UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA – PATOS DE MINAS
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

SAMUEL SANTOS DE OLIVEIRA

ROBÔ MÓVEL AUTÔNOMO COM SISTEMA DE NAVEGAÇÃO BASEADO EM VISÃO COMPUTACIONAL PARA O ENSINO DE ENGENHARIA.

Patos de Minas - MG
2020

SAMUEL SANTOS DE OLIVEIRA

ROBÔ MÓVEL AUTÔNOMO COM SISTEMA DE NAVEGAÇÃO BASEADO EM VISÃO COMPUTACIONAL PARA O ENSINO DE ENGENHARIA.

Relatório final apresentado à Universidade Federal de Uberlândia como requisito para conclusão da iniciação científica do curso de graduação em Engenharia Eletrônica e de Telecomunicações, da Faculdade de Engenharia Elétrica.

Orientador: Prof. Dr. Daniel Costa Ramos

Patos de Minas - MG
2020

ROBÔ MÓVEL AUTÔNOMO COM SISTEMA DE NAVEGAÇÃO BASEADO EM VISÃO COMPUTACIONAL PARA O ENSINO DE ENGENHARIA.

RESUMO

Os robôs moveis autônomos apresentam diversas aplicações e tem destaque na área de mobilidade urbana. No entanto, ainda existem diversas barreiras e desafios para esta tecnologia como o alto custo, tecnologias mais confiáveis, normas, regulamentos e aspectos de segurança. Existem diversas iniciativas para ensino de conteúdos de engenharia por meio da robótica, como a proposta do MIT denominada Duckietown, na qual utiliza robôs de baixo custo denominados Duckiebots com grande capacidade de processamento e sensoriamento, incluindo o uso de visão computacional. Baseado em estudos da área e nas iniciativas para ensino, é proposto que a elaboração de plataforma similar ao Duckiebot, mas com os ajustes necessários para minimizar custos e refletir a realidade brasileira, como identificação de placas em português e de outras possíveis particularidades nacionais. Para esta proposta foi estabelecido como objetivo inicial, a construção e adaptação de um robô móvel autônomo móvel, para ensino para engenharia. O desenvolvimento do protótipo apresentou vários desafios, principalmente pela dificuldade de utilizar os algoritmos originais do projeto Duckiebot e a suspensão das atividades acadêmicas na UFU devido a pandemia. Após passar por adaptações, o projeto apresentou resultados promissores ao utilizar códigos de elaboração própria, onde o protótipo foi capaz de utilizar a câmera, o hat de motores de menor custo e identificar placas de “pare”, atendendo de forma parcial os objetivos almejados para o projeto.

Palavras-chave: robô móvel autônomo, visão computacional, robôs educacionais.

AUTONOMOUS MOBILE ROBOT WITH A COMPUTER VISION NAVIGATION SYSTEM FOR TEACHING IN ENGINEERING

ABSTRACT

Autonomous mobile robots have several applications and stand out in the area of urban mobility. However, there are still several barriers and challenges for this technology such as high cost, more reliable technologies, standards, regulations and safety aspects. There are several initiatives for teaching engineering content through robotics, such as the MIT proposal called Duckietown, in which it uses low-cost robots called Duckiebots with great processing and sensing capacity, including the use of computer vision. Based on studies of the area and teaching initiatives, it is proposed the development of a platform similar to Duckiebot, but with the necessary adjustments to minimize costs and reflect the Brazilian reality, such as identification of signs in Portuguese and other possible national particularities. For this proposal, the initial objective was to build and adapt a mobile autonomous mobile robot for teaching engineering. The development of the prototype presented several challenges, mainly due to the difficulty of using the original algorithms of the Duckiebot project and the suspension of academic activities at UFU due the pandemic. After undergoing adaptations, the project showed promising results when using codes of its own elaboration, where the prototype was able to use the camera, the lower cost hat for motors and it was able to identify “pare” plates, partially meeting the objectives aimed at the project.

Keywords: autonomous mobile robot, computer vision, educational robots.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

A robótica é um ramo multidisciplinar que envolve diversos setores e que tem se destacado nos últimos anos, de modo que suas aplicações estão presentes em toda cadeia produtiva e para os mais diversos fins, abrangendo casos de extrema importância como satélites, agricultura e indústria em geral, assim como casos de entretenimento e educação.

A área da robótica está em constante evolução e está cada vez mais presente no dia a dia, principalmente no que diz respeito aos robôs autônomos. Um robô autônomo é um dispositivo físico capaz de realizar tarefas e tomar decisões em diversos tipos de ambientes e situações, sem interferência humana, utilizando seus sensores e atuadores [1].

Para tomar as decisões de forma autônoma o robô necessita ter um adequado conjunto de sensores, cujas as tecnologias podem envolver conceitos de visão computacional, ondas ultrassônicas, mapeando 3D à laser, dentre outras [2]. Essa versatilidade e multidisciplinaridade dos robôs os tornam ideais para serem utilizados no ensino STEM (*Science, Technology, Engineering, and Mathematics*) [3], focando principalmente na área de engenharia [4][5].

A visão computacional, definida como o processo de modelagem e replicação da visão humana usando software e hardware, foi e é amplamente utilizada em diversos sistemas autônomos [6]. Este sistema permite que a máquina compreenda uma cena, vídeo ou foto, extraindo informações úteis. Quando a visão computacional é utilizada em conjunto com os robôs autônomos, estes tornam-se capazes de identificar e classificar objetos, obstáculos e recursos do ambiente, permitindo que o robô compreenda o meio no qual está inserido, tornando a mais eficiente e menos suscetível a erros.

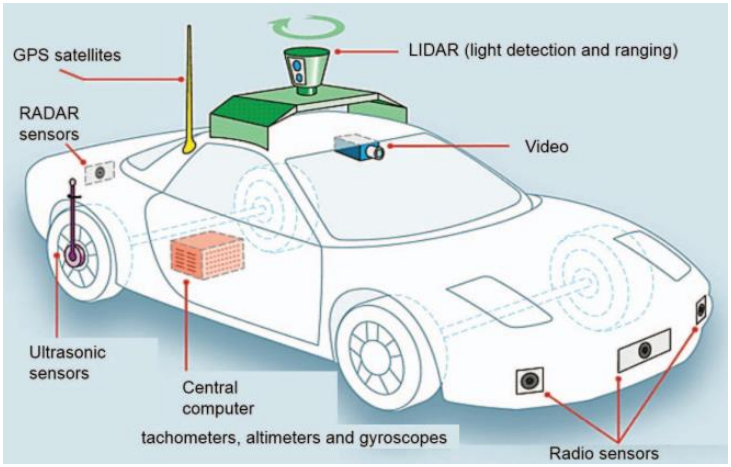
São vários os exemplos do benefício desta integração nos mais diversos campos de atuação da robótica: o sistema de visão computacional pode ser utilizado na agricultura para identificar pragas, fazer a colheita seletiva de alimentos ou semeadura de precisão, na indústria para identificação de peças, manter perímetros de segurança ou avaliação de conformidade, na interação homem-máquina para interagir e reagir às emoções humanas ou no transporte de pessoas, atuando na navegação autônoma e no desvio de obstáculos.

Um exemplo de aplicação é na área de mobilidade urbana, onde os carros autônomos prometem mudar a maneira como nos locomovemos, proporcionando redução dos congestionamentos, redução de poluição para o meio ambiente e principalmente a redução de acidentes de trânsito [7]. Isso porque, um veículo controlado automaticamente não precisa lidar

com aspectos humanos – cansaço, distração, irritabilidade, imprudência ou consumo de bebidas alcoólicas – na direção [8].

As tecnologias utilizadas no carro autônomo detectam as informações e realizam os cálculos, com processamento bem mais rápido que o ser humano. Para isso, além do conjunto de câmeras e da visão computacional, são utilizadas tecnologias de Radar, Sonar e o Lidar (Light Detection And Ranging), GPS, acelerômetro, odômetro e de redes de comunicação entre carros [8]. Essa combinação de informações provenientes dos sensores e das câmeras permite que o carro execute tarefas.

Figura 1. Sistema de sensoriamento de um carro autônomo.



Fonte: Adaptado de [7].

O nível e complexidade de automação dos carros autônomos, segundo a norma SAE J3016 da Sociedade Internacional de Engenheiros Automotivos (SAE International) [9], pode ser dividida em seis níveis:

- **Nível 0 - Nenhuma automação:** carros tradicionais, sem nenhum tipo de automação.
- **Nível 1 - Assistência do motorista:** modelos que se mantêm dentro das marcações de pistas e tenha *cruise control* adaptável.
- **Nível 2 - Automação parcial:** o veículo é capaz de ajudar com direção e aceleração e permite que o motorista deixe algumas tarefas de lado.
- **Nível 3 - Automação condicional:** o veículo controla o ambiente por inteiro e a automação total funciona em situações e ambientes específico, como estradas.
- **Nível 4 - Automação alta:** para um grande número de situações o veículo é capaz de dirigir, frear, acelerar e monitorar o que ocorre ao seu redor, determinar quando trocar de faixas e virar.

- **Nível 5 - Automação total:** a direção não exige nenhuma intervenção humana. O veículo é capaz de se dirigir em qualquer situação, com engarrafamentos ou em pistas sinuosas pela cidade.

Entretanto apesar de ser um assunto relevante e de envolver interesse de diversos âmbitos, o desenvolvimento de carros autônomos são de elevada complexidade e exigem conhecimentos específicos de diversas áreas, incluindo conhecimentos de eletrônica, sistemas embarcados, mecânica, programação de baixo e alto nível, controle, sensoriamento e de visão computacional. Uma forma de mitigar essa dificuldade e incentivar a pesquisa nesta área, se dá por meio da utilização de protótipos robóticos simplificados, ou seja, versões miniaturizadas dos carros autônomos em forma de robôs autônomos em um ambiente controlado.

Baseado neste contexto, é proposto neste trabalho o desenvolvimento de um protótipo robótico com recursos de visão computacional com foco principal no processamento de imagens.

Desta forma, tem-se como objetivo uma plataforma de baixo custo, de fácil confecção, reprodução e alteração, de tecnologias abertas e de uso comum. Como objetivo secundário, visa-se promover a divulgação científica por meio de trabalhos atraentes ao público.

1.1 OBJETIVOS

O objetivo deste trabalho é realizar um estudo sobre as técnicas de visão computacional e consequente a elaboração de um protótipo de carro móvel autônomo, compatível com a realidade do brasileiro e com capacidade de reconhecer placas nacionais e outras particularidades.

Objetivo Específico 1: Projetar e implementar um algoritmo para controle do robô;
Objetivo Específico 2: Desenvolver e implementar um algoritmo para identificação de placas de trânsito através do Raspberry Pi.
Objetivo Específico 3: Projetar e construir um protótipo de um carro móvel autônomo utilizando conceitos de visão computacional.

2 ATIVIDADES TEÓRICAS DESENVOLVIDAS

Para compreender o funcionamento da visão computacional aplicada a robôs, foi desenvolvido um protótipo robótico de um carro autônomo. Para este fim foi realizado diversos estudos sobre a visão computacional e robótica, e projetar um robô que atenda a essas demandas.

2.1 TRABALHOS RELACIONADOS

As pesquisas com carros autônomos tem sido alvo de diversas empresas, como a Google, a Mercedes-Benz, Nissan, General Motors, dentre outras [7]. Mas ainda existem diversas barreiras e desafios para esta tecnologia como o alto custo, tecnologias mais confiáveis, normas, regulamentos e segurança.

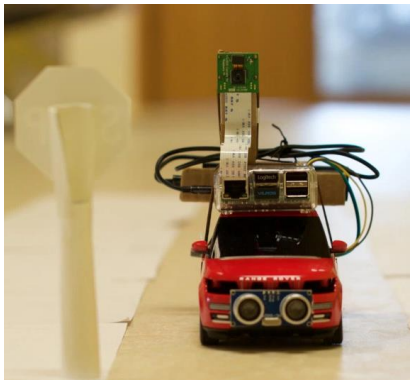
No Brasil, dois projetos de carros autônomos se destacam, o Cadu [10] e a CaRINA [11]. O Cadu foi desenvolvido pelo Grupo de Pesquisa e Desenvolvimento de Veículos Autônomos (PDVA), vinculado à Escola de Engenharia da Universidade Federal de Minas Gerais (UFMG), onde seu movimento é guiado por coordenas GPS inseridas em seu sistema. Já na Universidade de São Paulo (USP), o CaRINA é um projeto pioneiro, sendo o primeiro carro autônomo do país a operar de maneira autônoma em sua plenitude e o primeiro projeto a receber autorização oficial para testes em vias públicas na américa latina.

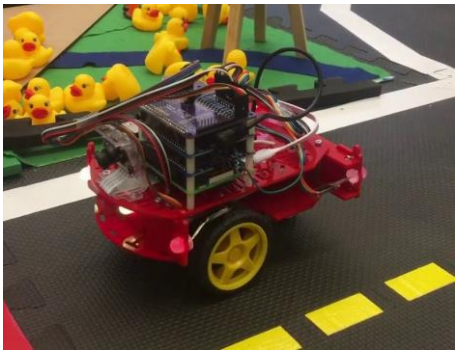
A utilização de protótipos robóticos visa diminuir a complexidade do problema dos carros autônomos, possibilitando a manipulação e um melhor controle do ambiente e o teste de conceitos de hardware e software antes de uma implementação em escala real.

O projeto Donkey Car [12], indicado na Figura 2 - a), surgiu do interesse de dois entusiasmados com os carros autônomos, sendo hoje uma biblioteca livre que permite uma experimentar com o conceito de carro autônomo. O protótipo é baseado em carros radio-controlados (RC) adaptados para serem controlados por um Raspberry Pi, conectado a uma câmera e um shield para os servo-motores.

Outro projeto, também baseado na modificação de um carro RC, foi desenvolvido em [13], indicado na Figura 2 – b), utilizando o modelo atual do Raspberry Pi (3 B+), sensores de ultrassom, câmera e comunicação via Wi-Fi. O projeto permite o processamento de imagens e a utilização de redes neurais para o controle autônomo do robô.

Figura 2. Projetos de robôs autônomos utilizando visão computacional: a) Projeto Donkey Car [12]; b) Zheng Wang RC Car [13]; c) Android's Autonomous Vehicle [14]; e projeto Duckietown [15][16].

a)  b) 

c)  d) 

Fonte: [12], [13], [14] e [15].

Há iniciativas utilizando outras tecnologias, como o projeto realizado em [14], indicado na Figura 2 – c). Neste projeto é utilizado um smartphone como controlador e câmera, onde o mesmo se comunica via Bluetooth com uma placa Arduino que realiza o controle de baixo nível dos sinais do robô (sensoriamento e atuação dos motores).

Dentre os diversos projetos desta área, o projeto aberto Duckietown [15], indicado na Figura 2 - d), foi criado como instrumento de aula no Massachusetts Institute of Technology – MIT e hoje, é uma ferramenta utilizada em escala global para estudos sobre robótica e inteligência artificial em carros autônomos.

O projeto utiliza robôs baseados na tecnologia Raspberry, conectado à sensores e à uma câmera, para emular um carro autônomo capaz de reconhecer marcações nas pistas, sinais de transito, semáforos e outros robôs, estes denominados de Duckiebots. A recepção ao projeto foi tão positiva, que em poucos anos já é utilizado em mais de 500 instituições de ensino [15] para ensino de disciplinas STEM, pesquisa e entretenimento.

Uma pesquisa realizada recentemente [16], cita diversas plataformas de robótica para ensino, ressaltando o custo benefício da plataforma Duckietown.

Estes trabalhos permitiram a observação de umas grandes quantidades de estudos e propostas de aplicação, sem haja um estudo detalhado sobre os classificadores utilizados para visão computacional e algoritmo de controle do robô e bem como também estudos em português na área em estudo.

2.2 REFENCIAL TEÓRICO

Para compreensão do projeto é necessário aprofundar o conhecimento em alguns conceitos teóricos importantes, dentre eles, foram destacados o algoritmo Viola-Jones, o AdaBoot e o sistema de acionamento por motores de corrente contínua.

2.2.1 Raspberry Pi

Para que o robô móvel autônomo possa se locomover e identificar placas, é preciso que o robô adquira informações dos meios externos através de uma câmera por exemplo. Existem diversas soluções para realizar o processamento do sinal da câmera, como o uso de celulares, computadores ou de pequenos microcomputadores embarcados, como o Raspberry Pi. Este último foi escolhido devido a sua simplicidade e de seu uso habitual na robótica.

O Raspberry Pi é um microcomputador de baixo custo altamente popular. Foi desenvolvido em 2006, na Universidade de Cambridge, Reino Unido, por uma equipe de professores e alunos, com o objetivo simples levar as escolas uma ferramenta para ensino da computação.

A fundação mantedora do Raspberry Pi, recomenda a instalação de uma versão do Debian otimizada para o uso no dispositivo. Entretanto atualmente com a ascensão do IoT o Raspberry Pi virou uma ferramenta amplamente utilizada, surgindo outros sistemas igualmente otimizados para serem utilizados neste dispositivo, inclusive uma versão da Microsoft chamada Windows 10 IoT Core.

Dentre os modelos mais utilizados, destaca-se o modelo 3 B+ e o mais recente, o Raspberry 4.

2.2.2 Câmera

Para a identificação das placas o robô utiliza uma câmera de grande abertura focal denominada câmera olho de peixe (Fisheye).

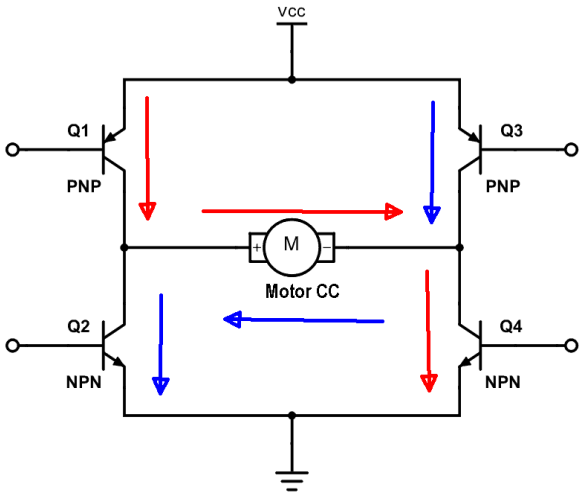
Figura 3. Câmera olho de peixe.

Fonte: [22]

2.2.3 Motor CC e Ponte H

Para o deslocamento do robô são utilizados dois motores de corrente contínua, onde um é posicionado na parte traseira do robô e o outro na parte dianteira. O motor posicionado na região traseira do robô irá determinar a velocidade e sentido de deslocamento do robô. O motor na região dianteira determinara a direção do robô. E para determinar o sentido e a velocidade dos motores é utilizado um circuito denominado Ponte H (*H-Bridge*).

Figura 4. Esquema eletrônico da Ponte H.



Para atender os requisitos do projeto de desenvolvimento do robô, que tenha compatibilidade com o Raspbarry Pi o driver de motor Stepper Motor HAT v0.1 (Figura 5) é adequado.

Figura 5. Driver dos motores, modelo Stepper HAT v0.2.

Fonte: [23]

2.2.4 Python

Python é uma linguagem de programação de alto nível, desenvolvida na Holanda pelo professor Guido Van Rossum do departamento de programação do Instituto Nacional de Matemática e Ciência da Computação da Holanda em 1991. E foi desenvolvida para ser de fácil compreensão e altamente eficaz. Esta linguagem ainda apresenta algumas caraterísticas que a tornam extremamente simples e coesa, como o baixo uso de caracteres especiais, coletor de lixo para gerenciamento automático do uso da memória, se apresenta como uma linguagem expressiva, voltada para o programador e outros muitas características.

Como python é denominada uma linguagem de alto nível é geralmente caracterizada como uma linguagem orientada a objetos, mas também pode ser utilizada para programas em estrutura simples e rápidos, mas também em estruturas de dados mais complexas como tuplas e listas. A linguagem Python, é livre, ou seja, open souser, e inclusive pode ser utilizada par fins comerciais, e a origem de seu nome se dá através de um grupo humorístico britânico chamado Moty Python, embora atualmente as pessoas associam ao réptil python (em português cobra).

2.2.5 Visão Computacional

A visão computacional é um ramo da inteligência artificial, e é a ciência capaz de explicar e modelar a visão a humana para as maquinas interpretarem o mundo visual.

Entretanto reproduzir a visão humana não é uma tarefa simples e exige um alto custo computacional. Mas mesmo com as dificuldades esta vem ganhando cada vez mais espaço no

mundo atual, seja para uso militar, veículos autônomos, marketing, segurança e outros fins. Segundo Wayne Thompson, cientista de dados da SAS, a visão computacional é uma das tecnologias mais impressionantes geradas pela inteligência artificial.

2.2.6 Caraterísticas Haar Retangulares

As características Haar, são baseados em formas geométricas retangulares. E são utilizadas para representar características de uma imagem. Como pode se visualizar na figura [90], as características Haar são compostas de várias áreas em preto e branco combinadas de uma forma em que os valores de intensidade nos pixels nas áreas de diferentes tonalidades são acumulados de forma separada [27].

Figura 6. Alguns exemplos de features Haar utilizadas para detecção de padrões.

Fonte: [27]

O valor da característica de uma imagem é calculado pela combinação ponderadas dessas duas somas, onde a soma dos retângulos brancos e subtraídas das somas dos valores dos retângulos pretos, como mostra a equação abaixo.

$$\Delta = \frac{1}{n}\sum_{preto}^{n} I(x) - \frac{1}{n}\sum_{branco}^{n} I(x)$$

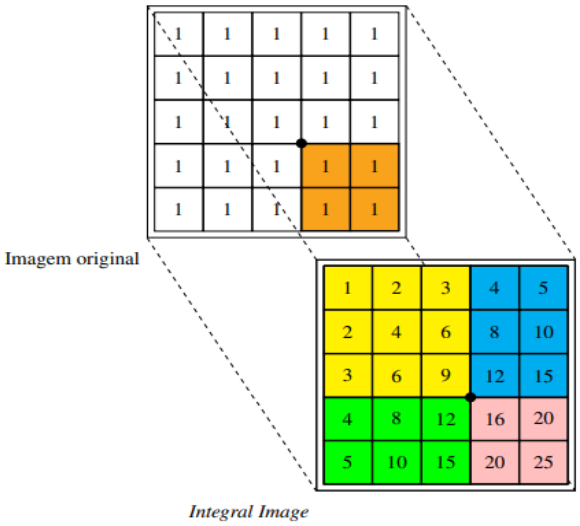
2.2.7 Imagem Integral

A imagem integral é uma matriz bidimensional do tamanho da imagem original, onde soma-se os valores de cada pixel a esquerda e acima do vértice em questão, como mostra a equação abaixo, aplicada a imagens original.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

onde ii(x, y) é a imagem integral e i(x', y') é a imagem original [https://juliobs.com/Julio_Batista_Silva-TCC-Face_Recognition.pdf]. A figura [T] exemplifica a Imagem Integral, sobre uma imagem 5x5.

Figura 7. Exemplo de uma Imagem Integral.



Fonte: [28]

2.2.8 Adabost

Um classificador é um algoritmo de inteligência artificial que tem por objeto simples, separar um dado conjunto de informações em dois ou mais grupos. Um classificador forte pode ser construído através da combinação de vários classificadores simples (simples = usar apenas uma característica Haar para cada característica).

O algoritmo Viola-Jones utiliza uma variante do AdaBost para selecionar as melhores características e treinar o classificador. O AdaBoost é baseado em uma técnica de classificação baseada na forma dos objetos a serem classificados, e foi inventado por Yoav Freund e Robert Schapire [30].

O algoritmo AdaBost utiliza o método Bosting [31] que forma um classificador forte através da combinação de classificadores simples [32]. O Boosting é utilizado a cada estágio do classificador em cascata, para selecionar as melhores características (classificador fraco melhor que os anteriores) e gerar um classificador robusto.

2.2.9 Classificadores em Cascata

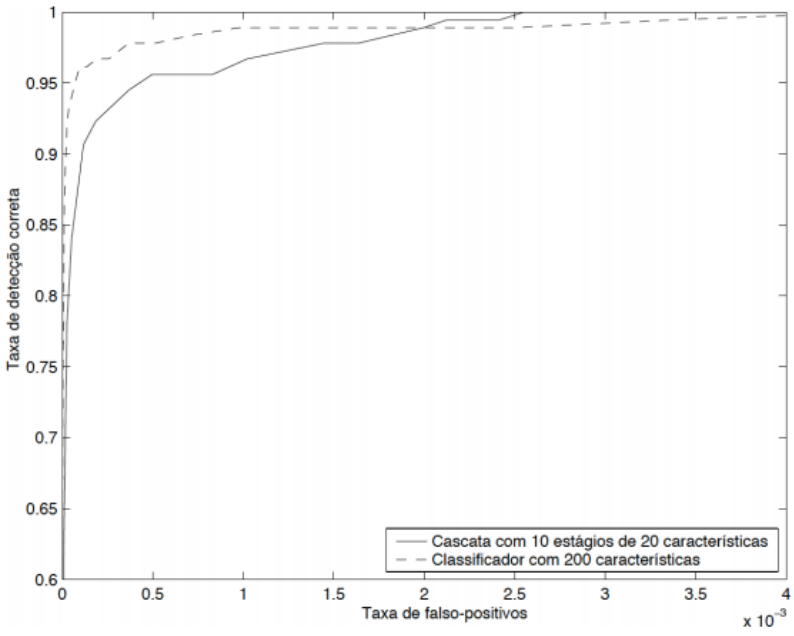
O algoritmo proposto por Paul e Jones [29] utiliza os classificadores Haar, e foi desenvolvido com o objetivo de detectar faces em tempo real, mas os classificadores Haar podem ser utilizados como já apresentado na detecção de diversos objetos, como placas de trânsito que é o objeto de estudo deste trabalho.

O classificador em cascata, é basicamente a junção de vários classificadores em sequência de forma a gerar uma estrutura complexa de classificação. Cada estágio do classificador em cascata contém um classificador eficiente gerado através do Boosting do AdaBoost.

O principal objetivo do AdaBoost aplicado ao classificador em cascata é redução dos falsos positivos, ou seja, se o objetivo for separar placas de trânsito de outras objetos, um falso negativo é classificar uma placa de publicidade como de trânsito por exemplo. E com a aplicação dos classificadores em cascata a probabilidade de incidência de um falso positivo ocorra em diminui a cada estágio do classificador.

Pode-se inferir a ainda a eficiência do classificador em cascata observando a Figura [26], onde é feita uma comparação entre um classificador com duzentas características e um classificador em cascata com 10 estágios de vinte características cada.

Figura 8. Curvas COR comparando classificador de 200 características com classificador em cascata com 10 estágios de 20 características cada



Fonte: [26]

Pode-se notar que o classificador em cascata obteve uma acurácia semelhante ao classificador com 200 características, e com um custo computacional reduzido e o tempo de execução quase 10 vezes menor como pode se ver em [26].

3 DESENVOLVIMENTO DO PROTÓTIPO

Para estudar e compreender o funcionamento dos carros autônomos, foi proposta a elaborado um protótipo de um robô móvel autônomo com equipamentos de baixo custo e facilmente encontrados em território brasileiro. Para este fim foi preciso estudar o hardware e softwares utilizados em outros projetos sobre carros autônomo e a parti deste estudo projetar um protótipo que atenda aos requisitos desejados. Os tópicos anteriores deste trabalho mostram o referencial teórico para desenvolvimento deste protótipo e os tópicos na sequencia mostram o desenvolvimento do protótipo, deste modo é apresentado neste capitulo o funcionamento de algumas funções do mesmo.

3.1 DESENVOLVIMENTO INICIAL E VISÃO COMPUTACIONAL

Para implementação do projeto de um simulador de carro autônomo foi utilizada as ferramentas Python, OpenCV e o Raspberry Pi.

A linguagem utilizada foi a Python na versão 3.7.3 em conjunto com algumas ferramentas como o numpy e scipy e a biblioteca de visão computacional OpenCV na versão 3.48, por causa de problemas de compatibilidade usou-se uma versão anterior em vez da mais atual.

A linguagem Python foi escolhida para ser utilizada no desenvolvimento do projeto, por apresenta uma comunidade ativa e uma grande presença de bibliotecas compatíveis como o OpenCV.

Para detecção das placas utilizou-se o algoritmo proposto por Paul Viola e Michael Jones em 2001 [29], para detecção de faces, mas que pode ser reproduzido para detecção de outros objetos, por apresentar uma performasse excelente, uma aplicação relativamente simples e rapidez de execução [24] e [25].

O detector de objetos baseado no algoritmo Viola-Jones apresenta 4 estágios: seleção de características Haar retangulares, geração de uma imagem integral, treino de classificadores por um algoritmo de machine learning baseado no AdaBoost e o último a utilização de classificadores em cascata, que tem por objetivo descartar regiões de fundo para focar em áreas mais prováveis de conter o objeto em estudo como pode ser visto em [26].

3.2 OPENCV

Para trabalhar com imagens é necessário realizar adequações no que se refere ao formato ou tamanho das mesmas. O OpenCV (Open Source Computer Vision) é uma biblioteca de visão computacional de código aberto, desenvolvida pela Intel no ano 2000, com o objetivo de levar a visão computacional a diversos ambientes tonando este conceito mais acessível para realização de pesquisas e desenvolvimento negócios.

A biblioteca do OpenCV é muito utilizada no âmbito empresarial devido a suas vastas funções e aplicações, e ainda apresenta uma grande documentação e uma comunidade ativa.

Para trabalhar com o OPENCV é necessário seguir os procedimentos exigidos pela organização no site (https://opencv.org/) e deve-se escolher a versão a ser baixada juntamente com o sistema operacional compatível. Como o OpenCV tem muitas funções e uma comunidade ativa recomenda-se aos usuários novatos utilizar versões anteriores e não as mais recentes, pois as versões anteriores são mais estáveis e já passaram por testes de correção de erros. E muitas dicas de programação existentes em artigos e sites já não funcionam em algumas das novas versões do OpenCV.

3.2.1 Biblioteca do OpenCV

A biblioteca do OpenCV foi desenvolvida para apresentar alto desempenho e alta aplicabilidade no âmbito profissional ou acadêmico. E possui interfaces em diversas linguagens de programação como C++, C, Python e Java. E apresenta muitas funções como manipulação de imagens e vídeos, calibração da câmera, percepção de profundidade, técnicas de reconhecimento de padrões, e um amplo suporte para as necessidades de visão computacional aplicada a robótica [25].

3.3 TREINAMENTO PARA IDENTIFICAÇÃO DE PLACAS PARE

Para detecção das placas o método escolhido é o classificador Haar cascade, este classificador consegue detectar padrões ao analisar imagens de objetos. Este classificador é baseado em aprendizado de máquina para detecção de objetos e atende aos requisitos para implementação em CPUs de baixa potência, como o Raspberry pi.

O classificador antes de ser utilizado deve ser treinado com muitas imagens 'positivas' (com placas de trânsito) e o mesmo tamanho de imagens 'negativas' (sem placas de trânsito).

Após o treinamento é gerado o arquivo .xml e as imagens são verificadas quadro a quadro e ocorre a detecção dos objetos em estudo, onde são extraídas as características e padrões usando o algoritmo viola Jones, disponível na biblioteca OpenCV. Os padrões gerados são baseados em características Haar que levam em consideração informações relevantes de uma imagem como o contraste de uma imagem.

A etapa de treinamento do classificador exige uma grande potência computacional e na etapa de detecção utilizando o classificador já treinado os custos computacionais é reduzido.

Notebook Samsung ExpertP
Processador de 3ª geração da linha Intel series i Core i5 3337U.
Memória RAM de 4GB, DDR3;
Disco Rígido SSD 240GB M.2;

Para o treinamento do classificador foi utilizado um conjunto de 36 imagens positivas com dimensões 100 x 100 e 252 imagens negativas com dimensões 100 x 100.

3.3.1 Banco De Imagens

Para composição do banco de imagens negativas foi utilizado o site IMAGENET (http://image-net.org/) e se utilizou no total 256 imagens com dimensões de 100 por 100 e extensão .png. As imagens coletadas eram diversas como as figuras abaixo.

Figura 9. Imagens diversas.

Fonte: IMAGENET

Um outro banco de dados foi construído realizando uma busca manual no buscador de imagens do Google, sendo utilizado no total 36 imagens com dimensões de 100 por 100 e extensão .png. A imagens coletadas eram de placas de trânsito, como pode ser visto nas Figura 10.

Figura 10. Placas de trânsito.

Fonte: Google.

3.3.2 Detecção de Placas

Na detecção das placas foi utilizado os arquivos com extensão .xml conhecido como Haar que é responsável por codificar informações sobre a existência de contrastes orientados entre regiões da imagem. As imagens foram amostradas quadro a quadro e as faces foram detectadas e extraídas usando o algoritmo Viola-Jones na implementação da biblioteca OpenCV.

Figura 11. Funcionamento do algoritmo.

Fonte: Autor.

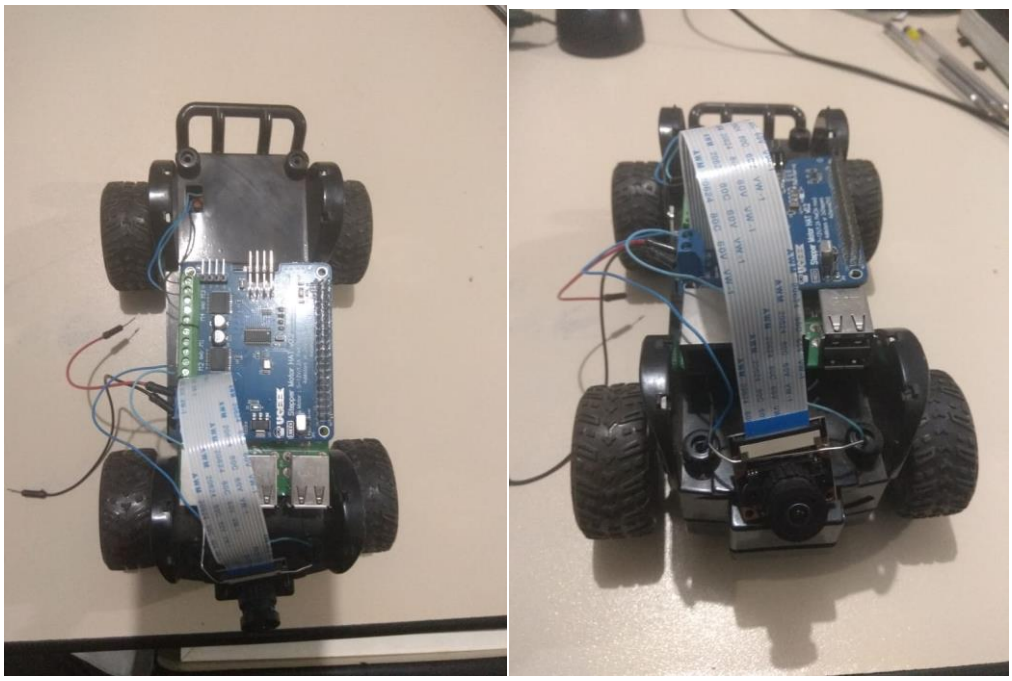
A figura mostra o funcionamento do algoritmo para detecção de placas aplicado a uma imagem presente na memória do Raspberry Pi. Entretanto apesar de que este algoritmo apresente um resultado preciso para identificação de placas em imagens estatística, o resultado deste mesmo processamento para identificação de placas em vario fremes simultâneos (vídeo) apresenta uma certa instabilidade na identificação e acaba gerando muitos falso-positivos na tentativa de identificar placas.

3.4 PROJETO CARRO AUTÔNOMO

Para simular e desenvolver um protótipo de carro autônomo, este foi projetado para atuar de maneira semelhante a um veículo automotor. A base do robô é um carro de controle remoto antigo que serve também como suporte a câmera e ao Raspberry Pi. O robô foi inspirado nos modelos de carros da Donkey Car.

O carro foi projetado conforme o esquema representado pela Figura 12.

Figura 12. Robô protótipo.



Fonte: Autor.

Tabela 1. Tabela de Custos do Protótipo.

Descrição	Quantidade	Custo
Raspberry Pi 3 Model B	1	R$ 389,99
Driver de Motor (Stepper Motor HAT v0.2)	1	R$ 120,00
Câmera	1	R$ 100,00
Cartão de memoria	1	R$ 34,90
Carregador	1	R$ 39,99
Total		R$ 684,88

Todos os componentes foram adquiridos pelo professor orientador**.**

Figura 13. Robô protótipo.

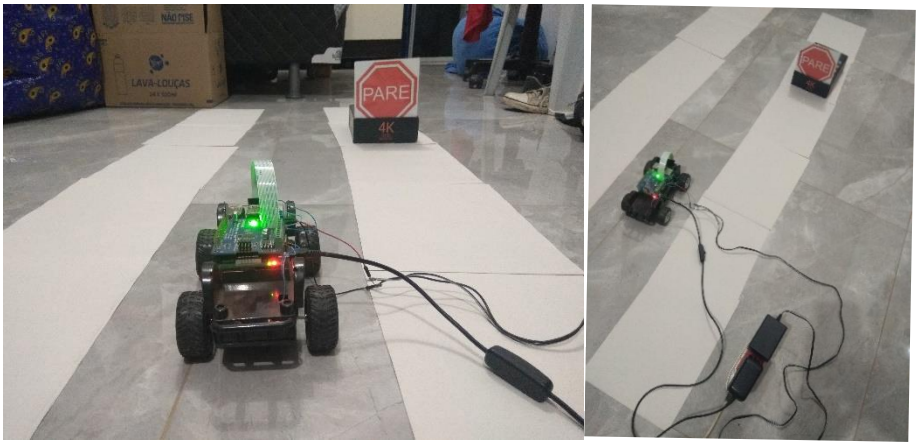
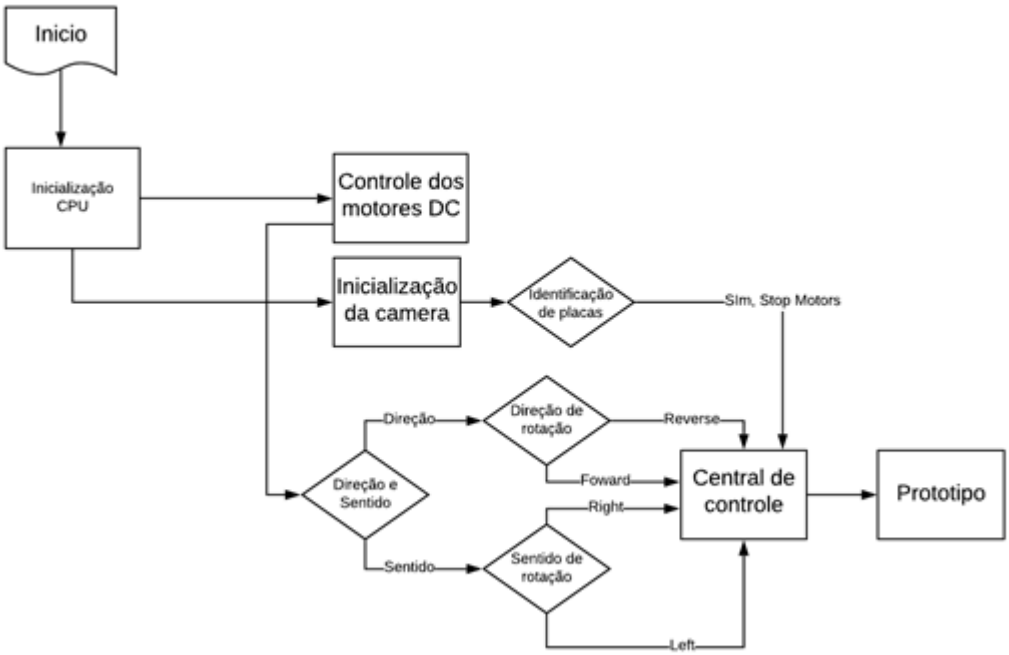


Figura 14. Fluxograma de funcionamento do protótipo



3.4.1 Projeto Código de Funcionamento Controle Veicular

O projeto do desenvolvimento do algoritmo para controle do robô protótipo, foi desenvolvido para ser controlado através de um teclado wireless conectado ao Raspberry Pi.

```python
#!/usr/bin/python
from Raspi_MotorHAT import Raspi_MotorHAT, Raspi_DCMotor

import sys
import time
import atexit



# create a default object, no changes to I2C address or frequency
mh = Raspi_MotorHAT(addr=0x6f)

# import curses
import curses

# Get the curses window, turn off echoing of keyboard to screen, turn on
# instant (no waiting) key response, and use special values for cursor keys
screen = curses.initscr()
curses.noecho()
curses.cbreak()
screen.keypad(True)



# recomendado para motores de desativação automática ao desligar!
def turnOffMotors():
    mh.getMotor(1).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(2).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(3).run(Raspi_MotorHAT.RELEASE)
    mh.getMotor(4).run(Raspi_MotorHAT.RELEASE)

atexit.register(turnOffMotors)

################################# Teste do motor DC!
myMotor = mh.getMotor(3)
myMotorD = mh.getMotor(1)


# Define a velocidade inical, de 0 (off) a 255 (max speed)
myMotor.setSpeed(80)
myMotorD.setSpeed(30)

myMotor.run(Raspi_MotorHAT.FORWARD);
myMotorD.run(Raspi_MotorHAT.FORWARD);

# ligue o motor
myMotor.run(Raspi_MotorHAT.RELEASE);
myMotorD.run(Raspi_MotorHAT.RELEASE);



#Tração veicular

###Forward = Para frente, Speed up = acelerar, Slow down = desacelerar, Backward = Para trás,
def forward(tf):
    #turnOffMotors()
    myMotor.run(Raspi_MotorHAT.FORWARD)
    time.sleep(tf)
    myMotor.run(Raspi_MotorHAT.RELEASE)


def reverse(tf):
    #turnOffMotors()
    myMotor.run(Raspi_MotorHAT.BACKWARD)
    time.sleep(tf)
    myMotor.run(Raspi_MotorHAT.RELEASE)

#Controle de direção

def left(tf):
    #turnOffMotors()
    myMotorD.run(Raspi_MotorHAT.FORWARD)
    time.sleep(tf)
    myMotor.run(Raspi_MotorHAT.RELEASE)
```

```
75
76
77  def right(tf):
78      #turnOffMotors()
79      myMotorD.run(Raspi_MotorHAT.BACKWARD)
80      time.sleep(tf)
81      myMotor.run(Raspi_MotorHAT.RELEASE)
82
83  def right(tf):
84      #turnOffMotors()
85      myMotorD.run(Raspi_MotorHAT.PAUSE_BRAKE)
86      time.sleep(tf)
87      myMotor.run(Raspi_MotorHAT.RELEASE)
88
89
90  try:
91      while True:
92          char = screen.getch()
93          if char == ord('q'):
94                  break
95
96      #Tração veicular
97          elif char == curses.KEY_UP:
98              print ("Para Frente! ")
99              forward(0.05)
100
101
102
103         elif char == curses.KEY_DOWN:
104             print ("Para trás! ")
105             reverse(0.05)
106
107     #Controle de direção
108         elif char == curses.KEY_RIGHT:
109             print ("Para direita! ")
110             right(0.05)
111
112         elif char == curses.KEY_LEFT:
```

```
113             print ("Para esquerda! ")
114             left(0.05)
115
116       #Freio
117         elif char == curses.PAUSE_BRAKE:
118             print ("Freio! ")
119             left(0.05)
120
121
122 finally:
123     #Close down curses properly, inc turn echo back on!
124     curses.nocbreak(); screen.keypad(0); curses.echo()
125     curses.endwin()
126     myMotor.run(Raspi_MotorHAT.RELEASE)
127     myMotorD.run(Raspi_MotorHAT.RELEASE)
```

3.4.2 Projeto Código de Detecção de Placas De Trânsito

O projeto de desenvolvimento do algoritmo de detecção de placas, foi desenvolvido para ter baixo custo computacional e que se houve suporte para o Raspbarry Pi.

```python
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier('Cascades/placas.xml')

cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height

while True:
    ret, img = cap.read()
    img = cv2.flip(img, -1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.01,
        minNeighbors=5,
        minSize=(20, 20)
    )

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

    cv2.imshow('video',img)

    k = cv2.waitKey(30) & 0xff
    if k == 27: # press 'ESC' to quit
        break

cap.release()
cv2.destroyAllWindows()
```

4 EXPERIMENTAÇÃO E RESULTADOS

Após o término da montagem do robô, iniciaram-se os testes no robô para verificar se havia necessidade de ajustes finos no protótipo.

Figura 15 – Experimentação com o Protótipo.

Para testar a movimentação do robô e a comunicação do Raspberry PI com a placa de acionamento dos motores, foi realizado o controle remoto do protótipo via instruções-comandos de um teclado. O robô apresentou bom desempenho, indicando que o Stepper Hat escolhido funciona adequadamente.

O segundo teste foi em a utilização da câmera para identificação da placa PARE, onde o robô deve se movimentar e ao detectar a placa, o mesmo pare suas funções por um determinado período. Durante os testes, o robô parou corretamente ao encontrar as placas no campo de visão da câmera.

Outras experimentações não foram possíveis devido a pandemia e a necessidade de uso do laboratório, além de vários desafios encontrados durante a execução do projeto. O primeiro problema ocorreu no funcionamento da câmera, que não foi detectada facilmente pelo Raspberry PI. Necessitando uma ampla investigação dos drivers e de sua instalação, até conseguir fazer a mesma funcionar corretamente.

O segundo problema ocorreu no treinamento do classificador, o detector de placas Haar Cascade funcionava perfeitamente no notebook utilizado para o treinamento. Entretanto, não funcionou no Raspberry PI, snedo necessário aumentar o número de imagens positivas e negativas para o treinamento e consequentemente a geração de um classificador mais forte.

O terceiro problema ocorreu devido à baixa potência computacional do Raspberry PI, onde o algoritmo de visão computacional aplicado mesmo sendo de baixo custo computacional, provocou travamentos e gargalos no frequentes. Isto indica a necessidade de otimização do

código, sendo observado que o desenvolvimento de um novo código, ao invés de utilizar os das plataformas citadas no início do trabalho, pode não ser viável.

Algumas adversidades também provocaram alguns atrasos, como a não presença de uma bateria de alta potência, onde foi necessário energizar o protótipo com cabos de alimentação durantes todas as etapas.

5 CONCLUSÃO

Com a realização deste trabalho foi possível estudar e desenvolver um protótipo de um robô móvel autônomo de baixo custo que estivesse o mais próximo da realidade e para este fim foi utilizada o microcomputador Raspbarry Pi, com uma câmera e a linguagem de programação python juntamente com a biblioteca de visão computacional OpenCV.

Durante elaboração deste protótipo, ocorreu alguns imprevistos, como erros na programação, dificuldade no acionamento dos motores, dificuldade no treinamento para identificação de placas, erros na implementação de funções ao identificar placas, dentre outros erros.

Realizadas as correções e ajustes, os objetivos iniciais deste trabalho no desenvolvimento de um carro móvel autônomo foram atingidos. No entanto, para o pleno desenvolvimento de um robô móvel autônomo, objetivo final do trabalho, ainda faltam a aplicação da inteligência artificial para identificação das pistas para locomoção e consequente o robô realizar as funções de maneira autônoma, pontos nos quais, foram apontados como possíveis trabalhos futuros dado a já longa extensão do corrente trabalho.

5.1 TRABALHOS FUTUROS

- Instalação de sensores como de distância, para maior controle do ambiente externo.
- Implementar conceitos de computação em nuvem para implementação de novas funções visto a limitação do Raspberry Pi para processamento de imagens.
- Implementar recursos e módulos GPS.
- Desenvolvimento de um chassi mais robusto onde haverá maior controle da navegação e melhor dirigibilidade em terrenos acidentados.
- Implementar os recursos de inteligência artificial para pleno desenvolvimento de um robô móvel autônomo.

REFERÊNCIAS

[1] ROMERO, Roseli A. F. et al (2014) **Robótica Móvel**. 1ª edição, LTC.

[2] MATARIC, Maja J. (2014) **Introdução À Robótica**, Editora Blucher, 1ª Ed.

[3] TAY, T. T. et al. (2017) **Utilizing autonomous mobile robot to increase interest in STEM,** 3rd International Conference on Science in Information Technology, DOI: 10.1109/ICSITech.2017.8257103.

[4] ORTIZ, O. O. et al. (2016) **Innovative Mobile Robot Method: Improving the Learning of Programming Languages in Engineering Degrees.** IEEE TRANSACTIONS ON EDUCATION, DOI: 10.1109/TE.2016.2608779.

[5] GÓMEZ-DE-GABRIEL, J. M. et al. (2014) **Mobile Robot Lab Project to Introduce Engineering Students to Fault Diagnosis in Mechatronic Systems.** IEEE TRANSACTIONS ON EDUCATION, DOI: 10.1109/TE.2014.2358551.

[6] LINS, R. G. et al. (2015) **A Novel Machine Vision Approach Applied for Autonomous Robotics Navigation.** IEEE International Conference on Systems, Man, and Cybernetics, DOI: DOI 10.1109/SMC.2015.334.

[7] RAJASEKHAR, M. V., Jaswal, A. K. (2015) **Autonomous vehicles: The future of automobiles.** In: IEEE International Transportation Electrification Conference (ITEC), DOI: 10.1109/ITEC-India.2015.7386874.

[8] UDACITY BRASIL (2018) **Carros autônomos: entenda o funcionamento e a construção do transporte do futuro.** Udacity, disponível em <br.udacity.com/blog/post/carros-autonomos-funcionamento>, acessado em outubro de 2018.

[9] SAE INTERNATIONAL (2018) **Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles**. Norma J3016, disponível em: <www.sae.org/standards/content/j3016_201806/>, acessado em outubro de 2018.

[10]Sabbagh, V. B. (2009). **Desenvolvimento de um sistema de controle para um veículo autônomo**. Projeto Final de Curso. Belo Horizonte, Brasil. Disponível em <http://coro.cpdee.ufmg.br>, acessado em outubro.

[11] P. Y. Shinzato, T. C. Dos Santos, L. A. Rosero, D. A. Ridel, C. M. Massera, F. Alencar, M. P. Batista, A. Y. Hata, F. S. Osorio, D. F. Wolf, "**CaRINA dataset: An emerging-country urban scenario benchmark for road detection systems**", *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 41-46, nov.

[12] Makezine (2017) **Build an Autonomous R/C Car with Raspberry Pi,** disponível em: <https://makezine.com/projects/build-autonomous-rc-car-raspberry-pi/>, acessado em outubro de 2018.

[13] Wang, Zheng. (2018) **Self Driving RC Car**. Disponível em: <https://zhengludwig. wordpress.com/projects/self-driving-rc-car/> , acessado em outubro de 2018.

[14] platis.solutions (2018) **The world's first Android autonomous vehicle**, disponível em: <https://platis.solutions/blog/2015/06/29/worlds-first-android-autonomous-vehicle/>, acessado em outubro de 2018.

[15]DUCKIETOWN FOUNDATION (2018) **The Duckietown Project**, disponível em: <https://www.duckietown.org/>, acessado em outubro de 2018.

[17] hackaday.io (2018) **A Self-Driving Car using a Raspberry Pi Zero**, disponível em: <https://hackaday.io/project/27173-a-self-driving-car-using-a-raspberry-pi-zero>, acessado em outubro de 2018.

[18] GitHub (2018) **Self-driving RC car using OpenCV and Keras**, diponível em: <https://github.com/pseudoyim/galvaneye>, acessado em outubro de 2018.

[19]ELEC/CENG 499 (2018) **IMP AUTONOMOUS VEHICLES INC**, disponível em: <http://www.ece.uvic.ca/~tyu/>, acessado em outubro de 2018.

[20] GitHub (2018) **Autonomous RC Car using Neural Networks, Python and Open CV**, disponível em: <https://github.com/sidroopdaska/SelfDrivingRCCar>, acessado em outubro de 2018.

[21] GitHub (2018) **Autonomous RC Car**, disponível em: <https://github.com/beytullahoglu /AutonomousRCCar>, acessado em outubro de 2018.

[22] Fish Eyes Câmera Raspberry Pi, disponível em:< https://pt.aliexpress.com/item/32897842264.html?spm=a2g0s.9042311.0.0.2742b90aZDLBQT> , acessado em junho de 2020.

[23] Driver de Motor UGEEK modelo Stepper Motor HAT v0.2, disponível em: < https://pt.aliexpress.com/item/32536369104.html>, acessado em junho de 2020.

[24] Bradski, Gary R. e Vadim Pisarevsky. "Biblioteca de visão computacional da Intel:

aplicativos em calibração, segmentação estéreo, rastreamento, gesto, reconhecimento de rosto

e objeto." Proceedings IEEE Conference on Computer Vision and Pattern Recognition.

CVPR 2000 (Cat. No. PR00662) . Vol. 2. IEEE, 2000.

[25] G. Bradski e A. Kaehler, Learning OpenCV: Comput. Aprendendo OpenCV: Visão computacional com a biblioteca OpenCV, Sebastopol, CA, EUA: O'Reilly Media, 2008.

[26] LEVADA, A. L. M.; VALENTE, F. J.; RIBEIRO, M. X.. Explorando o algoritmo de Viola-Jones na detecção e reconhecimento facial. 2018. Trabalho de Conclusão de Curso (Graduação em Bacharelado em Ciência da Computação) - Universidade Federal de São Carlos.

[27] PEREIRA, Rafael Cardoso. Técnica de rastreamento e perseguição de alvo utilizando o algoritmo Haar cascade aplicada a robôs terrestres com restrições de movimento. 2017. 104f. Dissertação (Mestrado em Engenharia Mecatrônica) - Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal, 2017.

[28] PAES, Alexandre Santos Lima. Reconhecimento automático de placas automobilísticas. 2017. Projeto de Graduação submetido ao curso de Engenharia Eletrônica e de Computação. – Universidade Federal do Rio de Janeiro.

[29] VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, v. 1, p. I–I, 2001.

[30] Freund, Yoav e Robert E. Schapire. "Uma generalização da teoria da decisão da aprendizagem on-line e uma aplicação para impulsionar." Journal of computer and system sciences 55.1 (1997): 119-139.

[31] - Y. Freund and R. Schapire, “A desicion-theoretic generalization of online learning and an application to boosting,” Computational learning theory, vol. 55, pp. 119–139, Aug. 1995.

[32] - P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, no. C, pp. I–511–I–518, 2001.

[33] - VIOLA, P.; JONES, M. J. Robust real-time face detection. International Journal of Computer Vision, Springer, v. 57, n. 2, p. 137–154, 2004.

[34] - “Python: O que é? Por que usar?”, disponível em: <http://pyscience-brasil.wikidot.com/python:python-oq-e-pq>, acessado em julho de 2020.

[35] - “Visão Computacional – O que é e qual a sua importância?”, disponível em: < https://www.sas.com/pt_br/insights/analytics/computer-vision.html >, acessado em julho de 2020.