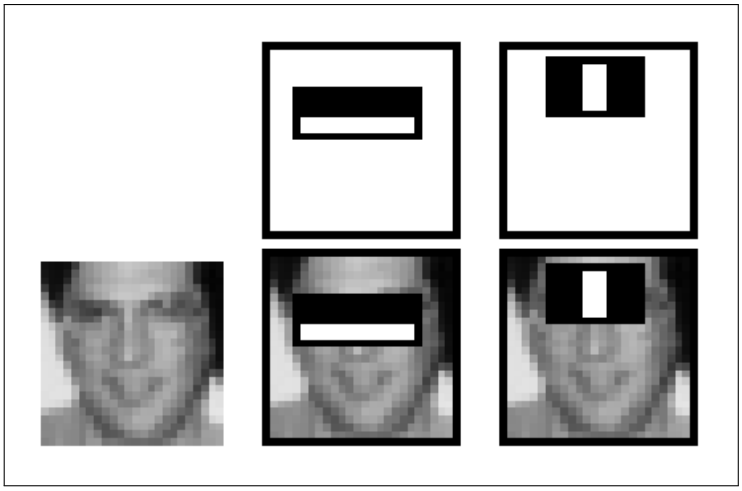
2.3.1 Haar Cascade

Entre os classificadores, muito popular na visão computacional, existe o *Haar cascade*. Ótimo detector e classificador de objetos, leve e rápido para ser executado em tempo real em suas aplicações. É um algoritmo baseado em recursos para detecção de objetos, proposto por (VIOLA, 2001). Originalmente, utilizado para detecção da face frontal, porém a sua execução abrange outros objetos pré-treinados.

Utiliza-se de recursos retangulares, semelhantes ao kernel, conforme mostrado Figura 4 abaixo. Eles são usados para detectar diferentes partes do objeto, conforme mostrado na ilustração. A aplicação do seu algoritmo é disponibilizado e de fácil acesso na biblioteca (OPENCV, 2022b).

A utilização desse método possui as etapas de treinamento de classificadores e aplicação de classificadores. A etapa de treinamento de classificadores inclui a obtenção das imagens, aplicação de processamento das imagens, categorização e treinamento dessas imagens usando o algoritmo em cascata. E a etapa de aplicação do classificador contém a coleta de imagens de vídeo e detecção do objeto treinado. Como é mostrado no trabalho (HAK, 2019), o qual utiliza o método para detecção de equipamentos de segurança nos trabalhadores industriais.

Figura 4 – Exemplos dos recursos retangulares aplicados na detecção de faces



FONTE: Adaptado de (VIOLA, 2001)

A principal característica do *Haar cascade* é fazer com que seus estágios iniciais descartem uma grande quantidade de regiões que não contém o objeto desejado, e deixar o trabalho mais simples para os estágios mais avançados, aumentado sua eficiência, rapidez e precisão. Descrito por (VIOLA, 2001), como uma árvore degenerativa de decisão, contém um encadeamento de classificadores do mais genérico ao mais específico, segundo o qual os primeiros níveis da cascata são menos precisos, apesar de conseguir classificar uma grande quantidade de amostras com uma pequena quantidade de características.

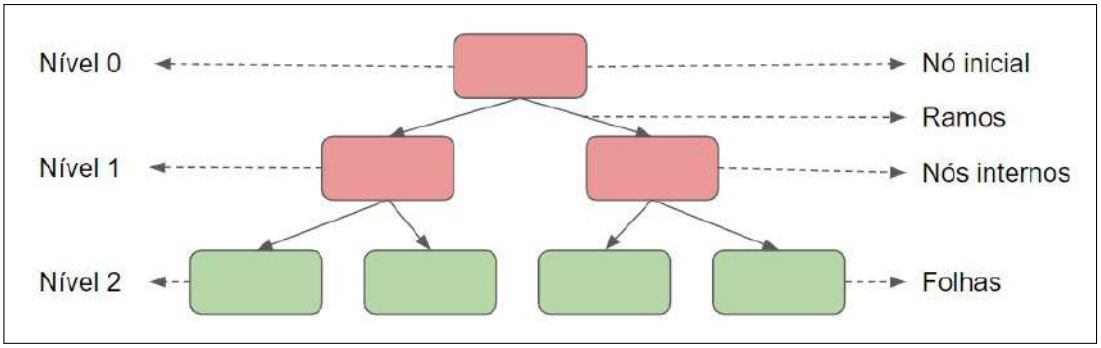
2.3.2 Árvore de Decisão

As árvores de decisão são um meio eficiente para produzir classificadores a partir de uma base de dados. São diagramas que representam e avaliam problemas que envolvem decisões sequenciais, sendo largamente utilizadas devido à sua eficiência em relação ao tempo de processamento em comparação com outras técnicas, (TAYLOR, 1994). E fornecem um meio intuitivo de analisar os resultados obtidos, apresentando uma forma de representação simbólica, o que pode facilitar a análise do problema em questão, (STONE, 1984).

Um classificador hierárquico é um algoritmo para rotular um padrão desconhecido usando uma sequência de decisões. Na Figura 5, mostra uma árvore de decisão composta de um nó inicial, um conjunto de nós internos e nós terminais chamados de folhas. O nó inicial e os nós internos, chamados coletivamente de nós não terminais, são vinculados aos estágios de decisão. Os nós terminais representam a classificação final, (MATHER, 2009).

Entre as principais vantagens do uso das árvores de decisão, se destaca a fácil interpretação dos seus resultados, pois a classificação é obtida de forma explícita, simplificando a sua interpretação. E seus resultados são fornecidos rapidamente, geralmente, devido à sua eficiência computacional, (DEFRIES, 1996).

Figura 5 – Classificador de árvore de decisão hierárquica simples



FONTE: Autoria Própria

2.4 Robôs Autônomos

Para (BEKEY, 2001), os robôs autônomos podem ser definidos como máquinas que sentem, pensam e agem. Distinguindo de softwares, pois eles estão incorporados no mundo real, podendo ser tocados, ter dimensões físicas e ter interações com outros objetos. Estão sujeitos às leis da física, possuem massa, partes móveis que geram atrito e, portanto, calor, estão sujeitos as peças quebrarem. Contêm também computadores, ou seja, possuem hardwares e softwares, sendo assim, vulneráveis as suas limitações e falhas. Afinal, o local de atuação desses robôs muitas vezes sofrem alterações nas características e eles precisam se adaptar e tomar decisões próprias com antecedência.

Os robôs autônomos atuam em diversas áreas: industriais, alimentos, eletrônica e entre outras. No Brasil, por exemplo, foi desenvolvido um sistema para monitorar a ocorrência de vazamento de óleo em plataformas marítimas de extração de petróleo. Constituído por um barco, responsável por levar um drone nas áreas predeterminadas, e sem a interferência humana, o drone decola automaticamente para suas sessões de monitoramento, e retorna para à embarcação quando necessário recarregar a bateria, (CHEVRAND, 2022).

Para a realização de tarefas industriais, frequentemente é utilizado os dispositivos autônomos AGV (Veículo Guiado Automatizado) e AMR (Robô Móvel Autônomo) (LEGOWIK, 2016). Realizando manuseio de materiais, atividades de pesquisa, trabalho colaborativo com humanos e atividades cooperativas entre os veículos autônomos. Um AGV tem inteligência interna mínima e só pode obedecer a instruções de programação simples. Para navegar, ele precisa ser guiado por fios, tarjas magnéticas ou sensores. O AMR navega por meio de mapas que seu *software* constrói no local ou por meio de desenhos de instalações pré-carregados. Usa dados de câmeras, sensores embutidos e *scanners* a laser, bem como um *software* sofisticado que permite detectar seus arredores e escolher a rota mais eficiente para o alvo.

2.5 Ferramentas Computacionais Utilizadas

O uso de programas e algoritmos computacionais são essenciais para a execução de um carro autônomo. Como por exemplo:

- **Linguagem de Programação** - É um método padronizado para comunicar instruções para um computador, um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.
- **Biblioteca** - É uma coleção de recursos utilizados no desenvolvimento de software, contém código e dados auxiliares.

A seguir, é apresentado a linguagem de programação utilizada no trabalho e as bibliotecas.

2.5.1 Linguagem de Programação

A linguagem de programação utilizada foi Python, uma linguagem de alto nível, com uma sintaxe voltada para o entendimento humano, sendo mais intuitiva, amigável e fácil de aprender. Utilizada em diferentes tipos de aplicações, como desenvolvimento *WEB*, *machine learning*, desenvolvimento de jogos e entre outros. Na elaboração deste trabalho, foi utilizada a versão 3.9.7, lançada em 30 de Agosto de 2021.

Sendo uma das linguagens mais populares, ela possui uma grande quantidade de ferramentas para diversas aplicações. Com isso, neste trabalho foi aplicado algumas bibliotecas, as quais são explicadas em detalhes a seguir.

2.5.1.1 Manipulações de Matrizes

A biblioteca *Numeric Python (Numpy)*, é um projeto de código aberto e gratuito, amplamente utilizada no tratamento de dados, possuindo uma vasta quantidade de operações matemáticas e suporte para cálculos com matrizes, (NUMPY, 2022).

Aproveitando o auxílio sobre operações de matrizes, este trabalho utilizou esta biblioteca no decorrer de toda a execução.

2.5.1.2 Manipulação de Arquivos

Encontrar os arquivos de imagens no sistema operacional e utilizá-los nos algoritmos, são funções que a biblioteca *glob* possui. Com o seu método, também chamado de *glob*, retorna uma lista de caminhos de arquivos, em ordem arbitrária, por meio de uma condição estabelecida. Podendo especificar quais formatos de arquivos desejam ser extraídos.

2.5.2 Visão Computacional

Com a visão computacional é possível obter informações de imagens, sejam elas astronômicas, microscópicas ou em tamanho real. Com a utilização de algoritmos na programação, é capaz de descrever e analisar objetos de uma imagem digital.

A biblioteca *Open Source Computer Vision Library (OpenCV)* fornece uma infraestrutura para aplicações de visão computacional. Possui algoritmos que podem identificar objetos, detectar e reconhecer rostos, rastrear objetos em movimento, entre outros, (OPENCV, 2022a). A seguir é listado as principais funções utilizadas da biblioteca.

- ***CascadeClassifier*** - Método de treinamento do *Haar cascade*, para sua execução é necessário carregar o arquivo XML que irá determinar o seu funcionamento.
- ***matchTemplate*** - Aplica o *template match*, deslizando a imagem modelo sobre a imagem original, comparando-os a fim de encontrar uma correspondência.

2.5.3 Aprendizado de Máquina

A fim de melhorar os resultados por meio de dados de treinamento e com uma aprendizagem supervisionada, o *scikit-learn* contribui para isso.

O *scikit-learn* é uma biblioteca de código aberto, utilizada para aprendizado de máquina por meio da linguagem de programação *Python*, (PEDREGOSA et al., 2011). As ferramentas podem ajudar em algoritmos de classificação, regressão, agrupamento, entre outros relacionados ao aprendizado de máquina. A seguir é listado as principais funções usadas desta biblioteca.

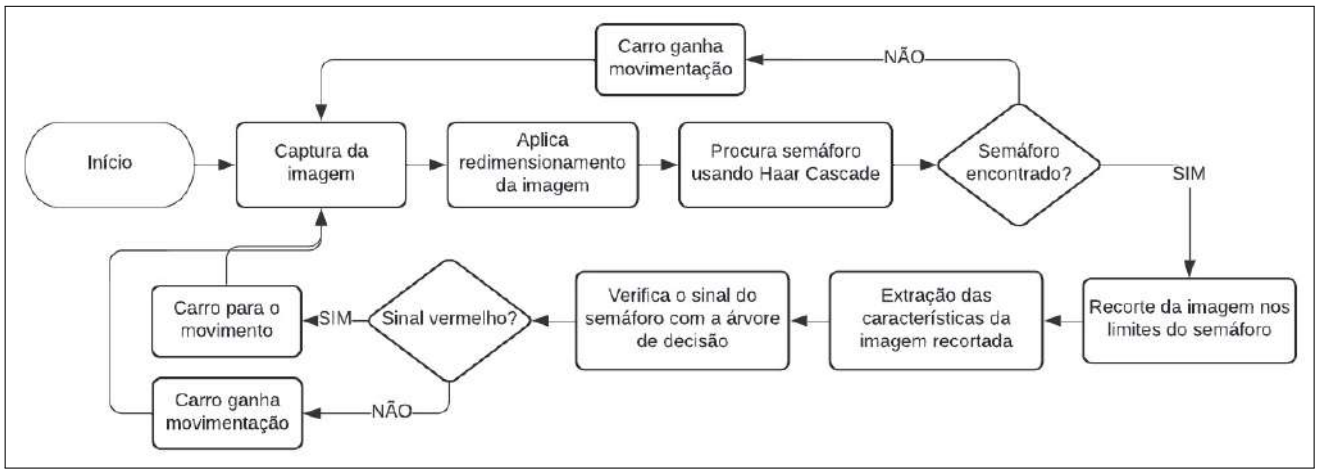
- ***confusion_matrix*** - Calcula a matriz de confusão para avaliar a precisão de uma classificação.
- ***DecisionTreeClassifier*** - Cria o objeto de classificação por meio do método da árvore de decisão.
- ***fit*** - Treina e retorna a árvore de decisão através dos dados informados.
- ***predict*** - Prever o resultado por meio da árvore de decisão já treinada.
- ***train_test_split*** - Divide os *array* em subconjuntos aleatórios de treinamento e teste.

3 Materiais e Métodos

Este trabalho deu continuidade ao trabalho Controle de Trajetória em Robô Autônomo Utilizando Processamento de Imagens e Redes Neurais Artificiais, (JÚNIOR, 2022). O qual consistia na automação dos movimentos do carro por meio da análise de imagem, adquirida da câmera, utilizando redes neurais artificiais de múltiplas camadas. Junto com essa aplicação, foi acrescentado a análise de semáforo durante o trajeto do carro.

Para manter a funcionalidade do trabalho já desenvolvido, este o reproduziu mantendo as mesmas configurações, componentes eletrônicos e algoritmos. As configurações mantidas equivalentes nos dois trabalhos são especificados no decorrer deste capítulo. Os procedimentos para desenvolver as etapas de processamento de imagem e reconhecimento do semáforo são ilustrados no fluxograma da Figura 6.

Figura 6 – Etapas para movimentação do carro de acordo com o sinal do semáforo



FONTE: Autoria Própria

3.1 Montagem dos Protótipos

3.1.1 Protótipo do Semáforo

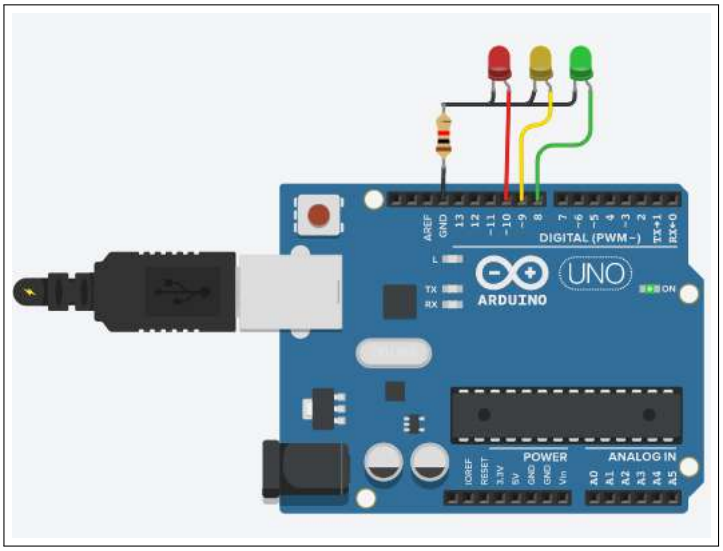
Para simular um semáforo, proporcional ao tamanho do carro, foi construído um próprio para o trabalho. Este consiste na utilização de três LEDs, nas cores vermelha, amarela e verde, com um revestimento na cor preta, como mostra a Figura 7. Para a armação do semáforo foi utilizado papelão e a base de madeira MDF (fibras de média densidade).

O controle dos LEDs foi através de um microcontrolador, Arduino UNO. Responsável por acionar cada LED no momento exato, utilizando a função *digitalWrite()* para ligar e desligar um LED, e a função *delay()* para a espera na troca de cada sinal. Na Figura 8, mostra o circuito aplicado para o funcionamento do semáforo.

Figura 7 – Primeiro protótipo do semáforo

FONTE: Autoria Própria

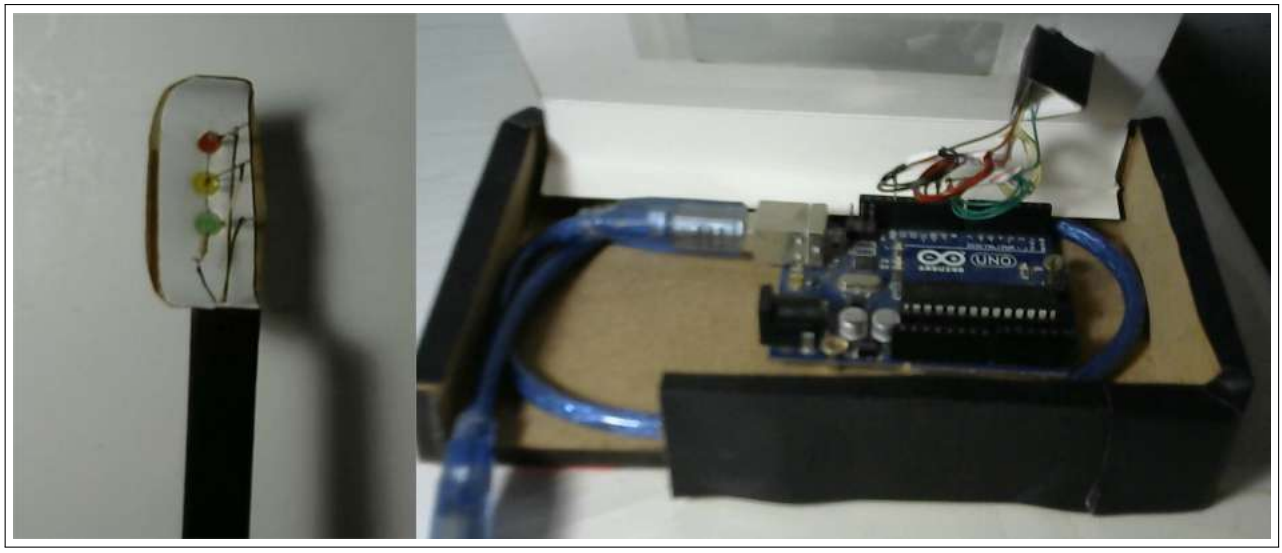
Figura 8 – Circuito do protótipo do semáforo



FONTE: Autoria Própria

Após alguns testes utilizando o semáforo, foi necessário aplicar uma melhoria. Devido a sua base ser grande em relação a pista, ocupava muito espaço e entrava em conflito com o carro. Com isso, a parte dos LEDs foi prolongada e a base foi posicionada fora da pista. Dando maior espaço para o movimento do carro. A Figura 10 mostra como como ficou o protótipo com a melhoria.

Figura 9 – Conexões do semáforo com o Arduino UNO e base de MDF



FONTE: Autoria Própria

Figura 10 – Protótipo final do semáforo com prolongamento

FONTE: Autoria Própria

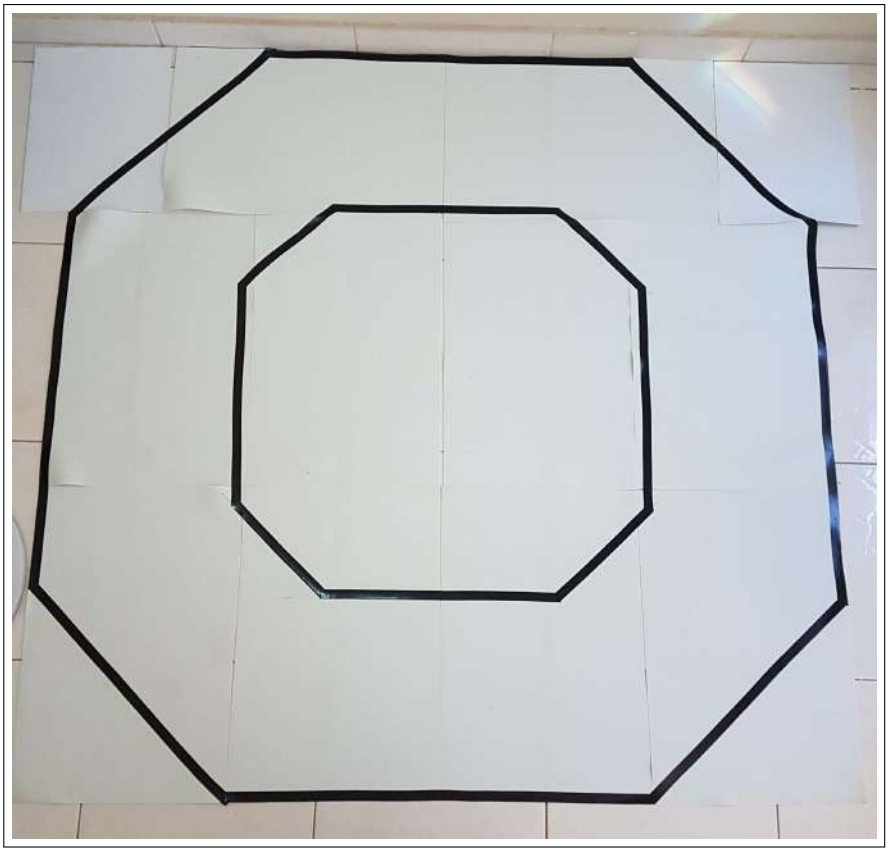
3.1.2 Protótipo da Pista

Foi construído uma pista para executar os testes com todos os componentes juntos. Feito com EVA de cor branca para oferecer maior aderência as rodas do carro e demarcada com fita isolante preta para indicar os limites da pista. Na Figura 11, mostra o resultado da construção e na Figura 12, as dimensões utilizadas, proporcional ao tamanho do protótipo do carro e do semáforo.

3.1.3 Protótipo do Carro

Para simular um carro foi utilizado o Carrinho Arduino 4WD, que é um kit chassi feito de uma estrutura de MDF com espaços adequados para a instalação dos componentes eletrônicos, listados na Tabela 2. Possuindo quatro motores CC (corrente contínua)

Figura 11 – Pista construída para os testes



FONTE: Autoria Própria

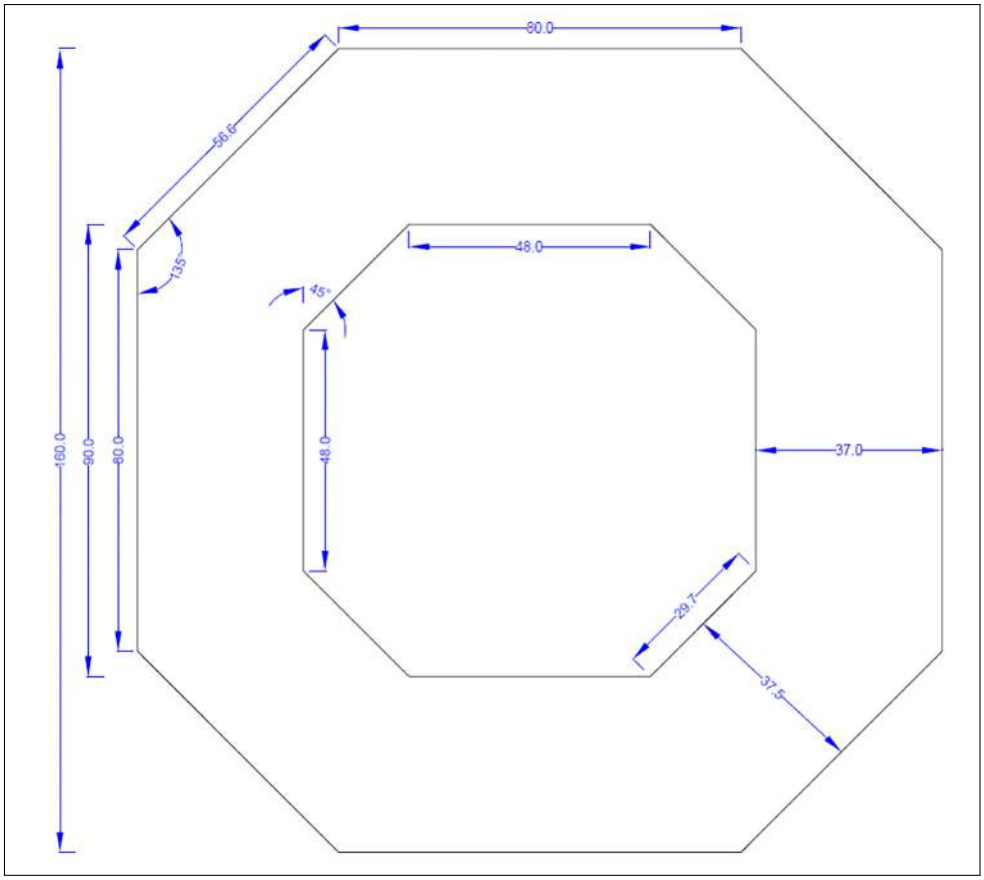
independentes, com uma velocidade máxima de 200 RPM. Controlados por uma ponte H e ligados diretamente no nível lógico alto, não sendo necessário a aplicação de um controle PWM para regulagem de velocidade.

Na dianteira do chassi contém um micro servo motor, para controlar a direção do carro. Conectado as duas rodas dianteiras, efetuando cinco direções diferentes, sendo elas: em frente (0^{o}), mínimo à direita (45^{o}), mínimo à esquerda (-45^{o}), máximo à direita (90^{o}) e máximo à esquerda (-90^{o}). Na parte traseira do carro, foi colocado a *raspberry*, uma *powerbank* para alimentar a *raspberry* e a câmera.

Quantidade	Componente
4	Motor CC
4	Roda de borracha
1	Ponte H
1	Servo Motor
1	*Raspberry* Pi Modelo 3B
1	Câmera Logitech C270 V-U0018
2	*Powerbank*

Tabela 2 – Componentes utilizados no protótipo do carro

Figura 12 – Dimensões da pista construída em centímetros



FONTE: (JÚNIOR, 2022)

3.1.3.1 Suportes da Câmera, *Raspberry* e *Powerbank*

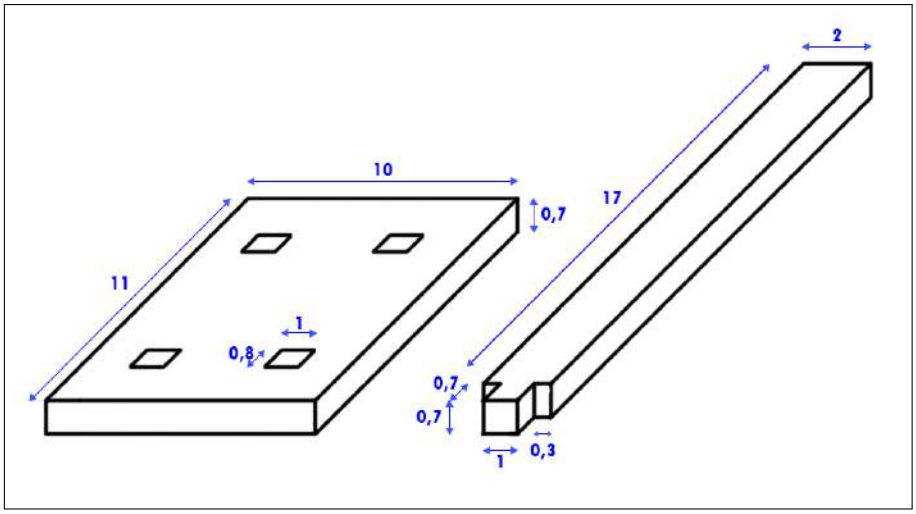
O posicionamento fixo da câmera foi essencial para manter a constância dos testes e evitar alterações indesejadas. Por isso, foi criado um suporte para segurar a câmera a 22 centímetros em relação ao chão. O material utilizado para criá-lo foi madeira MDF, mantendo o padrão do kit chassi com eixo móvel, e foi inserido no kit de forma que não precisasse fazer furos ou deformações, aproveitando as fendas já existentes. As dimensões da sua estrutura podem ser observadas na Figura 13 e seu resultado na Figura 15.

Da mesma forma, foi criado um suporte para elevar a bateria portátil, *powerbank*, e a *raspberry*. Com o objetivo de distanciá-los dos pneus traseiros, pois sem a presença do suporte, a *powerbank* encosta nos pneus, dificultando a rotação. Também foi feito de madeira MDF, aproveitando as fendas de encaixe do kit chassi. O desenho com suas dimensões se encontram na Figura 14 e seu resultado na Figura 15.

3.2 Aquisição de Imagens

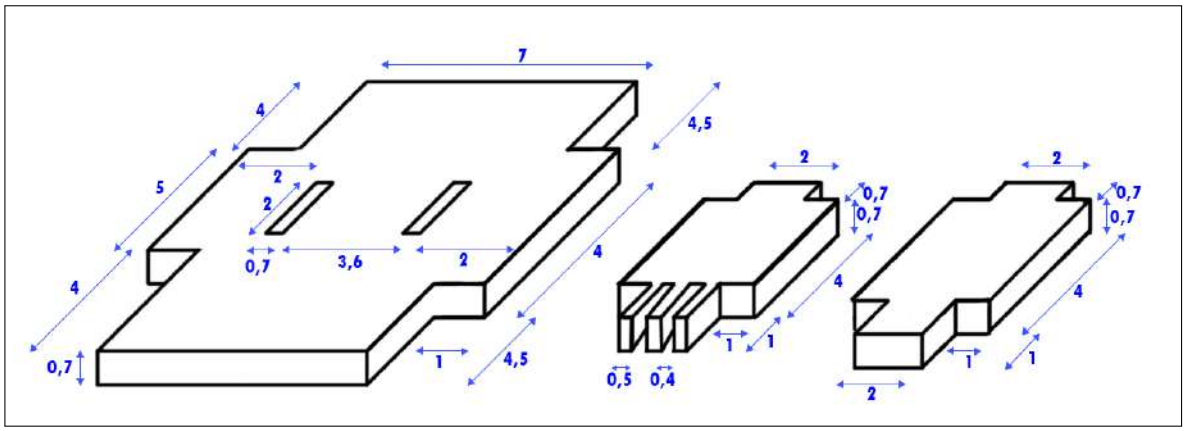
Em muitas das etapas deste projeto, foi preciso a utilização de várias imagens, sendo algumas delas adquiridas a partir de outro trabalho (FRIDMAN, 2019), e outras por meio da câmera Logitech C270 V-U0018. Os *frames* obtidos por meio dessa câmera

Figura 13 – Desenho do suporte para a câmera em centímetros



FONTE: Autoria Própria

Figura 14 – Desenho do suporte para a *raspberry* e *powerbank* em centímetros



FONTE: Autoria Própria

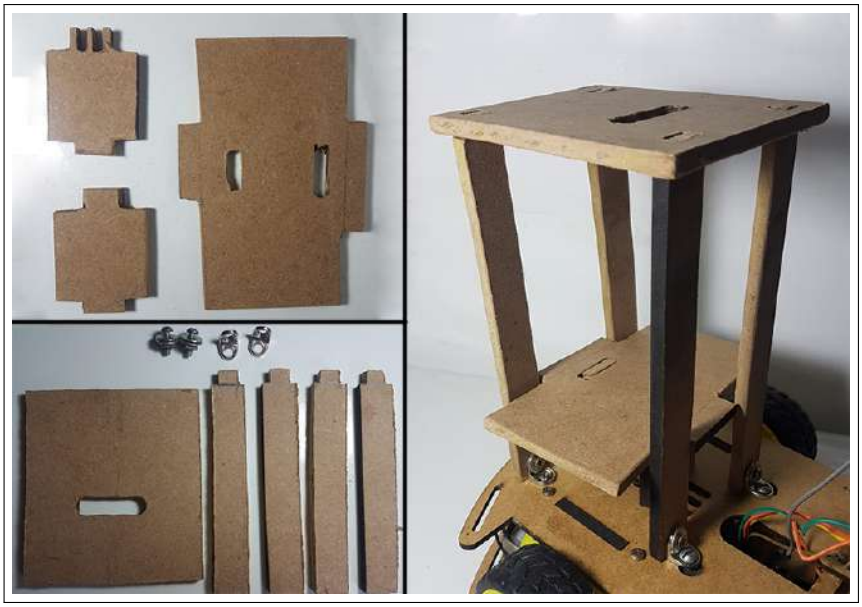
possuem uma resolução de 720p. Foi utilizada nos processos descritos a seguir, e serviu como visão para o protótipo do carro. Sendo assim, foi esperado que as imagens fornecessem informações suficientes para os processamentos adequados e que não fossem custosos para a memória do sistema embarcado, *Raspberry Pi*.

As imagens adquiridas por meio do trabalho (FRIDMAN, 2019), foram utilizadas para o treinamento da árvore de decisão. Estas imagens possuem fotografias de vários semáforos, no total são 1480 imagens, divididas entre as sinalizações, sendo 904 vermelhas, 44 amarelas e 536 verdes, possuindo tamanhos diferentes. Na Figura 16, é mostrado algumas dessas imagens.

Visto a necessidade de implementar imagens do protótipo do semáforo na árvore de decisão, para melhorar a apuração dos sinais, foi coletado 30 novas imagens, sendo 10 para cada sinalização. Na Figura 17, é apresentado alguns exemplos.

Para o *template match* foi necessário a utilização de uma imagem modelo do protó-

Figura 15 – Suportes da câmera e da *raspberry* com a *powerbank*



FONTE: Autoria Própria

Figura 16 – Exemplos de imagens do banco de dados do tutorial de *deep learning* do MIT

FONTE: (FRIDMAN, 2019)

Figura 17 – Imagens do protótipo do semáforo para a aplicação da árvore de decisão

FONTE: Autoria Própria

tipo do semáforo para a aplicação do método. Esta imagem foi obtida a partir da câmera Logitech C270 V-U0018, e recortada nos limites do semáforo. Com o formato justo nestes limites, possibilitou o melhor encontro dos seus semelhantes nas outras imagens. Estas imagens originais foram colocadas para o teste do método, no total foram 10 imagens contendo o semáforo, com variação do seu tamanho, posição e luminosidade. Na Figura 18, é mostrado a imagem modelo deste método.

Figura 18 – Imagem modelo para o uso do método *template match*

FONTE: Autoria Própria

Para o método *Haar cascade*, foi preciso obter imagens para o seu treinamento. Sendo elas separadas entre imagens negativas e positivas. As imagens negativas são aquelas as quais não possuem o semáforo, no máximo foram utilizadas 1600 imagens. Enquanto as positivas, são as que possuem o semáforo, contendo diferentes representações do objeto, como na perspectiva, condição da iluminação e tamanho, no máximo foram 210 imagens.

Figura 19 – Imagens utilizadas para o treinamento do *Haar cascade*

(a) Imagens positivas **(b)** Imagens negativas

FONTE: Autoria Própria

3.3 *Template Match*

A primeira tentativa para encontrar o semáforo foi utilizando o *template match* do *OpenCV*. Este método utilizava uma imagem modelo do semáforo para fazer a comparação com as imagens de entrada. A comparação era feita deslizando a imagem modelo

sobre imagem de entrada por uma convolução 2D. Para evitar algum erro em relação a diferença de tamanhos entre a imagem modelo e o semáforo na imagem original, foi aplicado um *multiscaling*, ou seja, uma variação de escala da imagem original, diminuindo e aumentando seu tamanho.

Além disso, a comparação era feita mais de uma vez na mesma imagem, variando a sua escala. E de acordo com um limiar estabelecido, era selecionado o melhor resultado. O limiar é a precisão que se deseja ter da imagem modelo sobre a imagem original. Por exemplo, caso o método fosse aplicado para reconhecimento facial e desejasse detectar os olhos de uma pessoa, um limiar de valor alto (limiar $> 80\%$) não seria tão eficaz, devido a grande variação de olhos existente. Porém, com um limiar menor, geraria maiores correspondências, deixando a aplicação mais genérica. Se o interesse é obter modelos quase idênticas, deve-se então manter o limiar alto.

Para aplicar o método com maior êxito, foi necessário fazer ajustes nos principais parâmetros do algoritmo. Os parâmetros testados a fim obter o melhor resultado foram:

- ***Threshold* (Limiar)** - Evitando ocorrências indesejadas e servindo como filtro, foi escolhido um limiar de 0,75. Este valor foi escolhido observando o quão próximo a imagem modelo deveria casar com o semáforo da imagem original. Caso esta imagem não possuísse nenhum semáforo, este parâmetro foi o responsável por eliminar as circunstâncias dispensáveis.
- **Variação de Escala** - Para que o método percorresse a imagem original, aproveitando possíveis tamanhos diferentes de semáforos, foi verificado um série de escalas diferentes da imagem original. Começando com uma redução de 50% até um acréscimo de 500% da sua escala.
- **Passos** - É a quantidade de intervalos entre as variações de escala. Por exemplo, caso haja uma variação de 50% à 150%, com 5 passos, então, consta intervalos a cada 25% do início ao fim.

Foi observado cada configuração nas 30 imagens de teste, verificando a quantidade de imagens as quais o método conseguiu encontrar o semáforo e fazer o melhor recorte, como mostrado na Figura 20. O retângulo branco representa o recorte feito no semáforo, e o texto no canto superior mostra a melhor escala escolhida pelo algoritmo. Os resultados dos testes são apresentados no Capítulo 4.1.

O método de correspondência utilizado foi o de correlação cruzada normalizado da biblioteca *OpenCV*, chamado de *TM_CCORR_NORMED*.

Quando este método foi colocado junto com a movimentação do carro, a identificação do semáforo ficou instável, impossibilitando a sua aplicação. Logo, foi preciso trocar a forma de identificá-lo, e o método que o substituiu foi o *Haar cascade*, explicado a seguir.

Figura 20 – Identificação e recorte do semáforo com *template match*

FONTE: Autoria Própria

3.4 *Haar Cascade*

Os recortes dos semáforos obtidos pelo *template match* nas imagens, em alguns casos, não foram adequados para a aplicação da árvore de decisão. Logo, foi colocado em prática outro método para a identificação do semáforo, o *Haar cascade*. Esperando que sua troca com o *template match*, aumentasse a quantidade de semáforos encontrados e com melhorias no recorte. E sua escolha foi devido a facilidade de acesso e usabilidade com a biblioteca (OPENCV, 2022b).

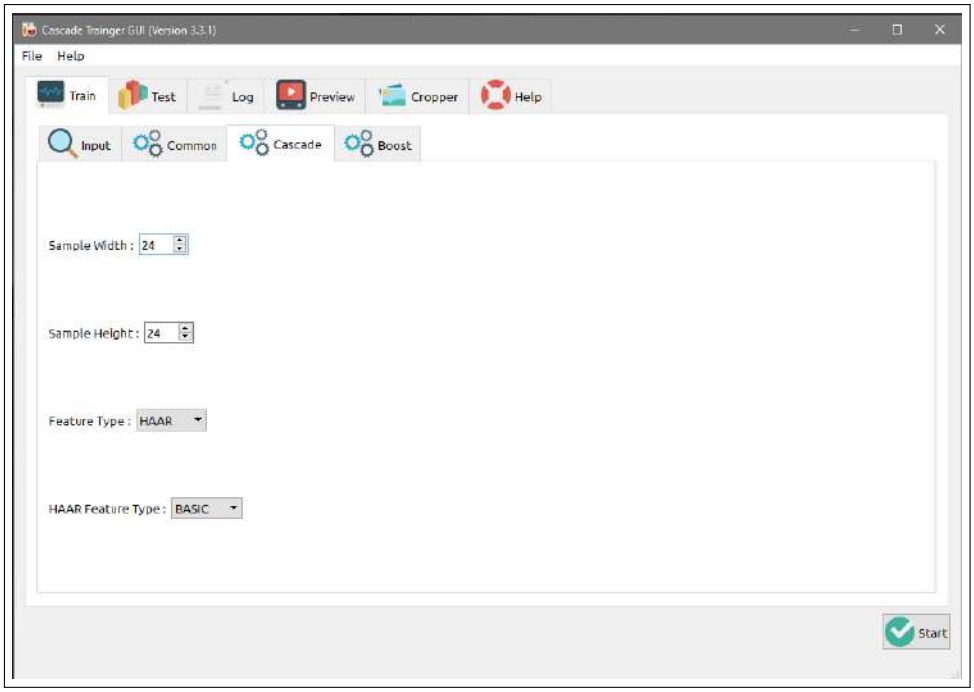
Primeiramente, foi preciso fazer o treino do classificador para o reconhecimento do semáforo. Para facilitar este processo, foi utilizado o programa *Cascade Trainer GUI*, versão 3.3.1, o qual possui uma interface gráfica para definir os parâmetros essenciais e aplicar as ferramentas do *OpenCV*. Na Figura 21, mostra a interface deste programa.

O *Cascade Trainer* inicia fornecendo o local da pasta onde se encontra os diretórios das imagens positivas e imagens negativas, na aba "*Input*". E clicando no botão "*Start*", começa o treinamento. Mas antes, é possível alterar algumas configurações, as quais influenciam no resultado final. Foi colocado o número de 15 estágios, 5 *threads* e tamanho da largura e altura da amostra de 30x40. Os resultados com alterações nas imagens de treino são encontrados no Capítulo 4.2.

Ao finalizar o treino, é gerado um arquivo *XML* (*cascade.xml*), o qual contém o *script* necessário para a aplicação do *Haar cascade*. Entre os arquivos *XMLs* produzidos, o maior deles possuía 36 KB, adequado para implementá-lo no sistema embarcado, sem prejudicar sua performance.

Visto que a aplicação do *Haar cascade* tinha como função apenas encontrar a forma do semáforo na imagem e fazer seu recorte, não era necessário encontrar a cor do sinal, neste momento. Então, para simplificar a análise do método foi transformado as imagens

Figura 21 – Interface do programa *Cascade Trainer GUI*



FONTE: Autoria Própria

de treino para a escala de cinza. Sendo analisado apenas um valor, a luminância em cada *pixel*, ao invés de 3 valores quando colorido.

Além disso, para diminuir ainda mais a complexidade, foi aplicado a suavização de imagem nas imagens transformadas na escala cinza, com o kernel de 5x5. Diminuindo a transição do valor da luminância nas arestas dos objetos das imagens. Na prática, aumentando a possibilidade de encontrar a forma do semáforo nas imagens.

Por fim, foi observado que a identificação do semáforo acontecia, frequentemente, apenas em uma determinada distância. E com o carro em movimento, constantemente a captura da imagem da câmera não encontrava o semáforo na distância requerida. Para aumentar a quantidade de encontros com distâncias diferentes, foi alterado aleatoriamente a escala das imagens positivas no treino do *Haar cascade*, variando entre 50% à 100% do tamanho original.

No Apêndice B.1, mostra o código do algoritmo para a identificação do semáforo com o *Haar cascade*.

3.5 Árvore de Decisão

Com a identificação do semáforo e seu recorte de 30x40 da imagem capturada, é passado para a próxima etapa, a de identificação do sinal. Sendo possível identificar o sinal vermelho, verde ou amarelo por meio do método classificador, árvore de decisão.

Este método foi treinado com 100 imagens de cada representação dos sinais, no

total foram 300 imagens. As imagens do treino foram adquiridas pela própria câmera do trabalho, mantendo a dimensão de 30x40 em cada imagem. E para a aplicação do treino foram extraídas as seguintes características:

- **Médias das cores RGB**
- **Médias dos valores HSV**
- **Posição de maior brilho**

Para cada imagem foi aplicada uma máscara de acordo com o nível de luminosidade ou de saturação, escondendo as partes de menos interesse. Primeiramente, era analisado o nível de luminosidade de cada *pixel*, selecionando aqueles com os valores entre 200 e 255. Caso não existisse nenhum *pixel* de acordo com esse limiar, então era analisado o nível de saturação de cada *pixel*, selecionando aqueles com valores entre 120 e 255. Na Figura 22, é mostrado a imagem após a aplicação da máscara.

Figura 22 – Imagem original e imagem com a máscara aplicada

FONTE: Autoria Própria

Para encontrar as médias das cores RGB, foi feito a soma individual das cores de cada *pixel* da imagem com a máscara e dividido por 255, como mostra a Equação 3.1. Obtendo assim, um valor único para a cor vermelha, verde e azul.

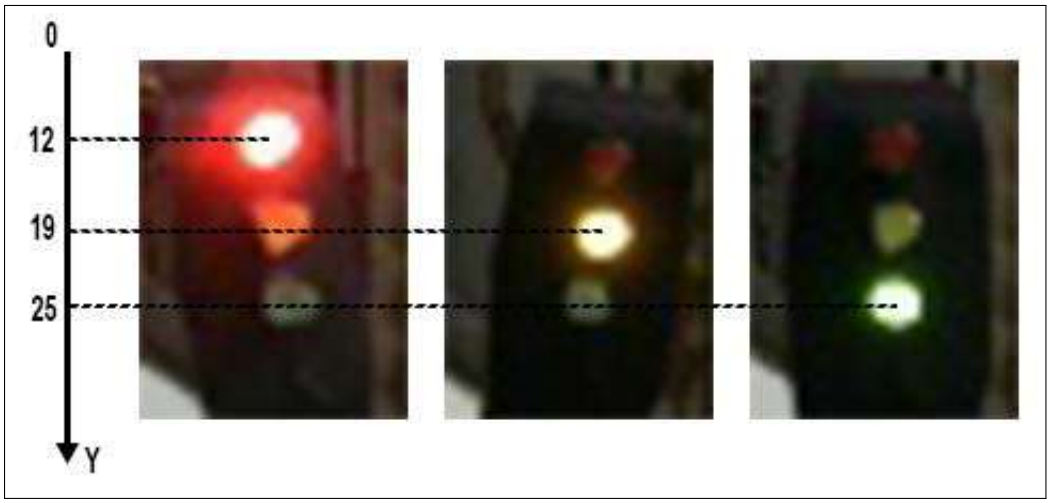
$$Médias\ das\ cores\ RGB = \begin{bmatrix} soma\_valores\_vermelho \div 255, \\ soma\_valores\_verde \div 255, \\ soma\_valores\_azul \div 255 \end{bmatrix} \tag{3.1}$$

Da mesma forma, para encontrar as médias das cores HSV, foi feito a soma individual dos valores de cada *pixel* da imagem com a máscara e dividido por 255, como mostra a Equação 3.2. Obtendo um valor único para a matiz, a saturação e a luminosidade.

$$Médias\ dos\ valore\ HSV = \begin{bmatrix} soma\_valores\_matiz \div 255, \\ soma\_valores\_saturacão \div 255, \\ soma\_valores\_luminosidade \div 255 \end{bmatrix} \tag{3.2}$$

A posição de maior brilho foi adquirido comparando e escolhendo a linha da matriz da imagem que possuía a maior soma dos valores de luminosidade ou saturação. Essa posição é em relação ao eixo Y, representando a localização do sinal ligado, sabendo que o arranjo dos sinais são iguais, vermelho, amarelo e verde, respectivamente. Na Figura 23, é mostrado um exemplo para cada sinal, indicando a posição do ponto de maior brilho e o seu valor no eixo Y.

Figura 23 – Posição de maior brilho nas imagens do semáforo



FONTE: Autoria Própria

No Apêndice B.2, mostra o código do algoritmo para a captura das características da imagem.

4 RESULTADOS

4.1 Parâmetros do *Template Match*

Com destino de escolher os melhores parâmetros deste método, foram capturadas 30 imagens estáticas do semáforo em posições, distâncias e iluminações diferentes. E observado a quantidade de imagens as quais o método conseguiu encontrar o semáforo.

A Tabela 3, apresenta os resultados dos testes para diferentes margens de escalas e passos. A apuração é mostrada na última coluna, informando a quantidade de imagens em que foi encontrado algum semáforo. No total foram testadas 30 imagens, as quais todas possuem um semáforo, com dimensão, iluminação e posição diferentes.

Variação de escala	Passos	Quantidade de imagens encontradas
Sem variação	Sem passos	8 / 30
0,5 - 1,5	5	19 / 30
0,5 - 1,5	10	19 / 30
0,5 - 1,5	20	21 / 30
0,5 - 2,0	20	24 / 30
0,5 - 5,0	50	30 / 30

Tabela 3 – Resultados do *template match* para a identificação e recorte do semáforo da imagem capturada pela câmera

Observando os resultados da Tabela 3, a última configuração foi a melhor opção. Porém, mesmo encontrando os 30 semáforos, os recortes dos semáforos, necessários para o próximo passo, não foram adequados. Na Figura 24, pode-se notar a ausência dos retângulos brancos ao redor do semáforo, os quais representam o recorte feito pelo programa.

Figura 24 – *Template match* com variação de escala de 50% à 500%, com 50 passos, encontrando 30 semáforos em 30 imagens

FONTE: Autoria Própria

Na Figura 25, mostra os resultados para o método sem variação de escala e passos. Os recortes podem ser vistos em cada imagem e em sua maioria conseguem deixar o semáforo centralizado no retângulo, o que era desejado. Mas só foi possível identificar o semáforo em apenas 8 imagens.

Figura 25 – *Template match* sem variação de escala, encontrando 8 semáforos em 30 imagens

FONTE: Autoria Própria

Com uma quantidade maior de semáforos encontrados e com alguns recortes centralizando o objeto, a opção de 20 passos com a variação de escala de 50% à 150% foi a melhor, mostrado na Figura 26. Contudo, na aplicação com o carro em movimento, o método não era viável, sendo substituído pelo *Haar cascade*.

Figura 26 – *Template match* com variação de escala de 50% à 150%, com 20 passos, encontrando 21 semáforos em 30 imagens

FONTE: Autoria Própria

4.2 *Haar Cascade*

Foi adquirido 100 imagens do protótipo do semáforo para testar configurações diferentes do *Haar cascade*, a fim de obter o melhor resultado.

Na Tabela 4, mostra o resultado para cada teste. Diferenciando-os entre as seguintes características:

- **Positivas** - Quantidade de imagens positivas utilizadas no treino.
- **Negativas** - Quantidade de imagens negativas utilizadas no treino.
- ***Gray*** - Aplicação da escala de cinza nas imagens de treino.
- ***Blur*** - Aplicação de suavização nas imagens de treino.
- **Escalável** - Aplicação de variação de escala nas imagens de treino.

Positivas	Negativas	Gray	Blur	Escalável	Acurácia
100	500	não	não	não	28%
210	1600	não	não	não	19%
200	1200	sim	sim	não	42%
200	1200	sim	sim	sim	74%

Tabela 4 – Resultados do *Haar cascade* para a identificação do semáforo com 100 imagens de teste

Obtendo no final a acurácia do teste. Observou-se que uma quantidade maior de imagens positivas e negativas não significavam maior acurácia do método. Aplicando a escala de cinza e suavização, melhorou a identificação do semáforo, visto que não era necessário as cores RGB nesta etapa. E aplicando a suavização dos contornos, diminuiu a complexidade da imagem.

E a variação de escala nas imagens de treino serviu para que o algoritmo fosse capaz de identificar o semáforo em diferentes distâncias. Uma vez que, sem isso, o algoritmo estava limitado a identificar apenas em uma determinada distância.

4.3 Árvore de Decisão

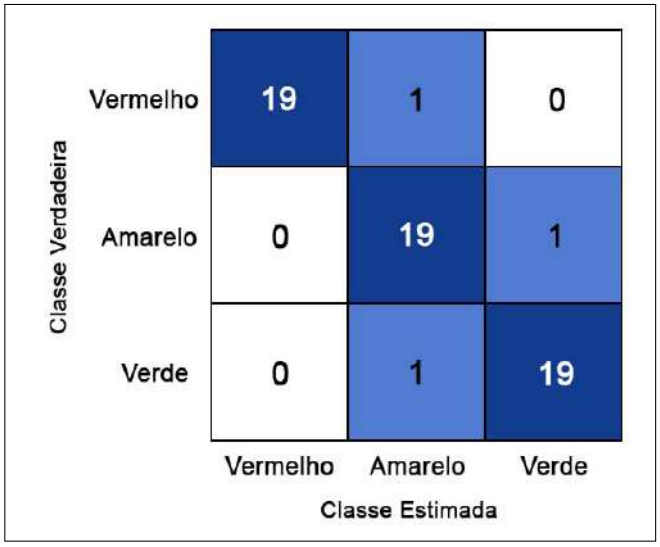
Para a indicação do sinal apresentado no semáforo, foi utilizado o algorítimo de classificação, árvore de decisão. Foram aplicadas 300 imagens do semáforo para a sua execução. Sendo 100 imagens para cada sinal do semáforo: vermelho, amarelo e verde.

Essas imagens serviram para o treino e teste da árvore de decisão. Utilizando o método *train_test_split()* do *sklearn* para dividir as imagens em subconjuntos aleatórios de treinamento e teste. Para este método é definido a porcentagem da quantidade de imagens para cada caso, ficando 80% para treinamento e 20% para teste.

As imagens de treinamento foram enviadas para o método *tree.DecisionTreeClassifier().fit()* para obter a árvore treinada. E seus resultados foram observados com o método *clf.predict()*, com 95% de acurácia.

Na Figura 27, mostra a matriz de confusão para os resultados obtidos, utilizando o método *confusion_ matrix()* do *sklearn*.

Figura 27 – Matriz de confusão para a árvore de decisão



FONTE: Autoria Própria

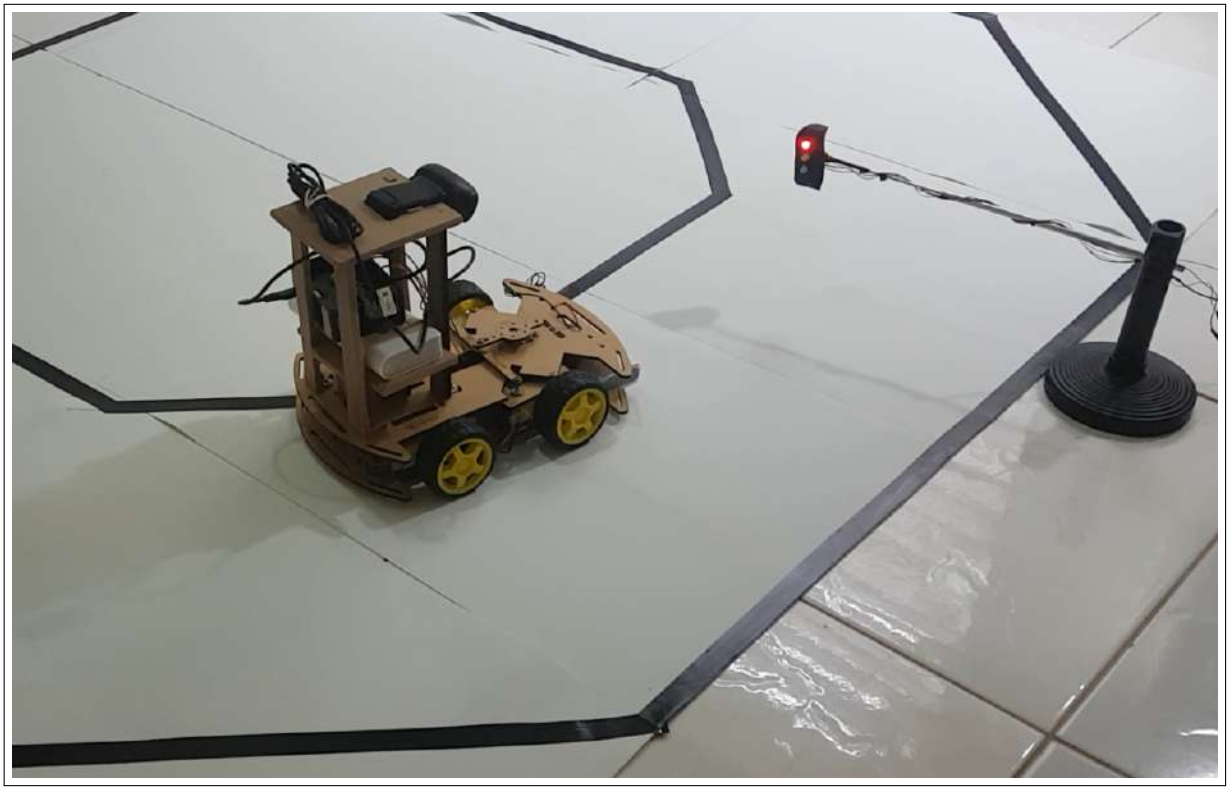
4.4 Experimento do Protótipo Final

Para testar todas as aplicações juntas, foi colocado o protótipo do carro na pista de tese e o semáforo em uma determinada posição da pista. O objetivo do teste era observar se carro seguia corretamente a trajetória da pista e se parava quando o sinal do semáforo fosse vermelho. Para que em todas as voltas, quando tivesse o encontro do carro com o semáforo, fossem aplicado a leitura do sinal vermelho, foi fixado o sinal vermelho no semáforo, fazendo a troca para o sinal verde quando tivesse a confirmação que o carro parou.

Foram observadas 15 voltas, em cada volta o veículo precisava parar no sinal vermelho e só continuar sua movimentação quando houvesse a troca para o sinal verde. Em 12 voltas, o protótipo efetuou corretamente o experimento. E em 3 voltas, o veículo não parou no sinal vermelho, devido ao não reconhecimento do semáforo pelo algoritmo. Na Figura 28, mostra uma imagem capturada no momento da execução do teste.

Toda a execução foi gravada e pode ser acessada no Apêndice A.1. O vídeo possui um contador de acertos e erros, legenda para a configuração do sinal e legenda para a velocidade do vídeo, destacando o encontro do protótipo do carro com o semáforo.

Figura 28 – Execução do protótipo final na pista de teste com o semáforo



FONTE: Autoria Própria

5 Considerações Finais

Neste trabalho, foi avaliado a autonomia do protótipo do carro em seu percurso na pista de testes. Seguindo corretamente o percurso com a leitura da pista e a interpretação do semáforo. Utilizando processamento de imagens e aprendizagem de máquina nas imagens capturadas pela câmera acoplada ao protótipo do carro. Foram executadas 15 voltas na pista, entre elas, o veículo obteve o reconhecimento correto do semáforo em 12 delas. Demostrando um bom resultado, e aprovação da combinação do método *Haar cascade* com a árvore de decisão.

Além do método *Haar casade*, o método *template match* foi experimentado para a identificação do semáforo, mas não obteve bons resultados. Identificou uma quantidade menor de semáforo e os recortes nas imagens, nos limites do semáforo, não foram adequados, principalmente com o protótipo do carro em movimento.

A utilização das imagens do trabalho do (FRIDMAN, 2019) foram essenciais para apuração das características das imagens, necessárias para a árvore de decisão. Porém, não foram empregadas no treino do método em seu estado final. Visto que, usando apenas as imagens do protótipo, obteve melhor classificação dos sinais no experimento final. Mas vale destacar que, para uma maior generalização do método, é apropriado a utilização dessas imagens.

Embora o experimento tenha mostrado resultados positivos, algumas melhorias podem ser realizadas a fim de aprimorar a aplicação. A verificação da decisão do carro junto com sua movimentação torna o seu deslocamento lento. Uma forma para aperfeiçoar isso, seria aplicar um novo método de inteligência artificial para prever o próximo movimento e executar em paralelo o algoritmo de reconhecimento de semáforo.

REFERÊNCIAS

ALMEIDA, P. R. M. T. D. *Introdução ao Processamento de Imagens de Sensoriamento Remoto*. Brasília: Universidade de Brasília, 2012.

BEKEY, G. A. Autonomous robots. *Cambridge University Press*, 2001. Disponível em: <https://www.cambridge.org/core/journals/knowledge-engineering-review/article/abs/on-autonomous-robots/5951E36C4F659FD5BB2DF5F89C0E54D9#article>.

BRADSKI, A. K. G. *Learning OpenCV 3 Computer Vision in C++ With The OpenCV Library*. 1ª ed. ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, 2017.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. 1ª ed. ed. Sebastopol: O'Reilly Media, 2008.

CHEVRAND, M. *Robôs em alto-mar*. 2022. Disponível em: <https://revistapesquisa.fapesp.br/robos-em-alto-mar/>.

COELHO, R. L. D. A. D. Aplicação de ferramenta open source em sistemas de visão computacional - desenvolvimento de um veículo seguidor autônomo. *Congresso Brasileiro de Educação em Engenharia*, 2011.

DEFRIES, M. H. R. D. R. Classification trees: an alternative to traditional land cover classifiers. *International Journal of Remote Sensing*, 1996.

FRIDMAN, L. Mit deep learning basics: Introduction and overview with tensorflow. *medium*, 2019. Disponível em: <https://medium.com/tensorflow/mit-deep-learning-basics-introduction-and-overview-with-tensorflow-355bcd26baf0>.

HAK, L. T. H. P. H. J. N. T. N. T. J. J. Applying the haar-cascade algorithm for detecting safety equipment in safety management systems for multiple working environments. *Electronics*, 2019.

HEINEN, F. J. *Sistema de Controle Híbrido para Robôs Móveis Autônomos*. Monografia (Dissertação de Mestrado) — Universidade do Vale do Rio dos Sinos - UNISINOS, São Leopoldo, Brasil, 2002.

JATAKIA, S. K. D. K. P. S. C. B. J. Human skin detection using rgb, hsv and ycbcr color models. *Atlantis Press*, 2017.

JÚNIOR, J. P. do N. *Controle de Trajetória Em Robô Autônomo Utilizando Processamento de Imagens e Redes Neurais Artificiais*. Monografia (TCC) — Instituto Federal da Paraíba - IFPB, João Pessoa, Brasil, 2022.

LEGOWIK, R. B. T. H. S. Mobile robot and mobile manipulator research towards astm standards development. *SPIE Commercial Scientific Sensing and Imaging*, v. 9872, 2016.

MATHER, B. T. P. *Classification methods for remotely sensed data*. 2ª ed. ed. New York: Taylor Francis Group, 2009.

NETO, O. M. F. H. V. *Processamento Digital de Imagens*. 1ª ed. ed. Rio de Janeiro: Brasport, 1999.

NUMPY. *NumPy Reference*. 1.22. ed. [S.l.], 2022. Disponível em: <https://numpy.org/doc/stable/reference/index.html>.

OPENCV. *OpenCV About*. [S.l.], 2022. Disponível em: <https://opencv.org/about/>.

OPENCV. *OpenCV modules*. 4.5.5-dev. ed. [S.l.], 2022. Disponível em: <https://docs.opencv.org/4.x/index.html>.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

RICHARD, C. R. E. *Digital Image Processing*. 4ª ed. ed. 330 Hudson Street, New York, NY 10013: Pearson Education, 2018.

SENATRAN. *Quantidade de Infrações (Notificação de Penalidade)*. 2022. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/conteudo-Senatran/estatisticas-quantidade-de-infracoes-denatran>.

SIDRA. *Taxa de Mortalidade por Acidentes de Trânsito*. 2019. Disponível em: <https://sidra.ibge.gov.br/tabela/6698>.

STONE, L. B. J. H. F. R. A. O. C. J. *Classification And Regression Trees*. 1ª ed. ed. 6000 Broken Sound Parkway NW: Chapman Hall, 1984.

TAYLOR, D. M. D. S. C. *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood, 1994.

VIOLA, M. J. P. "integrated feature extraction for image retrieval. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. Disponível em: <https://ieeexplore.ieee.org/document/990517>.

APÊNDICES

A Principais links de acesso

A.1 Vídeo da execução do resultado final do trabalho

O vídeo encontra-se disponibilizado na plataforma de compartilhamento de vídeos *YouTube*, disponível em: <https://youtu.be/GeN9sPDKGgs>

A.2 Repositório do código do trabalho

O repositório encontra-se disponibilizado na plataforma *GitHub*, disponível em: <https://github.com/mateusgomesc7/opencv-traffic-light>

B Principais códigos na execução do trabalho

B.1 Código para identificação do semáforo com o *Haar cascade*

```
def find_traffic_light(image):
    top_left = None
    bottom_right = None
    image_crop = None
    # Carregar arquivo XML para o metodo Haar Cascade
    cascade = cv.CascadeClassifier('./cascade.xml')
    # Identificando resultados encontrados
    faces = cascade.detectMultiScale(image, 1.3, 5)
    for (x,y,w,h) in faces:
        top_left = (x, y)
        bottom_right = (x+w,y+h)
        image_crop = image[top_left[1]:top_left[1]+h, top_left[0]:
   top_left[0]+w]
    # Retorno da imagem do semaforo recortada e sua posicao na imagem
   original
    return image_crop, top_left, bottom_right
```

B.2 Código para captura de características da imagem para a árvore de decisão

```
def get_features(image):
    # Redimensionando da imagem
    img = cv.resize(image, (30, 40), interpolation=cv.INTER_AREA)
    # Convertendo BGR para RGB
    img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    # Convertendo BGR para HSV
    img_hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
    # Pegando imagem com mascara e posicao com maior brilho
    img_rgb_mask, img_hsv_mask, pos = get_mask_and_pos(img_rgb, img_hsv)
    # Media das cores RGB
    colors_rgb = [np.sum(img_rgb_mask[:, :, 0] / 255.0),
                  np.sum(img_rgb_mask[:, :, 1] / 255.0),
                  np.sum(img_rgb_mask[:, :, 2] / 255.0)]
    # Media das cores HSV
    colors_hsv = [np.sum(img_hsv_mask[:, :, 0] / 255.0),
                  np.sum(img_hsv_mask[:, :, 1] / 255.0),
                  np.sum(img_hsv_mask[:, :, 2] / 255.0)]

    dataset = colors_rgb + colors_hsv
    dataset.append(pos)
    # Retorno da lista de caracteristicas
    return [dataset]
```

B.3 Código de combinação do *Haar cascade* com a árvore de decisão

```
def get_traffic_light(frame):
    # Identificacao do semaforo com o Haar Cascade
    frame_crop, top_left, bottom_right = hc.find_traffic_light(frame)
    # Verifica se o semaforo foi identificado pelo Haar Cascade
    if type(frame_crop) is np.ndarray:
        # Captura caracteristicas da imagem do semaforo
        feature = dt.get_features(frame_crop)
        # Pega sinal do semaforo com a Arvore de Decisao
        result = dt.predict_frame(feature, clf)

        # Verifica o resultado do sinal para retornar texto
        text = ''
        if (result == np.array([[1., 0., 0.]])).all():
            text = 'red'
        elif (result == np.array([[0., 1., 0.]])).all():
            text = 'yellow'
        elif (result == np.array([[0., 0., 1.]])).all():
            text = 'green'
        # Retorna sinal e posicao do semaforo na imagem original
        return text, top_left, bottom_right
    else:
        # Retorna vazio caso nao encontre semaforo
        return '', '', ''
```

B.4 Código para movimentação do carro com a verificação do semáforo

```
# Captura da imagem da camera
(ret, frame) = cam.read()
# Redimensionando da imagem
frame = cv.resize(frame, (largura_img, altura_img))
# Verifica mais de uma vez (number_of_check_signal vezes) a existencia
# de um semaforo com o sinal vermelho ou amarelo
signal = ''
for check_signal in range(number_of_check_signal):
    # Pega sinal do semaforo se existir semaforo
    signal, top_left, bottom_right = get_traffic_light(frame)
    # se vermlho ou amarelo, captura uma nova imagem
    if signal == 'red' or signal == 'yellow':
        break
# Se sinal verde ou nao existir semaforo, carro ganha movimento
if signal == 'green' or signal == '':
    output = get_road(frame)
    choose_movement(output)
    time.sleep(1)
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, João Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

TCC

Assunto: TCC
Assinado por: Mateus Gomes
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Mateus Gomes Correia**, **ALUNO (20162610012) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA**, em 11/12/2022 16:31:44.

Este documento foi armazenado no SUAP em 11/12/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 689109
Código de Autenticação: 4313e19854