UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CAMPUS DE ILHA SOLTEIRA

VINÍCIUS MONZEM GARCIA

DETECÇÃO E O RASTREAMENTO DE IMAGEM DE OBJETO DE COR USANDO OPENCV E A PLATAFORMA RASPBERRY PI VISANDO A APLICAÇÃO EM ROBÔS MÓVEIS.

Ilha Solteira
2023

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

VINÍCIUS MONZEM GARCIA

DETECÇÃO E O RASTREAMENTO DE IMAGEM DE OBJETO DE COR USANDO OPENCV E A PLATAFORMA RASPBERRY PI VISANDO A APLICAÇÃO EM ROBÔS MÓVEIS

Trabalho de Conclusão de Curso, apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Profa. Dra. Suely Cunha Amaro Mantovani
Orientadora

Ilha Solteira
2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Garcia, Vinícius Monzem.
G216d Detecção e o rastreamento de imagem de objeto de cor usando OpenCV e a plataforma Raspberry Pi visando a aplicação em robôs móveis / Vinícius Monzem Garcia. -- Ilha Solteira: [s.n.], 2023
80 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Suely Cunha Amaro Mantovani

Inclui bibliografia

1. Processamento digital de imagens. 2. Visão computacional. 3. OpenCv. 4. Navegação autônoma de robôs móveis. 5. Sensores ultrassônicos. 6. Hardware embarcado.

Amanda Sertori dos Santos
Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos dezoito dias do mês de dezembro do ano de dois mil e vinte e três, o discente **VINÍCIUS MONZEM GARCIA,** matriculado sob o RA: 172055521, tendo como banca examinadora sua orientadora a Profa. Dra. Suely Cunha Amaro Mantovani, o Prof. Dr. Jozué Vieira Filho e o Dr. Júlio Cesar Eduardo de Souza, apresentou o Trabalho de Graduação intitulado **DETECÇÃO E O RASTREAMENTO DE IMAGEM DE OBJETO DE COR USANDO OPENCV E A PLATAFORMA RASPBERRY PI VISANDO A APLICAÇÃO EM ROBÔS MÓVEIS**, obtendo a nota 10 e o conceito aprovado

Vinícius Monzem Garcia
- Discente -

Profa. Dra. Suely Cunha Amaro Mantovani
- Orientadora -

Jozue Vieira Filho

Assinado de forma digital por Jozue Vieira Filho
Dados: 2023.12.20 09:18:49 -03'00'

Prof. Dr. Jozué Vieira Filho
UNESP- Campus - São João da Boa Vista
- Membro da banca -

Dr. Júlio Cesar Eduardo de Souza
Pós-doutorando-DEE
- Membro da banca -

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão à minha professora Suely e à minha mãe, Milena, por todo o apoio e orientação que me deram durante minha jornada acadêmica. Desde o início da minha Iniciação Científica até a conclusão da apresentação deste trabalho, elas estiveram ao meu lado, oferecendo suporte e encorajamento. Sem a ajuda delas, eu não teria conseguido chegar até aqui.

RESUMO

A detecção e o rastreamento de objetos são características de grande importância em sistemas de visão computacional com aplicações em áreas tais como, médica, segurança, robótica detecção de falhas etc. Neste trabalho tem-se como proposta realizar a detecção e o rastreamento de objeto de determinada cor em tempo real, usando a biblioteca Open Source Computer Vision (OpenCV), aplicado à navegação de robôs móveis. A OpenCV é uma biblioteca específica para tratamento de imagens, própria para aplicação em sistemas em tempo real onde é necessário utilizar ferramentas com um tempo de resposta baixo. O algoritmo de detecção e rastreamento é otimizado prevendo ser embarcado em uma plataforma Single Board Computer, Raspberry Pi, em um robô móvel. Para isso, o protótipo do robô móvel conta com uma câmera tipo webcam para a detecção do objeto e sensores ultrassônicos que juntos auxiliam em sua navegação de modo autônomo pelo ambiente. A imagem adquirida pela webcam é convertida para o sistema de cores Hue, Saturation,Value, usando as técnicas de processamento digital de imagens da biblioteca OpenCV. Por fim, faz-se a montagem do hardware e software no robô visando os testes finais, obtendo como resultado o rastreamento do objeto de cor pelo robô, e o seu deslocamento no ambiente, sem bater nos obstáculos.

Palavras-chave: Processamento digital de imagens; visão computacional; navegação autônoma de robôs móveis; sensores ultrassônicos; hardware embarcado.

ABSTRACT

Object detection and a tracking are features of great importance in computer vision systems with applications in areas such as medical, security, robotics, fault detection, etc. In this work, it is proposed to perform the detection and tracking of objects of a certain color in the image, in real time, using the Open-Source Computer Vision (OpenCV) library, applied to navigation of mobile robots. OpenCV is a specific library for image processing, suitable for application in real-time systems where it is necessary to use tools with a low response time. The detection and tracking algorithm is optimized in anticipation of being embedded in a Single Board Computer platform, Raspberry Pi, in a mobile robot. For this, the prototype of the mobile robot it has a webcam-type camera for the detection of the object and ultrasonic sensors that together help in its autonomous navigation through the environment. The image acquired by the webcam is converted to the Hue, Saturation, Value color system, using the techniques of digital image processing from the OpenCV library. Finally, the hardware and software are assembled in the robot, aiming at the tests for the tracking of the color object by the robot and obtaining as a result its displacement and navigation in the environment, without hitting obstacles.

Keywords: Digital image processing; computer vision; autonomous navigation of mobile robots; ultrasonic sensors; embedded hardware.

ÍNDICE DE FIGURAS

Figura 1 – Relação em termos da informação entre o processamento de imagem, computação gráfica e a visão computacional 19
Figura 2 – Sistema de Processamento de imagens e Visão computacional. 20
Figura 3 – Espectro de luz visível pelo olho humano.....22
Figura 4 – Cubo tridimensional do sistema RGB. 23
Figura 5 – Diagrama do espaço de cores HSV. 24
Figura 6 – Conversão da cor laranja entre os espaços de cores RGB para HSV..... 27
Figura 7 – Classificação dos robôs. a) Manipuladores. b) Robôs móveis. c) Híbrido. d) Robôs móveis com pernas. e) Robôs móveis autônomos em fábricas..... 32
Figura 8 – Plataforma Raspberry P 3 Model B+ ANATEL 35
Figura 9 – Plataforma Arduino UNO R3. 37
Figura 10 – (a) Webcam. (b) Webcam e cabo flat..... 40
Figura 11 – Sensor ultrassônico US – 100. (a) Configuração dos pinos. (b) Verso..... 40
Figura 12 – Temporização dos sinais de TRIGGER e ECHO para o US-100. 41
Figura 13 – Módulo driver ponte H L298N. 42
Figura 14 – Circuito esquemático da ponte H L298N. 43
Figura 15 – Circuito interno para o CI L298. 44
Figura 16 – Motor CC MICRORED modelo P. 45
Figura 17 – Bateria de LiPo de 11,1 V, 2200 mAh.....45
Figura 18 – Módulo regulador de tensão LM2596. 46
Figura 19 – Circuito interno do módulo LM2596. 46
Figura 20 – Diagrama de blocos do processamento de imagem. 48
Figura 21 – (a) Bola anti-stress. (b) Garrafa térmica.....49
Figura 22 – Inicialização da câmera e leitura do frame. 50
Figura 23 – Comparando espaços de cores. (a) RBG. (b) HSV. 50
Figura 24 – Espaço e limite de cores no espaço HSV. 51
Figura 25 – Cálculo dos limites com barras - guia da cor azul. 51
Figura 26 – Bola anti-stress: (a) antes da aplicação da função inRange().(b) depois da aplicação da função inRange() 52
Figura 27 – Erosão e dilatação do frame. 52
Figura 28 – Processo de abertura seguido do processo de fechamento. 53

Figura 29 – Detecção dos contornos e posição do objeto. 53
Figura 30 – Abordagem visual das coordenadas x e y, largura e altura. 54
Figura 31 – Cálculo do centroide..... 55
Figura 32 – Código do cálculo do centro geométrico do objeto. 55
Figura 33 – Detecção e rastreamento da bola anti-stress. 56
Figura 34 – Detecção e rastreamento da garrafa térmica. 56
Figura 35 – Diagrama de blocos do hardware.....58
Figura 36 – O protótipo do robô. (a) Vista superior. (b) Vista inferior (2 motores e 2 baterias)..... 59
Figura 37 – Esquema de ligação do circuito. 60
Figura 38 – Fluxograma da comunicação serial entre Raspberry Pi e Arduino. 62
Figura 39 – Representação 3D do robô acoplado com a câmera..... 63
Figura 40 – Representação do alcance visual da câmera 63
Figura 41 – Redução do alcance visual da câmera e adição do objeto 64
Figura 42 – Representação da área de segurança do objeto. 64
Figura 43 – Objeto dentro da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D. 65
Figura 44 – Objeto perto do limite da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D. 65
Figura 45 – Objeto fora da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D. 65
Figura 46 – Mapeamento do ambiente..... 66
Figura 47 – Visualização superior do ambiente mapeado..... 67
Figura 48 – Visualização frontal do ambiente mapeado..... 68
Figura 49 – Testes dos sensores ultrassônicos do robô..... 69
Figura 50 – Detecção e rastreamento sob a visão do robô..... 70

ÍNDICE DE TABELAS E QUADROS

Tabela 1 – Especificações do Raspberry Pi 3 Model B+ Anatel 36
Tabela 2 – Características do Arduino UNO R3 37
Tabela 3 – Características do módulo OV5647 39
Tabela 4 – Especificações do US-100 41
Tabela 5 – Especificações técnicas do módulo driver ponte H L298N 42
Tabela 6 – Lógica de controle para o CI L298 44
Tabela 7 – Características do módulo regulador de tensão LM2596 46
Tabela 8 – Tabela verdade com as saídas dos sensores e a ação realizada pelo robô 61
Tabela 9 – Sintetização das ações do robô 69

Quadro 1 - Principais funções OpenCV 30
Quadro 2 - Conexões sob o controle do Arduino 78

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Objetivos 13
1.2 Organização do Texto 13
2 REVISÃO BIBLIOGRÁFICA 15
3 PERCEPÇÃO DE MÁQUINA 18
3.1 Visão Computacional 18
3.2 Imagem Digital 21
3.2.1 Espaço e modelo de cores 22
3.2.1.1 Sistema de cores RGB 23
3.2.1.2 Espaço de cores HSV 24
3.2.2 Conversão entre os espaços de cores. 25
3.3 Processamento de Imagem Digital e a Detecção e o Rastreamento de Objetos 28
3.4 Software de Suporte (OPENCV) 29
4 PRINCIPAIS CONCEITOS DA ROBÓTICA 31
4.1 A Robótica: definição e classificação 30
4.2 Percepção e ação na robótica 33
5 MATERIAIS UTILIZADOS E CARACTERÍSTICAS 35
5.1 Plataformas de desenvolvimento 35
5.2 Sensores 38
5.2.1 Webcam 38
5.2.2 Sensor ultrassônico 39
5.3 Acionamento do Motor CC 41
5.3.1 Módulo driver ponte H L298N 42
5.3.2 Motor CC 44
5.4 Fonte de Alimentação 45
6 DESENVOLVIMENTO DO ALGORITMO DE DETECÇÃO E RASTREAMENTO 48
6.1 Diagrama de Blocos do Algoritmo 48
6.2 O Algoritmo de Detecção e Rastreamento e os resultados 49
6.2.1 Transformação da imagem e o reconhecimento de cores 49
6.2.2 Morfologia matemática (erosão e dilatação) 52
6.2.3 Cálculo do centroide e detecção dos contornos 53

7 DESENVOLVIMENTO DO ROBÔ DOTADO DE VISÃO ARTIFICIAL 58
7.1 Diagrama de blocos do hardware 58
7.2 O protótipo 58
7.3 Esquema de ligação do circuito 59
7.4 Comunicação serial entre Arduino e Raspberry Pi 61
7.5 Mapeamento do ambiente 66
7.6 Resultados dos testes com o protótipo 69
8 CONCLUSÕES 72
9 REFERÊNCIAS 74
APÊNDICE A – Conexões ao ARDUINO 78
APÊNDICE B – Resumo publicado no XXXII Congresso de Iniciação Científica da UNESP - CIC 2020 79

1 INTRODUÇÃO

Dentro da teoria de Visão Computacional o rastreamento de objetos é uma importante característica com aplicação em diversas áreas, tais como na interação homem-computador-automática, monitoramento de tráfego de veículos, monitoramento automático de câmeras de segurança, na robótica e veículos autônomos, entres outras.

O aumento do interesse em algoritmos de rastreamento advém, principalmente, das disponibilidades do alto poder computacional e de vídeos de câmeras (ou webcams) de alta qualidade e de baixo custo. A análise de vídeo é realizada pela detecção de objetos de interesse em movimento, o seu rastreamento no quadro e a análise do objeto rastreado, é pela cor e/ou forma, visando o seu reconhecimento (YILMAZ; JAVED; SHAH, 2006; GONZALEZ; WOODS, 2009)

A detecção e o rastreamento de objetos utilizados em robótica e na navegação de veículos autônomos fornece a um robô (ou veículo) a possibilidade da visão humana, este "aprender a enxergar" e, com isso, fazer tarefas cada vez mais complexas e úteis ao dia a dia das pessoas como, reconhecer objetos, obter suas características, determinar seus movimentos e velocidade, identificar pessoas e/ou faces etc. Um robô móvel pode tomar decisões e navegar de modo autônomo pela percepção de um meio externo usando o processamento de imagens.

Com a crescente evolução da tecnologia, as pesquisas em robóticas têm avançado criando robôs robustos, autônomos e confiáveis que vem se tornando elementos fundamentais em setores das atividades humanas que ofereçam algum risco ou que necessitem de autonomia, confiabilidade e precisão. Nos últimos anos robôs autônomos têm sido usados em diversas áreas do conhecimento, como na medicina - robôs cirúrgicos (WARELINE, 2018) e na agricultura - robôs que detectam pragas (ESTADÃO, 2020). A robótica na agricultura está se desenvolvendo graças ao aumento dos startups direcionadas para esta área que estão se empenhando em criar modelos eficientes de robôs para auxiliar na produção de alimentos desde o plantio até o controle de pragas.

Encontram-se na literatura vários trabalhos com aplicação de rastreamento de imagem e aplicados a navegação robótica (ou veicular) como o de Heinen e Osório (2002), Osório e Bordignon (2014), Oliveira, Oliveira e Moreira (2014), Oliveira *et al* (2014), Schueroff (2013), que demonstram o crescente interesse dos pesquisadores e da comunidade acadêmica pelo tema, proporcionando o desenvolvimento de veículos autônomos e confiáveis.

Neste trabalho tem-se como objetivo implementar um algoritmo de detecção e o rastreamento de um objeto de determinada cor em vídeo, usando a biblioteca OpenCV (2019) e uma placa Single Board Computer (SBC), Raspberry Pi, aplicados a navegação autônoma de um protótipo de um robô móvel.

1.1 Objetivos

Tem-se como proposta neste trabalho realizar a detecção e o rastreamento de objetos de determinada cor, em vídeo, em tempo real, por meio de uma câmera tipo webcam e a biblioteca Open Source Computer Vision (OpenCV), visando a sua utilização na navegação de robôs móveis. O algoritmo de detecção e rastreamento é otimizado prevendo ser embarcado em uma placa SBC, Raspberry Pi em um protótipo de um robô móvel, viabilizando os testes práticos finais. O robô deve ser capaz de identificar e seguir um objeto de determinada cor em sua navegação pelo ambiente.

1.2 Organização do Texto

Além desta introdução o texto está organizado de forma que, no capítulo 2 têm-se a revisão bibliográfica apresentando alguns trabalhos encontrados na literatura que forneceram os conhecimentos necessários para o desenvolvimento desta pesquisa, mostrando a Visão Computacional e o Processamento Digital de Imagem aplicado à robótica.

São abordados no capítulo 3 os conceitos de percepção de máquina e visão computacional, apresentando como as máquinas interpretam e compreendem imagens e vídeos. São discutidos conceitos como processamento digital de imagens, visão artificial, visão de máquina e visão robótica, além de modelos e espaços de cores, como Red, Green e Blue (RGB) e Hue, Saturation e Value (HSV).

No capítulo 4 apresentam-se os conceitos da robótica que incluem as definições e classificação, a percepção do robô ao seu redor por meio de sensores e a ação usando os atuadores que torna possível a sua navegação.

Descrevem-se no capítulo 5 os materiais, componentes do hardware, utilizados neste projeto e as suas principais características e funcionamento fornecendo a fundamentação teórica necessária para o entendimento do trabalho desenvolvido.

O desenvolvimento do algoritmo de detecção e rastreamento do objeto de determinada cor é descrito no capítulo 6.

No capítulo 7 são apresentados o protótipo do robô e o hardware desenvolvido e embarcado, dotado de visão artificial, e os resultados dos testes de rastreamento e navegação realizados.

Por fim, as conclusões do trabalho e sugestões para trabalhos futuros estão no capítulo 8 seguido das referências bibliográficas e os apêndices, com o resumo do trabalho apresentado no Congresso de Iniciação Científica da UNESP-2020.

2 REVISÃO BIBLIOGRÁFICA

Tem-se conhecimento de que a constante evolução das tecnologias na área de visão computacional impulsiona projetos e soluções, direcionadas para a automação em geral, visando não somente o reconhecimento e o consumo de informações obtidas do ambiente, mas também para a economia de recursos como mão de obra e materiais. Nesta revisão bibliográfica são apresentados alguns artigos na área de visão computacional aplicada à robótica que nortearam o desenvolvimento deste trabalho, tais como Schueroff (2013), Silva (2014), Oliveira, Oliveira e Moreira (2014) e Braga e Costa (2015).

Em Schueroff (2013) é descrito o desenvolvimento de um sistema de visão artificial para um robô móvel explorador abordando o estado da arte sobre o processamento digital de imagens bem como a robótica móvel com ênfase em navegação baseada em visão artificial. Em seu trabalho é introduzido o conceito de arquiteturas deliberativas, reativas e híbridas, caracterizadas por descrever o funcionamento distinto de sistemas. Para isso, criou dois sistemas, em que um comanda, módulo deliberativo e o outro é o módulo reativo que, por meio do uso de sensores e atuadores traduzem as informações do ambiente. Utiliza os softwares Robotic Operation System (ROS) e Open Source Computer Vision (OpenCV) e componentes como sensores ultrassônicos, uma câmera, dois controladores — o Raspberry Pi modelo B e o Freescale FRDM-KL25Z. Para obter os resultados realizou três testes funcionais de processamento de imagem sendo o primeiro com o objetivo de encontrar um quadrado de cor amarela em um cubo mágico, o segundo com o intuito de encontrar objetos circulares dentre outros formatos e o terceiro mesclava o primeiro e o segundo experimento buscando objeto de cor de formato específico. Ao final relata que o robô identificou o objeto em diferentes situações e se posicionou em direção ao objeto mapeado.

É apresentado em Silva (2014) um sistema desenvolvido que recebe como entrada, imagens gravadas de uma câmera um dispositivo móvel (*smarphone),* utilizando processamento digital de imagens para calcular a quantidade total de telhas presente no vídeo. Com este projeto, o autor esperava padronizar os processos de contagem de telhas que ainda eram realizados de maneira manual, aumentando a capacidade de distribuição das cerâmicas, diminuindo o tempo do processo e a margem de erros durante a contagem. Usa Redes Neurais Artificiais (RNAs) que aprende por treinamento, um determinado padrão capaz de identificar posteriormente as telhas. A aplicação foi implementada na plataforma Android (Java) com o uso da biblioteca OpenCV, com a captura das imagens feita por uma câmera de um smartphone

Samsung Galaxy S III Duos. Relata que obteve uma taxa de acerto de 98,1%, mas durante os testes ocorreram alguns erros de segmentação entre a quantidade de telhas que o algoritmo detectava e a quantidade real de telhas. Atribui esses erros, principalmente, às fissuras nas telhas e/ou às suas proximidades umas das outras, pois o algoritmo não conseguia fazer a segmentação delas.

É apresentado em Oliveira, Oliveira e Moreira (2014) o desenvolvimento e a construção de um robô móvel com o uso da plataforma Raspberry Pi integrado a um sistema de visão computacional, com o objetivo de permitir que um robô interaja com o ambiente, utilizando uma webcam para a captura de imagens em tempo real, garantindo a movimentação do robô com base no reconhecimento da cor de objetos. Definem três módulos que norteiam a abordagem de seu projeto, o reativo, central e o receptivo. O Módulo central caracteriza-se por ser a parte do sistema em que se encontra o Raspberry Pi, a câmera, os processos de tratamento da imagem adquirida, e o transmissor de informações baseada na linguagem de programação Python, juntamente com a biblioteca OpenCV, para realizar a segmentação da cor do objeto de interesse, obtendo informações como tamanho e coordenadas do objeto. No módulo reativo têm os atuadores que fazem a integração e movimentação do robô no ambiente externo, recebe as informações do módulo central através da interface de hardware General Purpose Input Output (GPIO) e realiza as ações necessárias para garantir que o objeto esteja presente na área de captura da imagem. Quanto ao módulo receptivo é responsável pela visualização do usuário, verifica, em tempo real, as ações do robô, bem como consome informações disponibilizadas através de vídeo, pelo módulo central. Dos resultados obtidos foi observado que o sistema implementado é satisfatório, mas ressaltam que a baixa taxa de *frames* por segundo, pode se tornar um problema quando for necessário um processamento mais robusto com o uso de RNAs, por exemplo, exigindo opções de plataformas com maior poder computacional.

Em Braga e Costa (2015) é apresentado o desenvolvimento de um sistema aplicado a um robô móvel que realiza a detecção, localização e o rastreamento facial. Para isso, usa a plataforma Arduino Uno R3 como placa de prototipagem, Motor Shield (L293D) responsável pelo acionamento dos motores do robô; três baterias, uma de 9V para alimentar o Arduino e duas baterias de 3,7V, ligadas em série, para alimentar o Motor Shield; um regulador de tensão, LM2596 para ajustar a tensão de 7,2V para 5V para alimentação de uma Câmera IP DLink DCS - 930L; um módulo Bluetooth HC-06 e um sensor ultrassônico. Foi usada a linguagem Java em conjunto com a biblioteca OpenCV, no tratamento da imagem, considerando o baixo poder de processamento de imagens do Arduino. Durante a fase de testes o autor encontrou um problema

em relação ao recebimento de dados pela porta serial do Arduino, quanto a grande quantidade de informações enviadas da aplicação em Java e como o Arduino recebia as informações de forma sequencial, o robô acabava se perdendo por executar muitas vezes o mesmo comando. Para resolver este problema foi implementado um sistema de vetores de cinco posições, em que todas as informações eram lidas de uma só vez, em relação à posição da informação na matriz e não mais eram enviadas sequencialmente. Conclui que obteve bons resultados, com uma boa iluminação e estabilidade de superfície, além de fazer o teste com fotografias e posteriormente com faces reais, o robô atendeu as expectativas e fez a localização e rastreamento das faces.

Baseada nesta revisão bibliográfica obteve-se os conhecimentos necessários para orientar às escolhas das tecnologias usadas, como a utilização de duas plataformas, uma para o processamento da imagem como o Raspberry Pi e a Arduino para o controle dos sensores e motores; a linguagem de programação Python, em conjunto com a biblioteca OpenCV e os sensores e atuadores visando a aplicação ao módulo de visão computacional proposto.

3 PERCEPÇÃO DE MÁQUINA

Neste capítulo faz-se uma breve introdução aos conceitos envolvidos em sistemas que se baseiam em visão computacional, uma alternativa eficiente na questão da análise de dados, que proporciona a percepção visual de uma máquina, seja um robô ou qualquer outro dispositivo que seja dotado de visão. Inicia-se pela Visão Computacional, seguido do Processamento Digital de Imagens e suas aplicações.

3.1 Visão Computacional

A Visão computacional pode ser entendida como sendo a construção das descrições explicitas e significativas de objetos físicos, a partir de imagens, ou seja, é o estudo da extração de informação de uma imagem. Envolve principalmente, a segmentação, a classificação, reconhecimento de padrão e o rastreamento (BALLARD; BROWN, 1982).

Utiliza-se de computadores para emular a visão humana, incluindo o aprendizado e a capacidade de fazer interferências, agindo com base em informações visuais. Em se tratando do tópico de Visão Computacional têm-se a visão artificial, visão de máquina e visão robótica que são áreas relacionadas que utilizam técnicas computacionais para permitir que as máquinas possam interpretar e compreender imagens e vídeos.

Visão Artificial é a área mais ampla e genérica que engloba outras áreas da Visão Computacional como Visão de Máquina e Visão Robótica. O objetivo da Visão Artificial é permitir, por meio de técnicas e métodos, que as máquinas possam enxergar e entender o mundo visual de forma similar aos seres humanos, possibilitando a automação de tarefas que requerem interpretação visual (SZELISKI, 2010).

A Visão de Máquina concentra-se especificamente em projetar algoritmos e técnicas de processamento de imagens para permitir que máquinas possam interpretar e entender informações visuais de forma eficiente. Essa área tem como objetivo extrair informações úteis de imagens e vídeos, como reconhecimento de objetos, segmentação de imagem, detecção de bordas e outras técnicas de processamento de imagem. Tem como área de aplicação reconhecimento de padrões, diagnósticos médicos, detecção de falhas em sistemas industriais e outras (FORSYTH; PONCE, 2002).

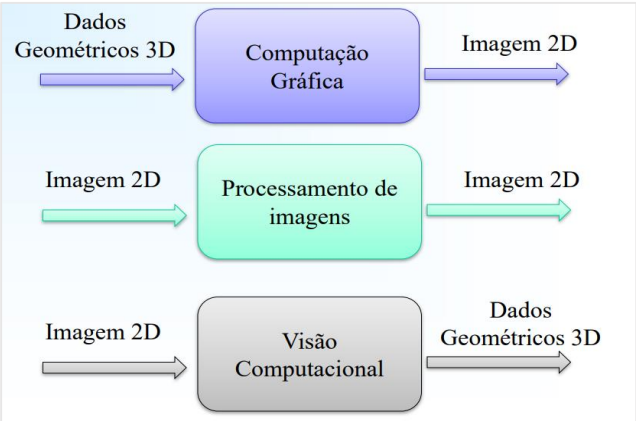
A Visão Robótica é a Visão de Máquina na robótica. Envolve o uso de técnicas de processamento de imagem para que os robôs possam identificar objetos, navegar em ambientes desconhecidos, realizar tarefas de manipulação, interagir com seres humanos, além de executar

outras tarefas que requerem percepção visual. Sua área de aplicação pode incluir robôs móveis, braços robóticos, veículos autônomos etc. (SICILIANO; KHATIB; KRÖGER, 2008; MASUTANI *et al*, 2013).

A visão de máquina e a visão robótica parecem muito semelhantes, e realmente são, uma vez que ambas são subconjuntos da Visão Artificial, mas a principal diferença é a área de aplicação e o escopo das técnicas utilizadas, ambas as áreas têm em comum o uso de técnicas de processamento de imagem, mas a visão de máquina tende a se concentrar em soluções específicas para problemas de processamento de imagem, enquanto a Visão Robótica lida com problemas mais amplos de percepção e interação em ambientes físicos, com foco na aplicação de robôs.

A cultura humana tem usado imagens como meios de expressão desde as pinturas pré-históricas, milênios antes do aparecimento da palavra escrita. A necessidade de processar imagens surgiu no início do século passado, em 1921, quando as primeiras imagens digitais foram geradas. O entendimento de imagem é muito diferente do seu processamento, o qual estuda o chamado *image-to-image transformation*, que significa a transformação de uma imagem de entrada em uma imagem desejada de saída, essa relação entre os dados de entrada e saída e a sua dimensão, é ilustrada na Figura 1. (VIEIRA; GONZAGA, 2020).

Figura 1 - Relação dos dados de entrada/saída entre o processamento de imagem, computação gráfica e a visão computacional

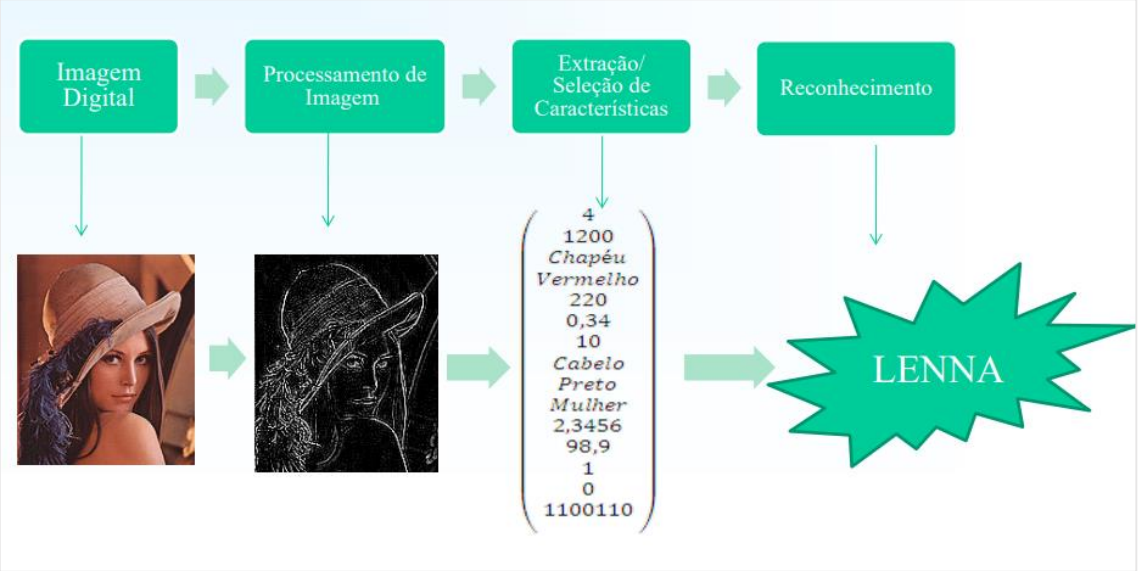


Fonte: Vieira e Gonzaga, 2020.

Visão computacional estuda como uma máquina, por exemplo, um computador, enxerga o meio à sua volta, extraindo informações úteis a partir de câmeras de vídeo, sensores e outros dispositivos. O computador é capaz de realizar tarefas inteligentes a partir dessas informações, aproximando-se cada vez mais da inteligência humana. Apresenta-se na Figura 2 uma estrutura

clássica de um sistema de Processamento de imagens e Visão computacional, usando como base icônica a imagem Lenna.

Figura 2 - Sistema de Processamento de imagens e Visão computacional.



Fonte: Vieira e Gonzaga, 2020.

Sabe-se também que o Processamento de Imagens, Reconhecimento de Padrões, Computação Gráfica, Inteligência Artificial, Visão Biológica e Psicologia, essa com relação à percepção de mundo, são áreas que fornecem conhecimentos à Visão Computacional, a qual abrange três níveis específicos:

- Visão em baixo-nível - referente ao pré-processamento da imagem que usa filtros, realces, restaurações e outros tratamentos à imagem;
- Visão em nível-intermediário - refere-se à extração de características em que ocorre a segmentação e a descrição de um aspecto ainda mais específico da imagem;
- Visão em alto-nível - implica no uso do conhecimento intrínseco para fazer interferências, tais como a análise e modificação da imagem conforme desejo do usuário, uso de inteligência artificial e outras aplicações. (VIEIRA; GONZAGA, 2020).

Processar uma imagem significa modificar as informações contidas nessa imagem sob vários aspectos, de modo que o resultado seja uma imagem ou informações extraídas dela. Um grande número de aplicações envolve o processamento de imagens, gráficos computacionais e a própria Visão Computacional. Alguns exemplos de aplicações de Processamento de Imagens Digitais são imagens baseadas em radiação, compreendendo o espectro eletromagnético; imagens baseadas em energias acústicas como ultrassônicas e eletrônicas e imagens sintéticas

utilizadas para a modelagem e visualização, essas geradas pelo computador. Encontram-se aplicações na área médica com o uso de Sistemas de Auxílio ao Diagnóstico. Aplicações na área da indústria como inspeção de dimensões, inspeção de modelos e padrões, bem como a classificação desses modelos, controle de qualidade por inspeção visual, como por exemplo, latas de cerveja, frutas e afins, robôs guiados por visão em linha de montagem.

Encontra-se aplicações na área da biometria na identificação e segurança por imagem da íris, câmeras de segurança com detecção de movimentos, gestos suspeitos, *watermaking* em que há a proteção e identificação de *copyright*. As áreas de abrangência e aplicação da Visão Computacional são extremamente relevantes para o contexto atual e futuro, com tendência de crescimento exponencial (VIEIRA; GONZAGA, 2020).

3.2 Imagem Digital

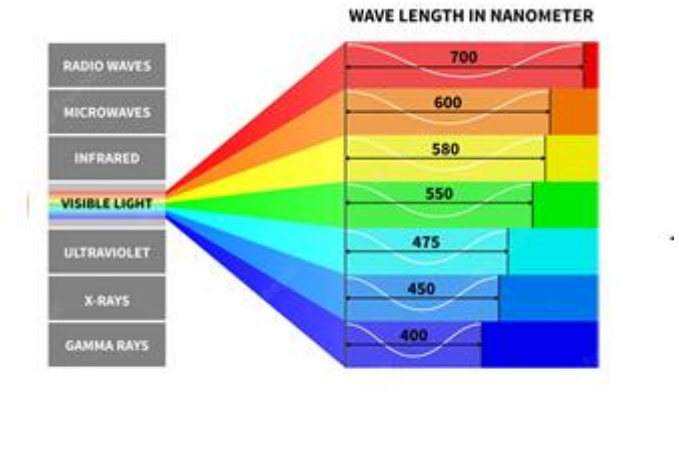
Uma imagem digital contém representações visuais de objetos por meio de alguma técnica ou processo. No ambiente computacional as imagens são representadas digitalmente por uma matriz bidirecional de pontos f(x,y) de bits ou binário (0,1) processados e transformados em representações conhecidas como *pixel* (menor medida de uma imagem). O *pixel*, normalmente um quadrado, é a unidade elementar de brilho e cor que constitui uma imagem digital, as junções de todos os pontos formam uma imagem. Na computação existem dois tipos destas representações, as imagens vetoriais e matriciais.

As imagens vetoriais são uma descrição dos dados na forma de um espaço contínuo, conhecido como descrição vetorial. Ao contrário, as imagens matriciais ou mapa de bits são imagens que contêm a descrição em uma escala de cor de cada ponto ou pixel. Correspondem a qualquer valor em uma escala de cores RGB. Em uma imagem com escala de 8 bits, por exemplo, o valor do pixel está entre 0 (preto) e 255 (branco) e o entremeio, a mistura das cores que representa uma imagem em tons de cinza.

Cor é luz e a luz percorre o espaço como uma forma de onda que se transforma em energia – radiação eletromagnética para cada cumprimento de onda de luz visível é associado a percepção de uma cor.

Distribuição espectral de potência - o espectro de luz visível pelo olho humano é delimitado a faixa de radiação infravermelha (baixa frequência) até a faixa de radiação ultravioleta (alta frequência), ilustrada na Figura 3.

Figura 3 – Espectro de luz visível pelo olho humano.



Fonte: Adaptado de-Helerbrock, 2023

A resolução de uma imagem é a quantidade de informação que a imagem contém por unidade de comprimento, isto é, o número de pixels por polegada, *pixels per inch* (ppi). Quanto maior a resolução de uma imagem maior será o tamanho do arquivo de armazenamento. O nível de detalhe de uma imagem depende da informação de cada pixel. Cada pixel é codificado de acordo com a cor e o brilho que representa, isto é, ocupa em memória um número de bits que varia de acordo com o número de cores, tons de cinza e brilho definido para uma determinada imagem (ALFAMÍDIA, 2019).

3.1.1 Espaço e modelo de cores

Um espaço de cores é uma representação matemática de um conjunto de cores. Existem vários espaços de cores, tais como o modelo RGB utilizado em computadores, o modelo YIQ (sendo "Y" a luminância ou brilho, "I" sendo *in-phase*, em fase, e "Q" sendo *quadrature*, quadratura), YUV ou YCbCr[1] usado em sistemas de vídeo e CMYK (sendo "C" a componente de cor ciana (Cyan), "M" a componente magenta da cor, "Y" a componente amarela da cor e "K" a componente preta da cor), utilizado em impressoras coloridas (GONZALEZ; WOODS, 2009). Como nenhum destes modelos de cores está relacionado com a noção mais intuitiva de tom, saturação e brilho, foi criado para a programação, o processamento e a manipulação destes

[1] YCrCb - O modelo YCrCb usa uma componente de luminância (Y) e duas componentes de crominância (Cr e Cb) e tem sido o modelo mais recomendado para realizar o reconhecimento de cor de pele.

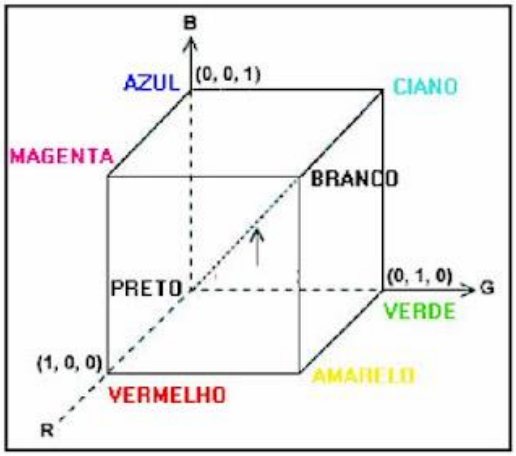
sistemas de cores modelos, como Tom, Saturação e Intensidade ou do inglês *Hue, Saturarion* e *Intensity* (HSI) e Tom, Saturação e Valor ou do inglês *Hue, Saturation* e *Value* (HSV).

O espaço de cores também definido como uma faixa definida de cores. Com relação a como os olhos humanos interpretam as cores comparadas com um computador, pode ser entendido como sendo uma organização específica de cores, que combinada com o perfil de funcionamento de um dispositivo físico (a câmera, o monitor, celular ou uma impressora), permite representações reprodutíveis da cor, tanto de forma analógica quanto digital (CAVALCANTI, 2020).

3.1.1.1 Sistema de cores RGB

Dentro do espaço de cores tem-se o modelo RGB que é específico quando definido pelas três cromaticidades das primárias aditivas vermelha, verde e azul que ao se somarem, produzem a cor desejada. Este modelo é representado por um sistema de coordenadas cartesianas tridimensional, conforme mostrado no cubo da Figura 4, onde a diagonal representa intensidades iguais nas três cores, o que resulta no conjunto de tons de cinza.

Figura 4 – Cubo tridimensional do sistema RGB.



Fonte: Masutani *et al*, 2013.

Na representação do espaço de cores RGB como um cubo unitário tridimensional, as cores primárias, vermelho (R), verde (G) e azul (B) formam o sistema de coordenadas: o vermelho "puro" Cor (R), verde (G), azul (B), ciano (C), magenta (M) e amarelo (Y) estão nos vértices do cubo.

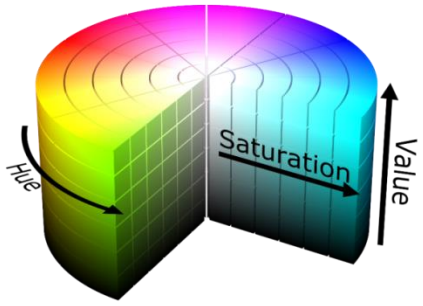
O sistema de cores RGB é o mais usado no sistema gráfico de computadores, pois disponibiliza diversas bibliotecas e API's gráficas simplificando a arquitetura no desenvolvimento de um projeto em computação.

Entretanto, o sistema de cores RGB não é o mais eficiente para tratar as imagens do mundo real. As três componentes de cor necessitam ter uma mesma faixa de valores para gerar qualquer cor do cubo de cores do sistema RGB. Como consequência, é necessário um sistema de armazenamento temporário, e com a mesma resolução para as três cores, por isso, utiliza-se a transformação para HSV (SANTOS, 2009).

3.2.1.2 Espaço de cores HSV

O modelo ou espaço de cores HSV é uma transformação não linear do espaço de cores RGB. A definição do modelo do espaço de cores HSV não pode ser dada de forma independente, somente com a definição relativa das intensidades, definições físicas das suas cromaticidades e ponto branco, do espaço de cores RGB é que se chega ao HSV. Tem-se na Figura 5 o diagrama do espaço de cores HSV.

Figura 5 – Diagrama do espaço de cores HSV.



Fonte: Helal, 2015.

O modelo foi criado em 1978 por Alvy Ray Smith, pioneiro na área de Computação Gráfica, que participou da fundação de centros de pesquisas e criações gráficas como a Altamira, Pixar, Lucasfilm e New York Tech (HSV, 2020).

Em comparação ao espaço de cores RGB, o espaço de cores HSV representa as mesmas cores, mas por meio de outras entidades como a Matiz ("Hue"), a Saturação ("Saturation"), e Valor ("Value"):

Matiz: se refere ao tipo da cor, abrangendo todas as cores do espectro visível, partindo do vermelho e indo até o violeta, com a adição do magenta. Os valores do Matiz variam de 0 a 360º e é comum ser encontrado em algumas aplicações seu valor normalizado indo de 0 a 100%;

Saturação: é a "pureza" da cor, sendo que um valor menor de saturação representa uma cor mais cinza, e um valor maior, uma cor mais pura, seus valores variam de 0 a 100%;

Brilho (Valor): define o brilho da cor, com um valor menor aproximando a cor de preto, e um valor maior tornando-a mais brilhante, valores variando de 0 a 100%.

3.1.2 Conversão entre os espaços de cores.

É importante o entendimento de como ocorre a conversão entre espaço de cores. Apesar de existirem outras conversões, tem-se o interesse nesta pesquisa na conversão entre os espaços de cores RGB para o HSV. O uso frequente do espaço de cores HSV dá-se por sua grande abrangência e disponibilidade em algoritmos de programação que têm como funcionalidade a conversão dos espaços de cores, especialmente do RGB para o HSV.

Apresenta-se a seguir, como são determinados os valores do espaço de cores HSV, a partir do RGB. Sendo MAX definido como o valor máximo dos parâmetros $R, G\ e\ B$ e MIN definido como sendo os valores mínimos, assim têm-se a:

Delimitação de valores:

$$\begin{cases} 0 \leq H \leq 360º; \\ 0 \leq S, V, R, G\ e\ B \leq 1 \end{cases} \quad (1)$$

Determinando H:

$$H = \begin{cases} indefinido, & se\ MAX = MIN; \\ \left[60° * \left(\frac{G-B}{MAX-MIN}\right)\right] + 0°, & se\ MAX = R\ e\ G \geq B; \\ \left[60° * \left(\frac{G-B}{MAX-MIN}\right)\right] + 360°, & se\ MAX = R\ e\ G < B; \\ \left[60° * \left(\frac{B-R}{MAX-MIN}\right)\right] + 120°, & se\ MAX = G; \\ \left[60° * \left(\frac{R-G}{MAX-MIN}\right)\right] + 240°, & se\ MAX = B; \end{cases} \quad (2)$$

Determinando S:

$$S = 0, \quad se\ MAX = 0 \quad (3)$$

Caso contrário:

$$S = 1 - \left(\frac{MIN}{MAX}\right)$$

Determinando V:

$$V = MAX. \tag{4}$$

Na Computação gráfica, normalmente cada parâmetro de HSV e RGB é normalizado por um inteiro variando de 0 até 255 (8 bits). Neste caso a transformação não cobre todos os pontos do espaço alvo e algumas distorções são causadas por arredondamentos.

A fim de exemplificar a conversão entre os espaços de cores, demonstra-se a conversão da cor laranja. A cor laranja mais pura, em que é composta de 100% de vermelho, 64,71% de verde e 0% de azul nos parâmetros RGB é representada por (255,165,0). (ENCYCOLORPEDIA, 2023).

O primeiro passo a ser feito é normalizar os componentes R, G e B para o intervalo entre 0 e 1, para isso, divide-se cada componente pelo valor máximo 255. Os componentes R, G e B normalizados serão representados com um apóstrofo.

$$RGB = (255, 165, 0) \rightarrow$$

$$\rightarrow \begin{cases} R = 255; \\ G = 165; \rightarrow \\ B = 0; \end{cases}$$

$$\rightarrow \begin{cases} R' = \dfrac{255}{255} = 1; \\ G' = \dfrac{165}{255} = 0{,}64705; \rightarrow \\ B' = \dfrac{0}{255} = 0; \end{cases}$$

$$(RGB)' = (1, 0{,}64705, 0).$$

Portanto, $MAX = 1, MIN = 0$.

Após a normalização dos componentes, utilizam-se as expressões descritas anteriormente.

Determinando H:

$$H = \begin{cases} indefinido, & se\ MAX = MIN; \\ \left[60° * \left(\dfrac{G - B}{MAX - MIN}\right)\right] + 0°, & se\ MAX = R\ e\ G \geq B; \\ \left[60° * \left(\dfrac{G - B}{MAX - MIN}\right)\right] + 360°, & se\ MAX = R\ e\ G < B; \rightarrow \\ \left[60° * \left(\dfrac{B - R}{MAX - MIN}\right)\right] + 120°, & se\ MAX = G; \\ \left[60° * \left(\dfrac{R - G}{MAX - MIN}\right)\right] + 240°, & se\ MAX = B; \end{cases}$$

$$\rightarrow H = \left[60° * \left(\frac{G - B}{MAX - MIN}\right)\right] + 0° \rightarrow H = \left[60° * \left(\frac{G' - B'}{R' - B'}\right)\right] \rightarrow$$

$$\rightarrow H = \left[60° * \left(\frac{0{,}64705 - 0}{1 - 0}\right)\right] \rightarrow H = \left[60° * \left(\frac{0{,}64705}{1}\right)\right] \rightarrow$$

$$\rightarrow H = \left[60° * \left(\frac{0{,}64705}{1}\right)\right] \rightarrow H = 38{,}82°.$$

Determinando S:

$$S = \begin{cases} 0, & se\ MAX = 0; \\ 1 - \left(\frac{MIN}{MAX}\right); \end{cases} \rightarrow S = \left\{1 - \left(\frac{MIN}{MAX}\right) \rightarrow\right.$$

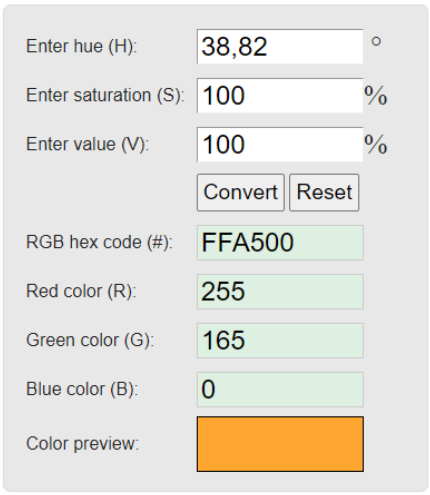
$$\rightarrow S = 1 - \left(\frac{0}{1}\right) \rightarrow S = 1 - 0 = 1.$$

Determinando V:

$$V = MAX \rightarrow V = 1.$$

Portanto, a cor laranja no espaço de cores HSV é representada por (38,82°, 1, 1). Utilizando um conversor online, têm-se a seguinte prova, na Figura 6:

Figura 6 – Conversão da cor laranja entre os espaços de cores RGB para HSV.



Fonte: Rapidtables, 2023.

3.3 Processamentos de imagem digital, a detecção e o rastreamento de objetos

Segundo Gonzalez e Woods (2009) a definição do que é o Processamento de imagem digital varia de acordo com o autor. Segundo YILMAZ et al. (2006), quando se trabalha com análise de vídeo existem três etapas principais no processo, a detecção dos objetos de interesse, o rastreamento destes e a análise do objeto para reconhecer seu movimento.

Os algoritmos tradicionais para detecção de objetos em imagens digitais utilizam imagens em tons de cinza (256 tons), porém no caso de imagens coloridas, entre as diversas formas de representação destas cores, a mais tradicional é o RGB. Porém, para segmentação de imagens coloridas em vídeos, é recomendado o modelo HSV, por se assemelhar mais com a forma de percepção do olho humano (OLIVEIRA *et al*, 2014).

Uma biblioteca bastante utilizada em visão computacional é a OpenCV que possui implementações otimizadas de muitas operações, possibilitando o processamento em tempo real de imagens digitais, mesmo quando o volume de dados é grande, como ocorre em um vídeo que pode ter, por exemplo, de 30 a 60 quadros (*frames*) por segundo para serem processados. Todavia a redução do tamanho da imagem, antes do seu processamento, proporciona a melhoria do tempo de processamento em tempo real e minimiza a perda de quadros (OLIVEIRA *et al*, 2014).

Na detecção de um objeto em movimento são executadas no tratamento de imagens, nesta ordem, as seguintes etapas: captura, detecção, pré-processamento/filtro, inspeção/análise do objeto para reconhecer seu movimento.

A detecção e o rastreamento de objetos são funcionalidades de grande importância em sistemas de visão computacional com aplicações em muitas áreas (monitoramento automático de câmeras de segurança, interação-humano-computador automática, monitoramento de tráfego de veículos, utilização em veículos autônomos e muitas outras). Em sistemas de tempo real, é necessário que se utilize ferramentas com um tempo de resposta baixo, portanto quando se refere à utilização de imagens, utiliza-se a OpenCV, biblioteca específica para tratamento de imagens.

Nesta pesquisa utiliza-se a OpenCV para realizar a detecção e rastreamento de objeto colorido em vídeo adquirido por *webcam*. Para realizar este processo as imagens capturadas são convertidas para o sistema de cores HSV, na segmentação faz-se o uso de técnicas de processamento digital de imagens, sendo que todas as funções utilizadas pertencem à esta biblioteca. Porém, somente são reconhecidos objetos de uma mesma cor, sendo, necessário estender a aplicação para que sejam tratadas mais de uma cor em uma mesma imagem.

3.4 Softwares de Suporte (OpenCV)

OpenCV é uma biblioteca do tipo *open source* direcionada para visão computacional e aprendizado de máquina que foi originalmente desenvolvida pela Intel, em 2000. É

disponibilizada em diversas linguagens, contendo várias funções em áreas como, análise de imagens médicas, calibração de câmeras, robótica, entre outras.

O OpenCV é uma biblioteca multiplataforma (Windows, Linux, OSX, Android IOS) que segue o modelo de licença BSD (Berkeley Software Distribution) Intel. Foi elaborada para fornecer uma infraestrutura comum para a aplicação de questões envolvendo visão computacional e também para acelerar o uso da percepção de máquina direcionado para produtos comerciais. Além disso, destaca-se nesta biblioteca, a modularidade e a portabilidade.

Esta biblioteca conta com mais de 2500 algoritmos (módulos) otimizados para aplicação em tempo real, que podem ser usados para a detecção e rastreamento de faces, identificação de objetos, classificação de ações humanas, rastreamento de movimento de câmera, rastreamento de objetos, extração de modelos 3D de objetos, filtros de imagens, calibração de câmera, análise estrutural, robôs móveis, realidade virtual e realidade aumentada e muitas outras aplicações. Atualmente a OpenCV suporta muitos algoritmos relacionados à visão de computador e à aprendizagem de máquinas e continua expandindo (PASSARELLI, 2017).

A linguagem principal e todos os novos desenvolvimentos e algoritmos do OpenCV é a C++. Existe compatibilidade com Java, Matlab/Octave e Python. Com a linguagem Python existem algoritmos aplicáveis às necessidades de muitas áreas de trabalho (OpenCV, 2019; BRADSKI; KAEHLER, 2008). As principais funções do OpenCV utilizadas neste projeto são apresentadas no Quadro 1.

Quadro 1 - Principais funções OpenCV

Funções	**Descrição**
videoCapture()	Classe para captura de vídeo a partir de arquivos de vídeo, sequências de imagens ou câmeras.
cvtColor()	Converte uma imagem de um espaço de cor para outro.
inRange()	Verifica se os elementos da matriz lida no frame estão entre os elementos de duas outras matrizes de cor desejada.
dilate()	A função dilata a imagem de origem usando o elemento de estruturação especificado que determina a forma de uma vizinhança de pixels sobre a qual o máximo é obtido.
erode()	A função corrói a imagem de origem usando o elemento de estruturação especificado, que determina a forma de uma vizinhança de pixels sobre a qual o mínimo é obtido.
findContours()	Recupera os contornos da imagem binária
contourArea()	Calcula uma área de contorno.

Fonte: Próprio autor.

No desenvolvimento deste trabalho e para programar com OpenCV foi utilizado o software PyCharm dotado de interpretador da linguagem Python, tecnicamente não havia necessidade de outros softwares de suporte.

O PyCharm é uma *Integrated Development Environment* — Ambiente de Desenvolvimento Integrado (IDE). É um editor de código inteligente, além de uma interface muito limpa e personalizável, é ideal para iniciantes e programação em Python. o PyCharm conta com funções internas de Python com facilidade de acesso e sem erros (VASCONCELOS, 2017).

4 PRINCIPAIS CONCEITOS DA ROBÓTICA

Neste capítulo são abordados os principais conceitos envolvendo a aplicação na robótica e em especial a robótica móvel e os seus elementos que fornecem a percepção e a ação de um robô como os sensores e atuadores, respectivamente.

4.1 A Robótica: definição e classificação

É atribuída ao teatrólogo checo Karel Capek (1890-1938) em sua peça R.U.R (Russum's Universal Robots), de 1920 a criação da palavra robô, " trabalhador forçado". Ao escritor Isaac Asimov (1920-1992) é atribuída a palavra robótica em seu conhecido livro "I, robot"(1950) onde designava um novo ramo do conhecimento associado à criação e programação de robôs.

Nos dias atuais a robótica é definida como a ligação inteligente entre a percepção e a ação. Trabalhar com robótica significa projetar, estudar e implementar sistemas ou dispositivos que, com o uso da percepção e certo grau de inteligência, sejam cada vez mais úteis na realização de determinadas tarefas, que podem ser pré-definidas ou não, e que pode envolver interação física entre o sistema robótico e o meio em que a tarefa está sendo realizada. Na concepção de um robô são envolvidos conceitos de mecânica, elétrica, eletrônica, computação (NIKU, 2010; HEINEN; OSÓRIO, 2002).

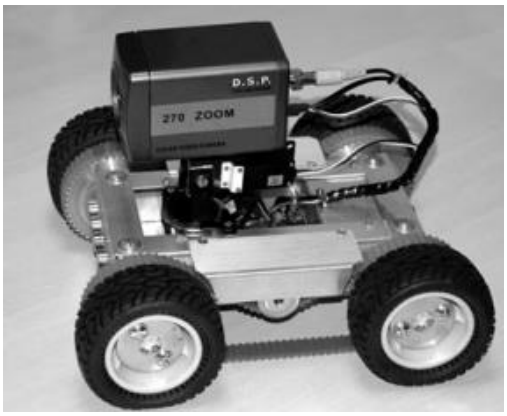
A robótica vem sendo estudada nos últimos anos em várias linhas de pesquisas, com a finalidade de auxiliar o ser humano em atividades repetitivas e de risco, e principalmente melhorar a vida das pessoas. Sua constante evolução tem impulsionado a criação de novos projetos de sensores e atuadores eficientes.

Os robôs podem ser divididos em robôs móveis, manipuladores ou híbridos que são robôs móveis equipados com dispositivos manipuladores, ilustrados na Figura 7. Os manipuladores são os mais comuns e os seus movimentos envolvem uma cadeia inteira de articuladores controláveis, permitindo que os atuadores sejam colocados em qualquer posição, usados principalmente em trabalhos repetitivos.

Figura 7 – Tipos de robôs. a) Manipuladores. b) Robôs móveis. c) Híbrido. d) Robôs móveis com pernas. e) Robôs móveis autônomos em fábricas.



(a) Festo cobot. www.festo.com



(b) Inspector Bot da Eventronic ES, um robô que inspeciona tubulações.



(c) Manipulador móvel da empresa Adept MobileRobots



(d) EESC-USP



(e) Robôs-móveis-autônomos, logística-interna-de-fábricas

Um robô móvel se desloca eu um ambiente por meio de rodas, pernas, ou mecanismos semelhantes e são usados principalmente no monitoramento e mapeamento de ambientes. Devem possuir algumas características, tais como, fusão de sensores, detectar e desviar de obstáculos, auto localização, mapeamento do ambiente, planejamento de trajetória, navegação robótica e capacidade de interagir e se comunicar com outros agentes no ambiente. Robôs móveis devem ter a capacidade de tomar decisões inteligentes para realizarem um planejamento dos seus movimentos, evitando principalmente choques com obstáculos.

4.2 Percepção e ação na robótica

A complexidade de uma aplicação robótica pode variar muito, uma vez que um agente robótico é uma entidade que influencia e é influenciado pelo ambiente em que se encontra. O robô percebe o ambiente por meio de sensores e age sobre o ambiente por meio de atuadores.

Os sensores extraem informações do ambiente que serão interpretadas e transformadas em informações úteis e necessárias para o robô executar uma ação inteligente e de forma autônoma.

Os sensores permitem a um robô detectar a presença de um objeto no ambiente e medir o alcance deste em uma determinada direção. A percepção na robótica pode ser realizada por vários tipos de sensores de temperatura, som, infravermelho, câmeras e outros.

Quanto aos atuadores, são mecanismos conectados por meio de articulações e juntas, que geram a força necessária para movimentar o sistema robótico, cujas principais funções são de locomoção e manipulação de objetos no ambiente. Os atuadores definem o grau de autonomia do sistema robótico (SCHUEROFF, 2013).

As informações do ambiente adquiridas por via dos sensores servem de entrada para um determinado algoritmo de controle para o robô que percebe o que está em sua volta e reage, de acordo as informações recebidas. Na robótica móvel são estudados principalmente três fatores fundamentais para a interação física com o ambiente que são a localização, o mapeamento e a navegação do robô.

Na localização um robô móvel deve conhecer a sua posição em relação ao mapa do ambiente, sendo que esse mapa é definido em um sistema global de coordenadas.

No mapeamento o levantamento dos dados do ambiente no qual está inserido o robô, é uma consequência das informações dos mapas que serão gerados e por meio destes o robô

verifica os possíveis caminhos que os levam à posição desejada e os obstáculos que devem ser evitados (SCHUEROFF, 2013).

Na navegação autônoma o robô usa apenas dados providos por sensores locais, coletados à medida que se desloca em um ambiente ainda não mapeado, sendo um dos desafios da robótica móvel autônoma. O uso de robôs móveis está cada vez mais perto do cotidiano das pessoas, o que demonstra que esta área é ainda bastante promissora. O uso de robôs para aplicações domésticas como aspiradores de pó e cortadores de grama são exemplos disso. Outras aplicações incluem as militares, como os aviões autônomos de monitoramento aéreo, aplicações industriais como robôs de logística e, por fim, aplicações urbanas, como transporte de pessoas por meio de veículos independentes.

Neste trabalho o interesse ficou restrito ao controle do robô implementado com visão artificial e no algoritmo de detecção e rastreamento de objeto aplicado à navegação, incluindo sensores ultrassônicos, sem o auxílio de mapas, portanto não sendo necessário nenhum conhecimento prévio sobre o ambiente, apenas são usadas técnicas baseadas em comportamentos pré-definidos pelo sistema de controle do robô.

5 MATERIAIS UTILIZADOS E CARACTERÍSTICAS

Neste capítulo são descritas as características dos componentes eletrônicos utilizados na execução do projeto tais como, as plataformas de desenvolvimento, os sensores ultrassônicos, webcam, acionamento e motores. Por fim, a fonte de alimentação do sistema e a regulação de tensão.

5.1 Plataformas de desenvolvimento

Para o controle do sistema embarcado do robô foram usadas duas plataformas de desenvolvimento, a Arduino UNO R3 e a Raspberry Pi 3 Model B+ da Agência Nacional de Telecomunicações (ANATEL).

A Placa Raspberry Pi 3 Model B+ ANATEL mostrada na Figura 8, usada neste projeto é uma revisão das placas Raspberry Pi 3, constituída por um processador, 1.4GHz quad-core de 64 bits, clock CPU de 1,4 GHz, dual-band wireless LAN wireless 802.11n, 2,4GHz/5.GHz, Bluetooth 4.2/BLE, alta velocidade de ethernet e consumo de 2,5A, considerada de baixo custo, pode ser conectada a um monitor (ou TV), teclado e mouse convencionais.

Figura 8 – Plataforma Raspberry Pi 3 Model B+ ANATEL.



Fonte: Raspberry Pi Foundation, 2019.

A Raspberry Pi 3 Model B+ ANATEL é um mini-PC que roda distribuições Linux como o Raspbian e Ubuntu, mas também suporta outros sistemas operacionais como o Windows 10 IoT e versões customizadas do Linux. Esta versão proporciona muito mais velocidade para a conexão, melhorando o desempenho da placa que é 17% maior em comparação a outros modelos de Raspberry, devido às suas especificações, aliadas à outras melhorias no conjunto do hardware.

Esta placa é capaz de navegar na internet, rodar vídeos de alta definição até jogos, processamento de câmeras de monitoramento e projetos embarcados com algoritmos mais

complexos. Pela capacidade de interagir com o mundo externo, tem sido utilizada em uma grande gama de projetos digitais, superando a ideia original desde que foi lançado o Raspberry Pi, que era de habilitar pessoas de todas as idades a explorar a computação e a aprender a programar em linguagens como Scratch e Python.

A certificação (homologação) por parte da ANATEL garante que o equipamento está dentro das especificações definidas pela agência, e em conformidade com a regulamentação brasileira de telecomunicações, o que significa que é permitido desenvolver e comercializar produtos utilizando esta placa. Na Tabela 1 tem-se um resumo das especificações deste modelo.

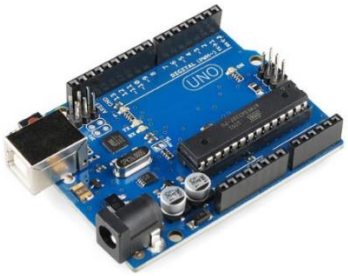
Tabela 1 – Especificações do Raspberry Pi 3 Model B+ Anatel.

Características	**Especificações**
Processador:	Broadcom BCM2837B0 64 bits ARM Cortex-A53 Quad-Core (SoC) , 1.4 GHz
Memória RAM:	1GB (LPDDR2 SDRAM)
Conectividade:	WiFi 2.4 GHz e 5.0 GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2 BLE
Portas USB:	4 portas USB 2.0
GPIOs:	40 pinos, compatíveis com as versões anteriores.
Conector de vídeo (saída):	1 porta HDMI (Full-size)
Periféricos:	Conector CSI para câmera; Conector DSI para display
Saída P2:	Conector de 4 polos para áudio estéreo e saída de vídeo
Slot para MicroSD:	Carregamento do sistema operacional e armazenagem
Alimentação:	Fonte: 5V, 2.5A (mín) através do conector micro USB
Dimensões:	85 x 56 x 17mm
Power Over Ethernet:	Compatível desde que usado o PoE HAT
Homologação da Anatel:	n º 01598-18-10629

Fonte: Raspberry Pi; Raspberry 4 pinout, 2019.

Quanto à plataforma Arduino UNO R3, mostrada na Figura 9, é uma das mais vendidas da linha Arduino e usada atualmente. Costuma ser a primeira opção de muitos, pois apresenta uma boa quantidade de portas disponíveis e grande compatibilidade com os *shields (*placas auxiliares) Arduino.

Figura 9 – Plataforma Arduino UNO R3.



Fonte: Arduino, 2018.

O Arduino UNO R3 é uma plataforma *open hardware*, possuindo seu próprio ambiente de desenvolvimento baseado na linguagem C, de fácil programação. Este Arduino, especificamente, é uma versão do original produzido na Itália, mas possui a mesma estrutura e componentes. O software é gratuito e disponível para Mac OS X, Windows e Linux.

Esta plataforma está baseada em hardware e software de fácil uso, capaz de ler entradas de sensor de luz, toque, botão e transformá-las em saídas, ativando um motor ou mesmo ligando um Light Emitting Diode (LED). O projetista programa usando a linguagem de programação do Arduino e seu sistema de desenvolvimento - IDE, depois disso a placa é programada. Por sua simplicidade no manuseio e acessibilidade de conhecimentos da comunidade em geral, tem sido usada do Ensino Fundamental até a Universidade, por hobbistas, etc. Um resumo de suas características é apresentado na Tabela 2.

Tabela 2 – Características do Arduino UNO R3.

Características	**Especificações**
Microcontrolador	ATmega328
Velocidade do *clock*	16 MHz
Memórias	Flash: 32KB (0,5KB de bootloader); SRAM: 2KB; EEPROM:1KB
Portas digitais	14 (6 podem ser usadas como PWM)
Portas analógicas	6
Correntes pinos I/O	40 mA
Corrente pinos 3,3V	50 mA
Tensão de entrada	7-12 V
Tensão de operação	5 V

Fonte: Arduino, 2018.

5.2 Sensores

Visando proporcionar uma melhor navegabilidade ao robô, enquanto busca na cena o objeto foram usados dois tipos de sensores no protótipo, uma webcam e os sensores ultrassônicos que medem a distância ao obstáculo.

5.2.1 Webcam

A webcam usada no projeto é o módulo OV5647, mostrado na Figura 10, trata-se de um sensor de 5 megapixels (5mp) capaz de gravar vídeos e imagens em 1080p, conectando diretamente no Raspberry Pi.

Figura 10 – (a) Webcam. (b) Webcam e cabo flat.



(a) (b)

Fonte: Raspberry Pi Documentation, 2021.

Está é a última versão do tipo plug-and-play diretamente compatível com o sistema operacional do Raspberry Pi, faz tomada de fotografias em *time-lapse*[2], gravação de vídeo, e aplicações de detecção de movimento. A placa da câmera é pequena com dimensões 25 mm x 23 mm x 8mm e pesa cerca de 3 gramas, importante para aplicações em que peso e dimensões são considerados.

Dentre as suas especificações têm-se uma lente de foco fixo, capaz de fornecer resolução de imagem estáticas de 2592x1944pixels, resolução de vídeo de 1080p/30 fps, 720p/60 fps e 640x480p 60/90fps de gravação (sem áudio), ângulo visual (FOV) de 65 graus, conector do tipo CSI e suporte para o Raspbian, sendo compatível com os Raspberry Pi 4 Model B/3B+/3B/2B/Zero wireless. Na Tabela 3 tem-se um resumo de suas características.

[2] **Time lapse-** pode ser traduzido como "lapso de tempo". Na prática, é um "vídeo" passando rápido, como se estivesse acelerado. A palavra vídeo entre aspas significa uma sequência de fotos, que na pós-produção transforma-se em vídeo.

Tabela 3 - Características do módulo OV5647.

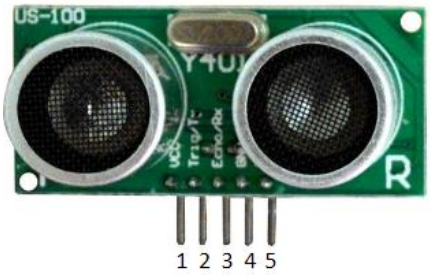
Características	Especificações
Sensor de 5mp	OV5647
Cor	CMOS QSXGA (5 – megapixel)
Resolução de imagem estática	2592×1944
Saída de vídeo	1080p/30 fps, 720p/ 60fps and 640 x480p 60/90
Abertura (F)	1,8
Ângulo de visão	65 degree
Saída de vídeo	1080p a 30 fps com codec H.264 (AVC)
Tamanho do sensor	3,67 x 2,74 milímetros (mm)
Lente	f = 3,6 mm, f / 2.9 (abertura)
Ângulo de visão	54 x 41 graus
Campo de visão	2,0 x 1,33 m, a 2 m
Full-frame SLR lente equivalente	35 mm
Foco fixo	1 m ao infinito
Interface	Conector CSI (Camera Interface Serial)
Compatível com	Raspberry Pi 4 Model B/3B+/3B/2B/Zero wireless
Tamanho da placa /Peso	25 x 24 x 8 mm/3g
Tamanho do cabo flat	15 cm

Fonte: Raspberry Pi Documentation, 2021.

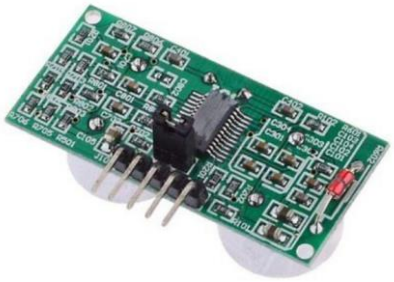
5.2.2. Sensor ultrassônico

O sensor ultrassônico US-100 foi utilizado para medir distância de um objeto na faixa de 2 a 450 cm, com baixo custo. Apresenta boa precisão que pode chegar a 3 mm e segurança de sinal para mais de 5 metros até seu sinal desaparecer por volta de 7 metros, com baixo consumo energético. Pode ser utilizado em projetos de robótica (robôs labirinto ou sumô) e em projetos de automação como sistemas de medição de nível ou distância. O módulo inclui dois transdutores: um transmissor e um receptor piezelétricos, e o circuito de controle, conforme mostrado na Figura 11.

Figura 11 – Sensor ultrassônico US – 100. (a) Configuração dos pinos. (b) Verso



(a)



(a)

Fonte: E-gizmo, 2022.

O funcionamento do sensor ultrassônico depende do piezelétrico emissor que emite 8 pulsos ultrassônicos (40KHz) ao receber a ordem no pino input TRIGGER (pelo menos 10 microssegundos em nível lógico alto), as ondas de som se propagam no ar e retornam ao encontrar o objeto. O som de retorno é detectado pelo piezelétrico receptor no pino output ECHO, que troca para nível alto (5V) por um tempo igual ao que demorou a onda desde que foi emitida até que foi detectada, seu diagrama de tempo é mostrado na Figura 12. Este tempo do pulso ECHO é medido pelo microcontrolador do módulo US-100 e assim se calcula a distância ao objeto utilizando a Equação (5),

$$\text{Distância(m)} = \{(\text{Largura do pulso ECHO}) * (\text{Velocidade do som})\}/2. \quad (5)$$

Sendo a velocidade do som =340m/s e largura de pulso em segundos

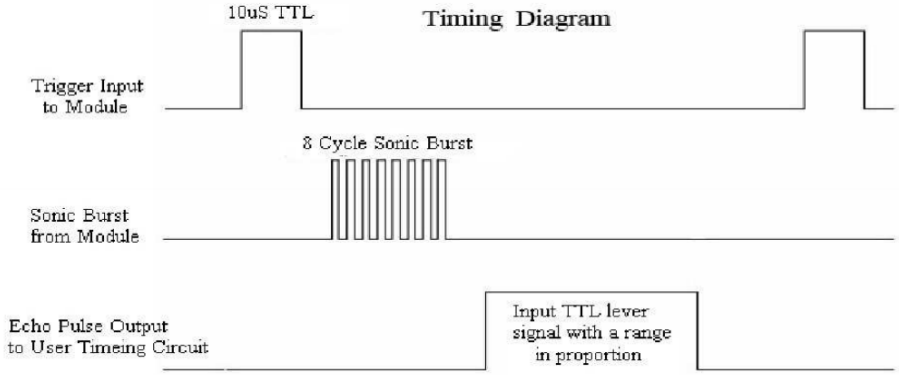
Como o sensor US-100 apresenta compensação de temperatura, o efeito da temperatura no cálculo da distância (E-GIZMO, 2022) é dado pela Equação (6), (6)

$$\text{Distância(m)} = (\text{Largura do pulso ECHO}) * 165{,}7\text{m}$$

O funcionamento do sensor não é afetado pela luz solar ou material de cor (embora os materiais acusticamente macios como tela ou lã possam ser difíceis de detectar).

O fabricante recomenda o uso de mais de 60ms de medida por ciclo, para prevenir que o sinal TRIGGER sobreponha ao sinal ECHO. Na Tabela 4 são resumidas as especificações do sensor US-100.

Figura 12 – Temporização dos sinais de TRIGGER e ECHO para o US-100.



Fonte: Sunfounder, 2019.

Tabela 4 - Especificações do sensor US-100.

Características	**Especificações**
Alimentação e Tensão lógica	DC 2.4~5.5 (V)
Corrente	2 mA
Saída digital	0-5V
Temperatura de operação	-20~70°C
Ângulo de medida	Menos de 15º
Distância de detecção	2 cm – 450 cm
Acurácia	0.3 cm + 1%
Dimensões	45 x 20 mm
Peso	9 g

Fonte: E-gizmo, 2022.

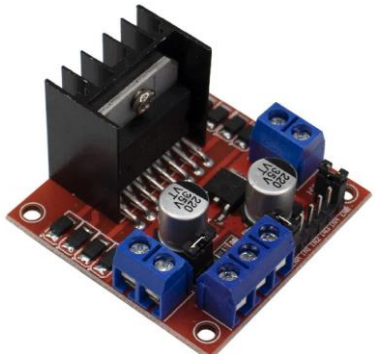
5.3 Acionamento do motor CC.

Para fornecer a potência necessária para acionar motores de Corrente Contínua (CC), utiliza-se normalmente um driver, Circuito Integrado (CI) L298, que possui internamente duas pontes H, capazes de atuar em dois dispositivos independentemente, pois cada uma possui uma entrada de ativação, na qual pode ser usado um sinal de PWM (Pulse Width Modulation). O driver amplifica o sinal de baixa potência fornecido pelas plataformas de desenvolvimento como o Raspberry ou Arduino. Neste trabalho era necessário acionar os dois motores CC do protótipo, para isso foi usado um módulo driver ponte H L298N.

5.3.1 Módulo driver ponte H L298N

Visando facilitar o desenvolvimento do projeto, foi utilizado um módulo driver ponte H L298N, mostrado na Figura 13, baseado no circuito integrado L298N. Este módulo driver como utiliza o CI L298 pode controlar independentemente a velocidade e rotação de 2 motores CC ou um motor de passo. Possui dois canais e permite controlar velocidade e sentido de rotação de até dois motores ao mesmo tempo, têm terminais parafusáveis para fácil instalação e furos nas extremidades da placa, para fixação ao protótipo, na Tabela 5 é apresentado um resumo das especificações deste módulo.

Figura 13 – Módulo driver ponte H L298N.



Fonte: Blog Eletrogate, 2020.

Tabela 5 – Especificações técnicas do módulo driver ponte H L298N.

Características	**Especificações**
Tensão de Operação	4,5 V a 46 V
Chip	ST L298N
Corrente de Operação máxima	2ª por canal ou 4ª max
Tensão lógica	4,5V a 7V
Corrente lógica	0~36mA
Limites de Temperatura	-20 a +135°C
Potência Máxima	25W
Dimensões	43 x 43 x 27mm
Peso	30g

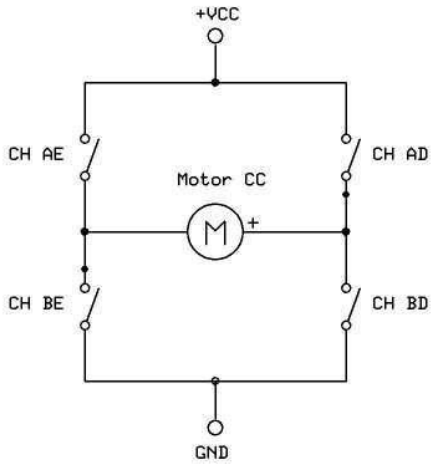
Fonte: Blog Eletrogate, 2020.

Uma das vantagens de se usar um circuito L298 em módulo é a confiabilidade das ligações por ser placa comercial, além do CI L298 fornecer uma maior dissipação de potência em relação ao acionamento direto por portas de circuitos microcontrolados que só conseguem fornecer alguns mA de corrente, contra 2A por canal da ponte H L298N.

O CI L298 foi desenvolvido de forma a aceitar sinais no padrão TTL, podendo controlar cargas indutivas, como relés, solenoides, motores CC e motores de passo, possuindo a capacidade de atuar com tensões de alimentação de até 46 V, e corrente total de até 4 A.

O circuito esquemático de uma Ponte H é apresentado na Figura 14, constituído de quatro chaves que permitem o giro dos motores para ambos os lados. Para que o motor gire em um sentido, as chaves CH, AD e CH, BE são fechadas. Caso o motor gire no sentido oposto abrem-se as chaves anteriores e fecham-se as CH, AE e CH, BD, funcionando alternadamente, duas de cada vez. Por meio de comutação de chaves, a Ponte H consegue fazer o motor ficar em vários estados diferentes de rotação.

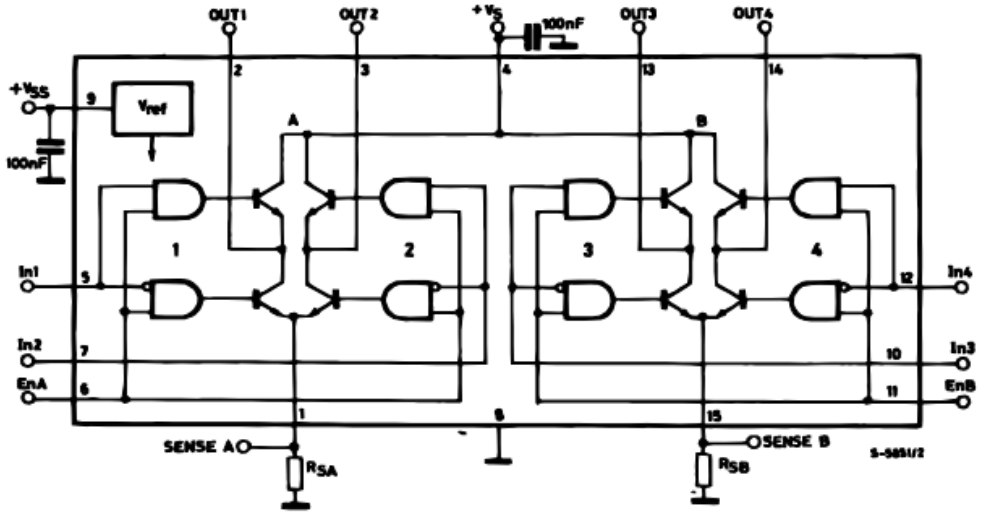
Figura 14 – Circuito esquemático da Ponte H.



Fonte: Blog Eletrogate, 2020.

O circuito elétrico interno do L298 é mostrado na Figura 15, cujo controle do sentido de rotação do motor CC é realizado eletronicamente (comutação de chaves da Figura 14) a partir da combinação de dois sinais digitais, IN1 e IN2 para um motor e IN3 e IN4 para o outro, controlando a polaridade da tensão aplicada. Além disso, para que seja fornecida tensão aos terminais, é necessário que as entradas de ativação ENA (para o motor com controle a partir de IN1 e IN2) e ENB (para o motor com controle a partir de IN3 e IN4) estejam em nível lógico alto ou que haja um sinal PWM para controlar a velocidade. A lógica do sentido de giro dos dois motores acoplados ao CI é apresentada na Tabela 6.

Figura 15 – Circuito interno para o CI L298.



Fonte: Stmicroeletronics, 2000.

Tabela 6 – Lógica de controle para o CI L298.

ENA(B)	IN1(IN3)	IN2(IN4)	Sentido
0	X	X	Parado
1	0	0	Parado
1	1	1	Parado
1	1	0	Horário
1	0	1	Anti-horário

Fonte: Stmicroeletronics, 2000.

5.3.2 Motor CC

Foi utilizado na carcaça do protótipo, o motor CC do fabricante Microred Microredutores e Motores Ltda (2010), modelo P, apresentado na Figura 16, que fornece tensão nominal de ± 12 V em CC, velocidade angular de 17 rpm (sem carga no seu eixo) e para aumentar o torque aplicado às rodas, possui uma caixa de redução.

Figura 16 – Motor CC da MICRORED modelo P.



Fonte: Microred [...], 2010.

5.4 Fonte de alimentação

Para a alimentação do robô e visando sua autonomia com o hardware embarcado, foi necessário utilizar duas baterias. A bateria escolhida foi a de Li-Po, de três células de 11,1 V e capacidade de 2200 mAh, mostrada na Figura 17, que possui uma vida útil melhor em relação aos demais tipos de baterias.

As baterias de Li-Po (Lithium Polymer Battery – Bateria de Polímeros de Lítio), tem a capacidade de descarregar altas energias sem sofrer danos à estrutura, característica diferenciada em relação às demais, mas assim como todas as outras também possui limitações físicas (DRONENG, 2022).

Figura 17 – Bateria de LiPo de 11,1 V, 2200 mAh.



Fonte: Arias, 2022.

Para regular a tensão fornecida pela bateria, de 11,1V para 5 V foi utilizado o módulo regulador CC/CC LM2596 ajustável comumente chamado de switch regulador *step - down*, apresentado na Figura 18. Este módulo regulador de tensão é constituído pelo regulador chaveado LM2596, que trabalha com tensões de entrada de 3,2V a 40 V, resultando em suas saídas tensões de 1,5V a 35 V ajustadas conforme necessidade por um potenciômetro e dadas pelos dois terminais de entrada $+V_{IN}$ e $-V_{IN}$ e dois de saída $+V_{OUT}$ e $-V_{OUT}$. Fornece uma corrente de saída de até 3A e velocidade de comutação de 150 kHz, características resumidas na Tabela 7 Apresenta-se na Figura 19, o circuito elétrico do módulo LM2596.

Figura 18 – Módulo regulador de tensão LM2596.



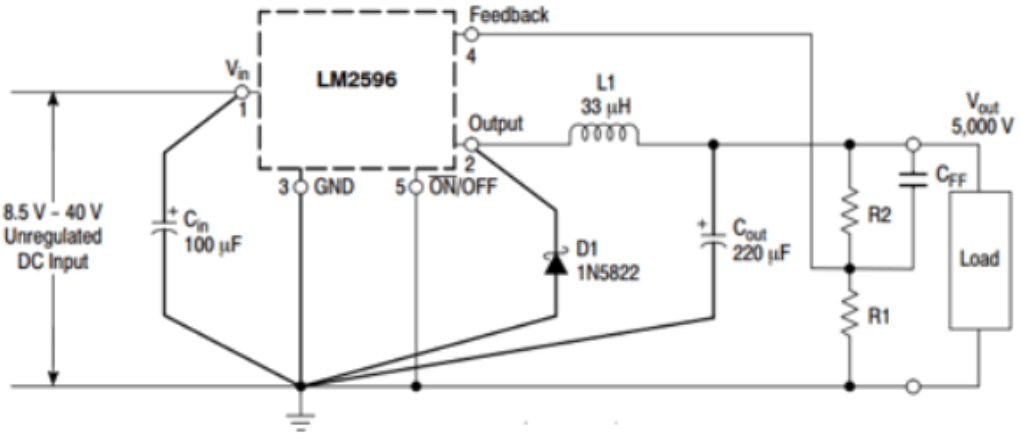
Fonte: Makerhero, 2022.

Tabela 7 – Características do módulo regulador de tensão LM2596.

Características	Especificações
Step-down de tensão de regulação variável	3,3 V, 5 V e 12 V
Tensão de alimentação de entrada	4,5 V a 40 V
Tensão mínima de saída	3, 16 V
Corrente continua de saída	3 A
Pico de corrente	6,9 A
Frequência de switch	150 kHz

Fonte: Makerhero, 2022.

Figura 19 – Circuito interno do módulo LM2596.



Fonte: Makerhero, 2022.

Com o esquemático do circuito do módulo LM2596 pode-se calcular, pelo pino de *feedbak*, a tensão de saída do LM2596 dada pelo divisor de tensão formado pelos resistores R_1 e $R_{2,}$, na Expressão (7),

$$V_{saida} = V_{referência(in)} * \left(1 + \frac{R_2}{R_1}\right) \quad (7)$$

O módulo Regulador de Tensão ajustável LM2596 foi desenvolvido para aplicação em projetos eletrônicos onde são necessárias diferentes tensões e correntes para acionamento de componentes eletrônicos. Para o seu funcionamento correto é necessário observar a diferença de tensão de entrada sempre de 1,5V maior que a de saída. É ideal para alimentar motores, relés, displays ou outros componentes elétricos que necessitam compatibilizar suas tensões, e operam dentro das faixas de tensões e correntes especificadas.

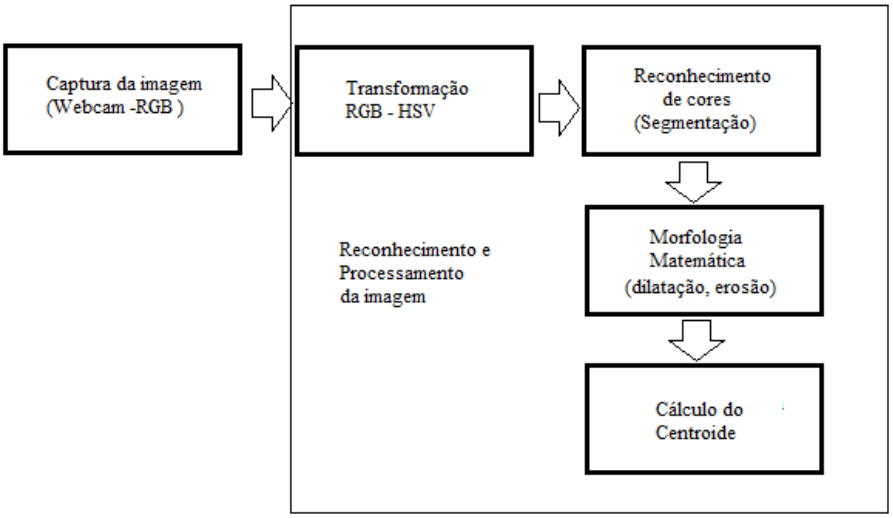
6 DESENVOLVIMENTO DO ALGORITMO DE DETECÇÃO E RASTREAMENTO

Neste capítulo descreve-se o desenvolvimento do algoritmo de detecção e rastreamento de um objeto de uma cor específica e os resultados dos testes do algoritmo, programado na linguagem Python e usando o software OpenCV que foi implementado, antes de ser aplicado na prática, ou seja, no robô.

6.1 Diagrama de blocos do algoritmo

Quando se processa uma análise de vídeo, segundo YILMAZ *et al*. (2006), existem três etapas principais, a detecção dos objetos de interesse, o rastreamento destes e a análise do objeto para reconhecer seu movimento. Desta forma, apresenta-se na Figura 20, o diagrama de blocos do algoritmo do processamento de imagem para a detecção e rastreamento de um objeto.

Figura 20 - Diagrama de blocos do processamento de imagem.



Fonte: Próprio autor

Inicialmente, no desenvolvimento do algoritmo de detecção do objeto, devido à pandemia e a impossibilidade de usar o laboratório, foi utilizada para a captura do frame, a câmera do notebook do autor junto com a biblioteca OpenCV.

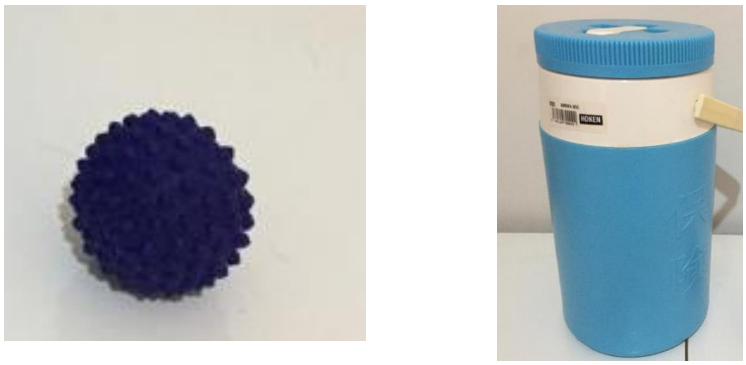
Conforme o diagrama de blocos do processamento da imagem mostrado anteriormente, depois da captura da imagem faz-se a conversão entre os espaços de cores, configurando a etapa de pré-processamento, na sequência têm-se às transformações morfológicas, em que o frame capturado é tratado para melhorar ainda mais o resultado de inspeção pela cor especificada pelo projetista. Após a especificação completa do tratamento realizado, faz-se a análise do objeto

em questão com relação ao ambiente em que se encontra, ou seja, a etapa de detecção, com isso obtêm-se as suas coordenadas - a altura e distância relativa à borda do frame e também a área que o objeto ocupa, completando assim, o rastreamento do objeto.

6.2 O algoritmo de detecção e rastreamento e os resultados

Conforme mencionado anteriormente, usando a biblioteca OpenCV em conjunto com a linguagem Python, foi desenvolvido o algoritmo de detecção e rastreamento para detectar a cor azul, escolhida aleatoriamente, para dois objetos disponíveis, uma bolinha anti-stress, Figura 21 (a) e uma garrafa térmica, na Figura 21 (b), mas com tonalidades de azul diferentes. Ressalta-se que é possível fazer a detecção e o rastreamento independente da tonalidade e textura. Também é importante observar que para esta detecção foi considerado um ambiente controlado, ou seja, com fundo branco visando facilitar a detecção e obter melhores resultados.

Figura 21 – (a) Bola anti-stress. (b) Garrafa térmica.



(a) (b)

Fonte: Próprio autor.

6.2.1 Transformação da imagem e o reconhecimento de cores

O trecho do algoritmo referente à utilização da câmera do notebook e à transformação da imagem e reconhecimento de cores é apresentado na Figura 22.

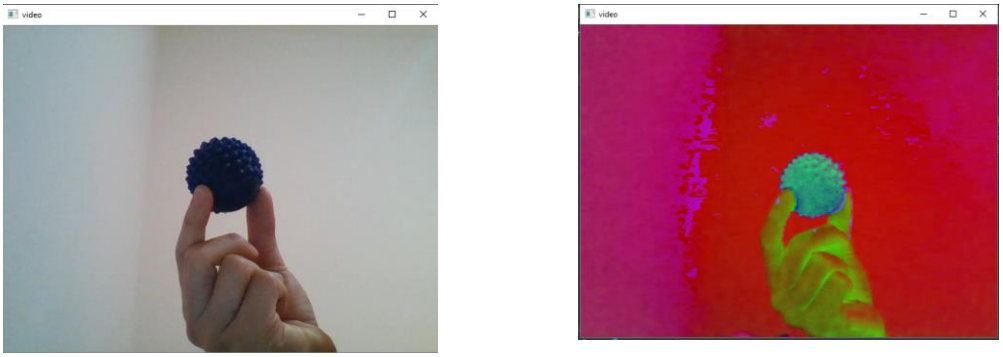
Figura 22 – Inicialização da câmera e leitura do frame.

```
176  cap=cv2.VideoCapture(0)
177  _, frame=cap.read()
178  linha, coluna, _=frame.shape
179  x_medio=int(coluna/2)
180  centro_servo_x=int(coluna/2)
181  y_medio=int(linha/2)
182  centro_servo_y=int(linha/2)
183
184
185  #inicia o loop da camera
186  while (cap.isOpened()):
187      ret, frame=cap.read()
188      if ret == True:
189
190    #aqui começa a parte de tratamento do frame!
191          frame=cv2.rotate(frame, cv2.ROTATE_180)
192          hsv=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Fonte: Próprio autor.

Na linha 176 inicia-se a webcam criando um objeto utilizando a função cv2.VideoCapture(*i*) que dá o acesso a webcam do notebook, cujo índice *i* (dentro do parêntese) é zero e na linha 177 é a função cap.read com a captura dos frames . Em seguida, a partir da linha 186, com os dados adquiridos da função cv2.VideoCapture() inicia-se o loop principal que faz a leitura dos frames provenientes da webcam, utilizando para isso a função read(). Posteriormente, na linha 192, utilizando a função cv2.cvtColor() modifica-se o sistema de cores, que antes era em RGB para o HSV, cujos resultados da transformação são mostrados na Figura 23.

Figura 23 – Comparando espaços de cores. (a) RBG. (b) HSV.



(a) (b)

Fonte: Próprio autor.

Em seguida definem-se os limites da cor azul no espaço HSV, cujo programa é mostrado na Figura 24.

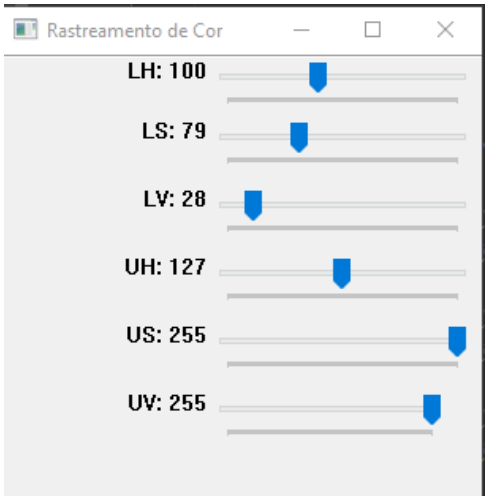
Figura 24 – Espaço e limite de cores no espaço HSV.

```
194  #define o espaço de cores
195      low_blue = np.array([0, 168, 89])
196      high_blue = np.array([12, 255, 255])
197
198  #bloco de tratamento da imagem
199      mascara_azul = cv2.inRange(hsv, low_blue, high_blue)
```

Fonte: Próprio autor.

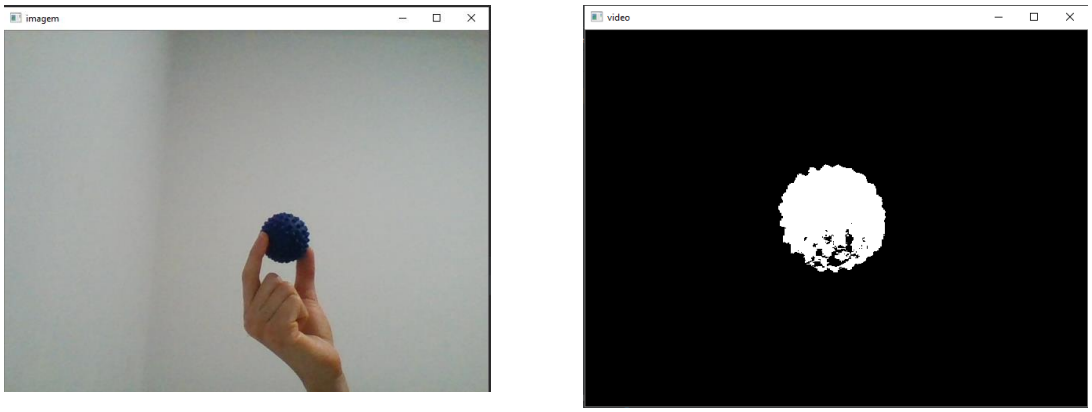
Usando a função cv2.inRange() definiu-se os valores inferiores (Low-blue) e superiores (high_blue), da cor azul no espaço HSV. Estes limites, dados nas linhas 195 e 196 são adquiridos separadamente, utilizando um algoritmo específico com barras-guia, conforme mostrado na Figura 25 e de acordo com a variação dos valores dos limites HSV, é possível visualizar o resultado em tempo real da máscara, para determinada cor. Os valores dos limites definidos são transportados para a função cv2.inRange(), cujo resultado para a imagem da bola anti-stress da Figura 26 (a) é mostrado na Figura 26 (b).

Figura 25 – Cálculo dos limites com barras - guia da cor azul.



Fonte: Próprio autor.

Figura 26 – Bola anti-stress: (a) antes da aplicação da função inRange(). (b) depois da aplicação da função inRange()



(a) (b)

Fonte: Próprio autor.

6.2.2 Morfologia matemática (erosão e dilatação)

Na imagem da Figura 26 são notadas algumas falhas devido à irregularidade na superfície do objeto. Para resolver esse problema utilizam-se os métodos de erosão e dilatação descritos no código dada Figura 27.

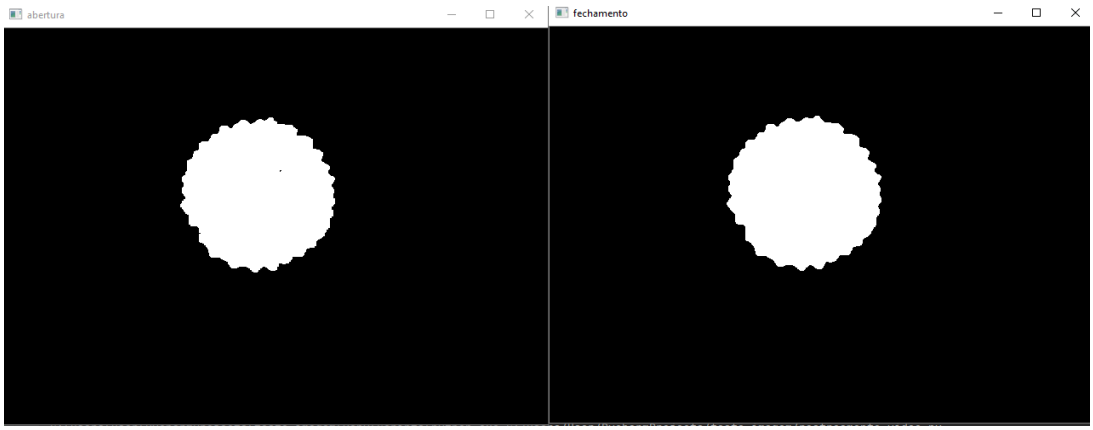
Figura 27 – Erosão e dilatação do frame.

```
198  #bloco de tratamento da imagem
199      mascara_azul = cv2.inRange(hsv, low_blue, high_blue)
200      kernel = np.ones((3, 3), np.uint8)
201      erosao=cv2.erode(mascara_azul, kernel, iterations=2)
202      erode_blue = cv2.erode(mascara_azul, kernel, iterations=1)
203      dilate_blue = cv2.dilate(erode_blue, kernel, iterations=1)
204      dilate_blue = cv2.dilate(dilate_blue, kernel, iterations=1)
205      erode_blue = cv2.erode(dilate_blue, kernel, iterations=1)
206      medianblur_blue = cv2.medianBlur(erode_blue, 3)
```

Fonte: Próprio autor.

Aplica-se primeiro o processo de “abertura” presente nas linhas 202 e 203, em que ocorre a erosão seguida de dilatação, e após isso o processo de “fechamento” nas linhas 204 e 205, em que ocorre a dilatação seguida de erosão, cujos resultados desses processos são apresentados na Figura 28.

Figura 28 - Processo de abertura seguido do processo de fechamento.



Fonte: Próprio autor.

6.2.3 Cálculo do centroide e detecção dos contornos

Posterior ao método de erosão e dilatação é a detecção dos contornos e aquisição da posição relativa dos objetos que aparecem na imagem, utilizando os códigos dados na Figura 29.

Figura 29 – Detecção dos contornos e posição do objeto.

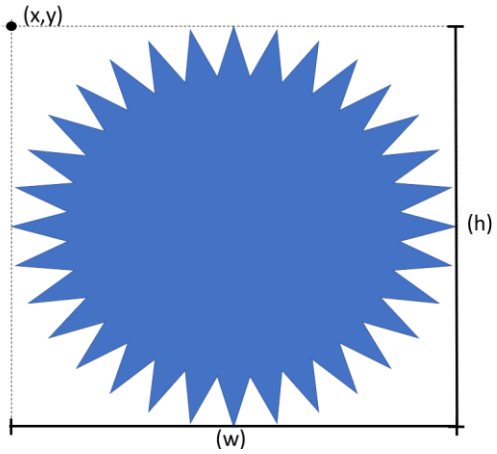
```
207        contorno_azul, _l= cv2.findContours(medianblur_blue, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
208        contorno_azul=sorted(contorno_azul, key=lambda x:cv2.contourArea(x), reverse=True)
209
210    #aqui começa a parte de rastreamento e aquisição das coordenadas e área do objeto
211        for contorno_azul in contorno_azul:
212            (x_azul, y_azul, w_azul, h_azul)=cv2.boundingRect(contorno_azul)
213            area=cv2.contourArea(contorno_azul)
```

Fonte: Próprio autor.

A detecção dos contornos dá-se pelo código presente na linha 207, cv2.findCountours(). Nesse caso específico faz-se a detecção considerando o vídeo em que foi utilizado o tratamento dos processos de abertura e fechamento. Usa-se a função cv2.boudingRect(), presente na linha 212, para aferir o centro geométrico do objeto e obter as coordenadas dos menores x e y, largura e altura da imagem, respetivamente, além de ser possível, na linha 213, aferir a área do objeto relativa ao tamanho total do frame.

Na linguagem Python essas coordenadas referem-se ao canto esquerdo superior do objeto e para facilitar o entendimento do cálculo do centro geométrico, apresenta-se uma abordagem visual das coordenadas x e y, bem como da largura e altura da bolinha anti-stress na Figura 30.

Figura 30 – Abordagem visual das coordenadas x e y, largura e altura.



Fonte: Próprio autor.

Com as coordenadas x e y, além da largura e altura do objeto, o cálculo do centro geométrico, Cg do objeto é função de Cx e Cy , variáveis dadas pela Equações em (8),

$$Cx = ((x) + (largura/2)) \qquad Cy = ((y) + (altura/2)) \qquad (8)$$

sendo, Cx a localização do centro geométrico do objeto no eixo x; Cy a localização do centro geométrico do objeto no eixo y.

Para melhorar a visualização do centroide do objeto, são utilizadas as funções cv2.line e cv2.circle que desenha retas e um círculo, respectivamente, no centro geométrico. O esquema de cálculo do centroide é mostrado na Figura 31, sendo que os pontos são calculados pelas Equações de (9) a (12),

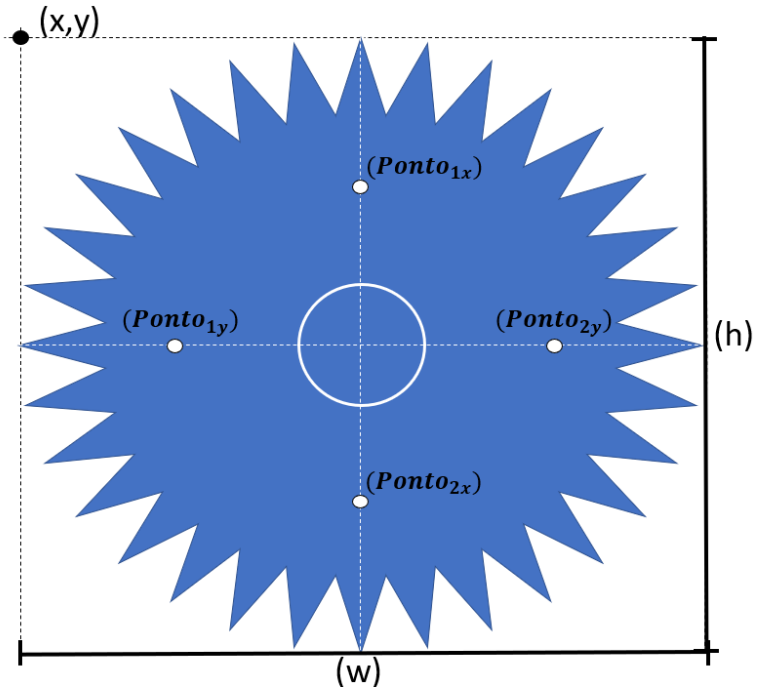
$$Ponto_{1x} = ((x + (largura/2)), (y + (altura/4))) \qquad (9)$$

$$Ponto_{1y} = ((x + (largura/4)), (y + (largura/2))) \qquad (10)$$

$$Ponto_{2x} = ((x + (largura/2)), (largura + ((3 * altura)/4))) \qquad (11)$$

$$Ponto_{2y} = ((x + ((3 * largura)/4)), (y + (altura/2))) \qquad (12)$$

Figura 31 – Cálculo do centroide



Fonte: Próprio autor.

O programa que calcula o centroide do objeto é apresentado na Figura 32.

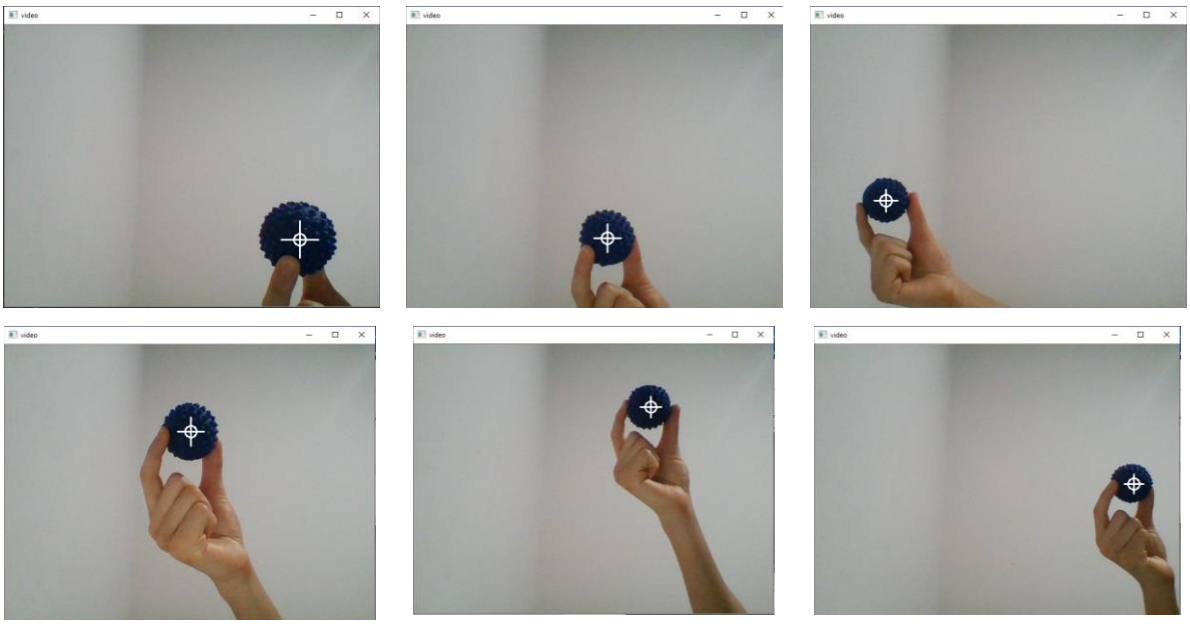
Figura 32 – Código do cálculo do centro geométrico do objeto.

```
215                 centro_x= int((x_azul)+((w_azul)/2))
216                 centro_y= int((y_azul)+((h_azul)/2))
217                 x_medio=int ((x_azul+x_azul+w_azul)/2)
218                 y_medio=int((y_azul)+((h_azul)/2))#((y_azul+y_azul+h_azul)/2)
```

Fonte: Próprio autor.

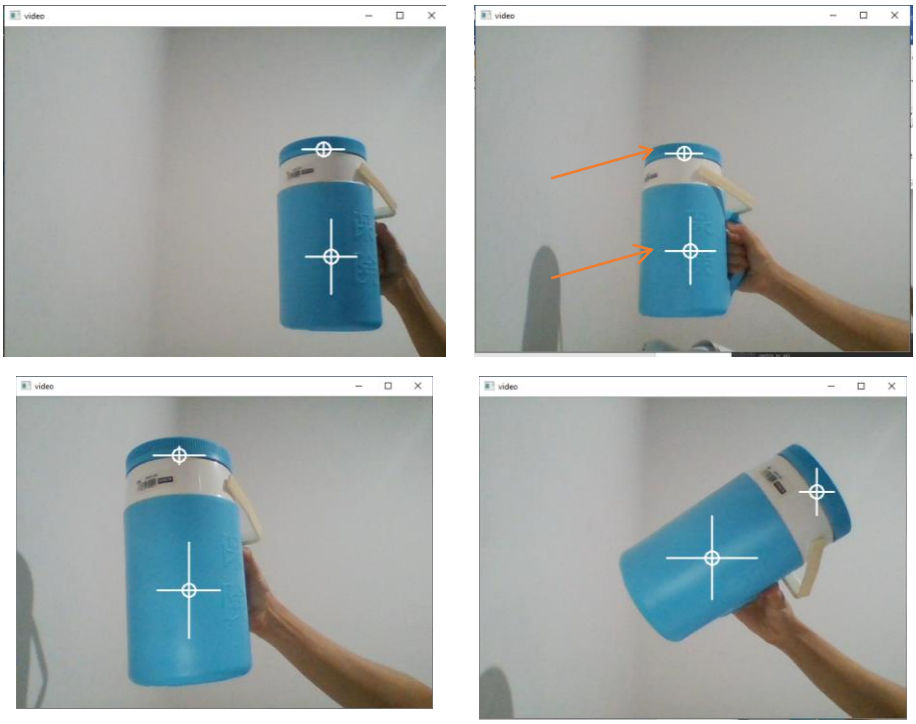
O resultado deste último procedimento é apresentado nas Figuras 33 e 34 demonstrando pela sequência de quadros, o rastreamento dos objetos, bola anti-stress e a garrafa térmica, na cor azul, respectivamente.

Figura 33 – Detecção e rastreamento da bola anti-stress.



Fonte: Próprio autor.

Figura 34 – Detecção e rastreamento da garrafa térmica.



Fonte: Próprio autor

Observando os resultados conclui-se, que a detecção e o rastreamento dos objetos em testes foram satisfatórios conforme o processamento de imagem empregado.

Nota-se que, para o caso da garrafa térmica, foram detectados dois centros geométricos de uma vez, pois a função cv2.findCountours() calcula todos os contornos que seguem as especificações dos limites de espaço HSV que nesse caso, seria da cor azul. Porém, isso pode gerar problema no rastreamento, uma vez que não é possível calcular de forma exata, a posição do objeto na tela, por conta da existência de mais de um contorno simultâneo.

Ressalta-se que foram testados mais dois métodos de detecção e rastreamento de objetos, sendo eles utilizando a função cv2.moments() e a função cv2.CamShift(). Para o caso do uso da função cv2.moments() o desenvolvimento é extremamente semelhante ao que foi utilizado para a técnica dos contornos, mas o resultado obtido não foi tão satisfatório quanto. O algoritmo era capaz de detectar o objeto, mas no momento do seu rastreamento ocorriam muitas variações no cálculo do seu centro geométrico.

Para o caso da função cv2.CamShift() era necessário determinar a posição inicial do objeto, e para isso foi desenvolvido um algoritmo específico. Determinada a posição inicial e as suas coordenadas, foram aplicados na função cv2.CamShift(), cujos resultados obtidos não se mostraram satisfatórios, uma vez que o algoritmo não conseguia rastrear o objeto por um período prolongado, em pouco tempo perdia sua referência. E como era necessário saber sua posição inicial, uma vez perdida a referência, o algoritmo não conseguia mais fazer o rastreamento do objeto.

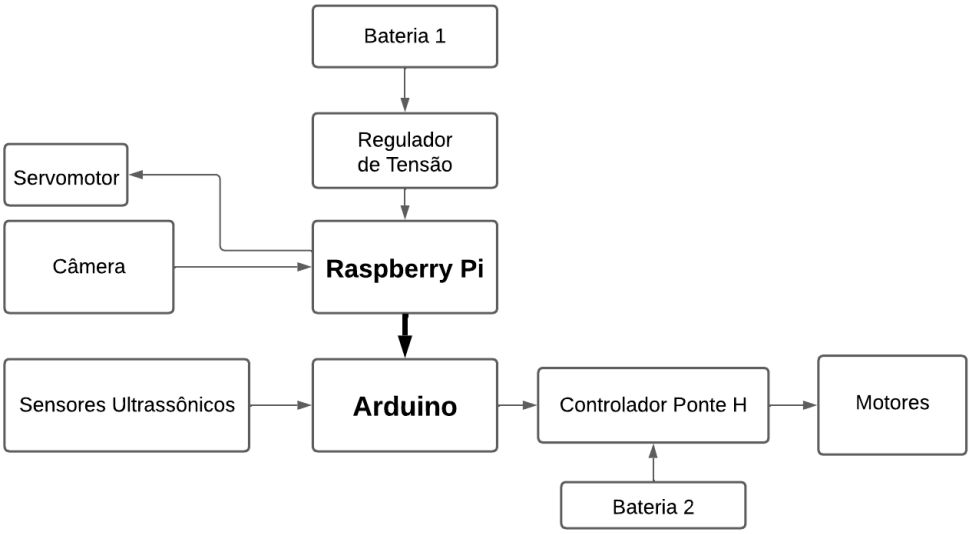
7 DESENVOLVIMENTO DO ROBÔ DOTADO DE VISÃO ARTIFICIAL

Depois da descrição dos materiais e metodologias necessárias para a implementação do hardware e software utilizados no projeto, são apresentados neste capítulo, o diagrama de bloco e o circuito esquemático embarcado no protótipo do robô. É apresentado também o fluxograma implementado para o rastreamento e detecção do objeto e os testes de navegação pelo ambiente.

7.1 Diagrama de blocos do hardware

Para o circuito elétrico do projeto foi elaborado o seguinte diagrama de blocos mostrado na Figura 35.

Figura 35 – Diagrama de blocos do hardware.



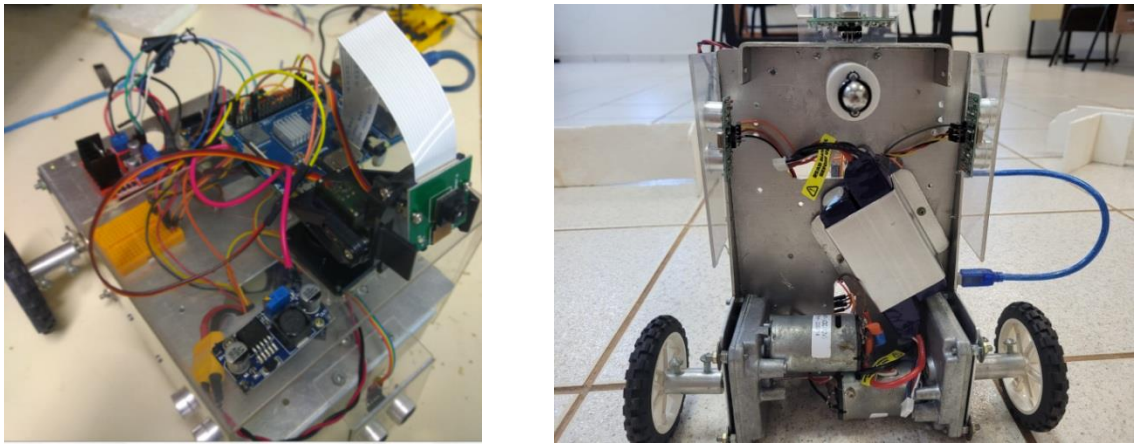
Fonte: Próprio autor.

7.2 O protótipo

O protótipo é apresentado nas Figuras 36 (a) e (b) sendo constituído de uma carcaça adaptada para receber uma webcam, módulo OV5647, três módulos sensores ultrassônicos US-100 localizados um em cada lateral, e um na frente, e um módulo driver ponte H, para o acionamento dos dois motores CC. O controle do robô é executado por duas plataformas, a Raspberry Pi 3 B+ que realiza o processamento da imagem e o comando do servo motor da webcam e um Arduino Uno R3 para o controle dos sensores ultrassônicos e o acionamento dos motores, sistema embarcado no robô.

Na alimentação são usadas duas baterias LiPo de 11,1 V, 2200mAh, (Bateria 1) uma para a alimentação do Raspberry (necessita em média, de 2,5 A) e o Arduino e a outra usada na alimentação dos motores e acionamento (Bateria 2), de forma a evitar sinais de ruído provenientes dos motores.

Figura 36 - O protótipo do robô. (a) Vista superior. (b) Vista inferior (2 motores e 2 baterias)



(a) (b)

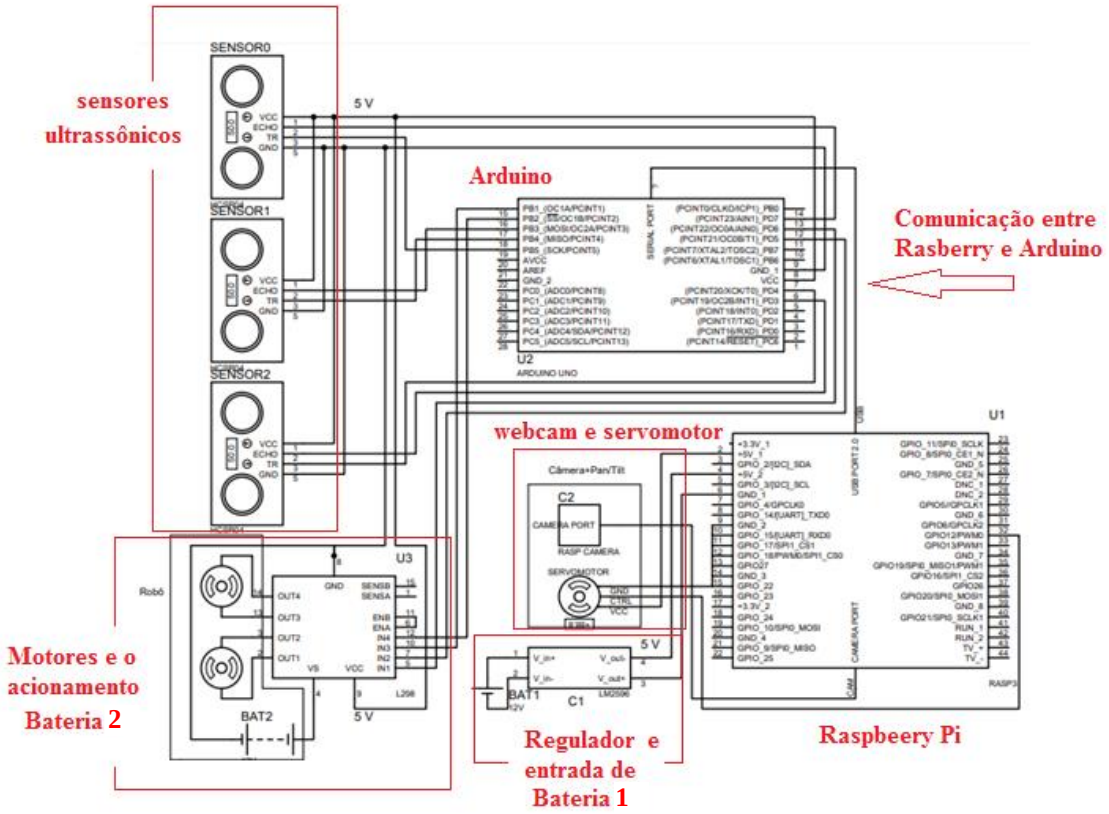
Fonte: Próprio autor.

7.3 Esquema de ligação do circuito

Antes de comentar o circuito esquemático é importante uma justificativa pelo uso de duas plataformas, a Raspberry Pi e a Arduino. No processamento de imagem a quantidade de informação gerada exige um processador potente e dedicado, por isso, foi utilizado a Raspberry Pi. A plataforma Arduino foi escolhida como *host*, pela sua simplicidade em hardware e programação. É importante observar também que, devido à pandemia, não foi possível realizar uma placa de circuito impresso para o protótipo.

Neste circuito esquemático estão presentes todos os componentes e as suas ligações, bem como a alimentação do circuito e as plataformas utilizadas no controle. Ressalta-se deste circuito que a Bateria 1 de Lipo de 11,1 V, está ligada ao regulador de tensão LM2596 para converter a tensão para 5V para o Raspberry Pi e os dispositivos sobre o seu controle, o Pan/Tilt que movimenta a webcam, bem como o Arduino. Enquanto a Bateria 2, alimenta os motores CC e o seu acionamento. No Apêndice A, apresenta-se o esquema de pinos ou GPIOs, (General Purpose Input/Output), do Raspberry Pi responsável por fazer a comunicação de entrada e saída de sinais digitais. O circuito esquemático para o robô foi construído no software PROTEUS, conforme mostrado na Figura 37.

Figura 37 – Esquema de ligação do circuito.



Fonte: Próprio autor.

Ressalta-se ainda, que no caso da regulação da tensão, cuja entrada de energia provém da Bateria 1, foi considerada a impedância do fio de ligação até a entrada da alimentação para os motores CC, para não haver discrepância entre as tensões.

O Pan/Tilt utilizado permite a varredura da câmera na direção horizontal, para isso utiliza o servo motor Micro servo 9g SG90, um atuador elétrico baseado em um motor CC e que apresenta um controle preciso de sua posição angular. O *flat cable* da webcam é ligado diretamente ao Raspberry Pi por meio *de um socket flat* de entrada apropriado, disponibilizado na placa.

Sob o controle do Arduino tem-se o módulo driver ponte H L298N e os três sensores ultrassônicos. A Bateria 2 é ligada aos motores via a ponte H, sendo que um *protoboard* foi utilizado para conectar os pinos GND (*ground*) dos componentes, assim como a alimentação, VCC = 5 V. As informações completas das ligações de todos os componentes e a pinagem sob o comando do Arduino são apresentadas no Quadro 11, do Apêndice A.

7.4 Comunicação serial entre Arduino e Raspberry Pi

A conexão do Raspberry Pi com o Arduino é realizada por meio de uma porta USB (Universal Serial Bus). O algoritmo do Arduino faz a aquisição das medidas de distância dos três sensores, recebe os dados da posição do objeto, proveniente do algoritmo do Raspberry Py, e aciona os motores. Caso o robô esteja próximo de um obstáculo e ao mesmo tempo estiver seguindo um objeto, primeiro ele desvia do obstáculo para posteriormente voltar a seguir o objeto.

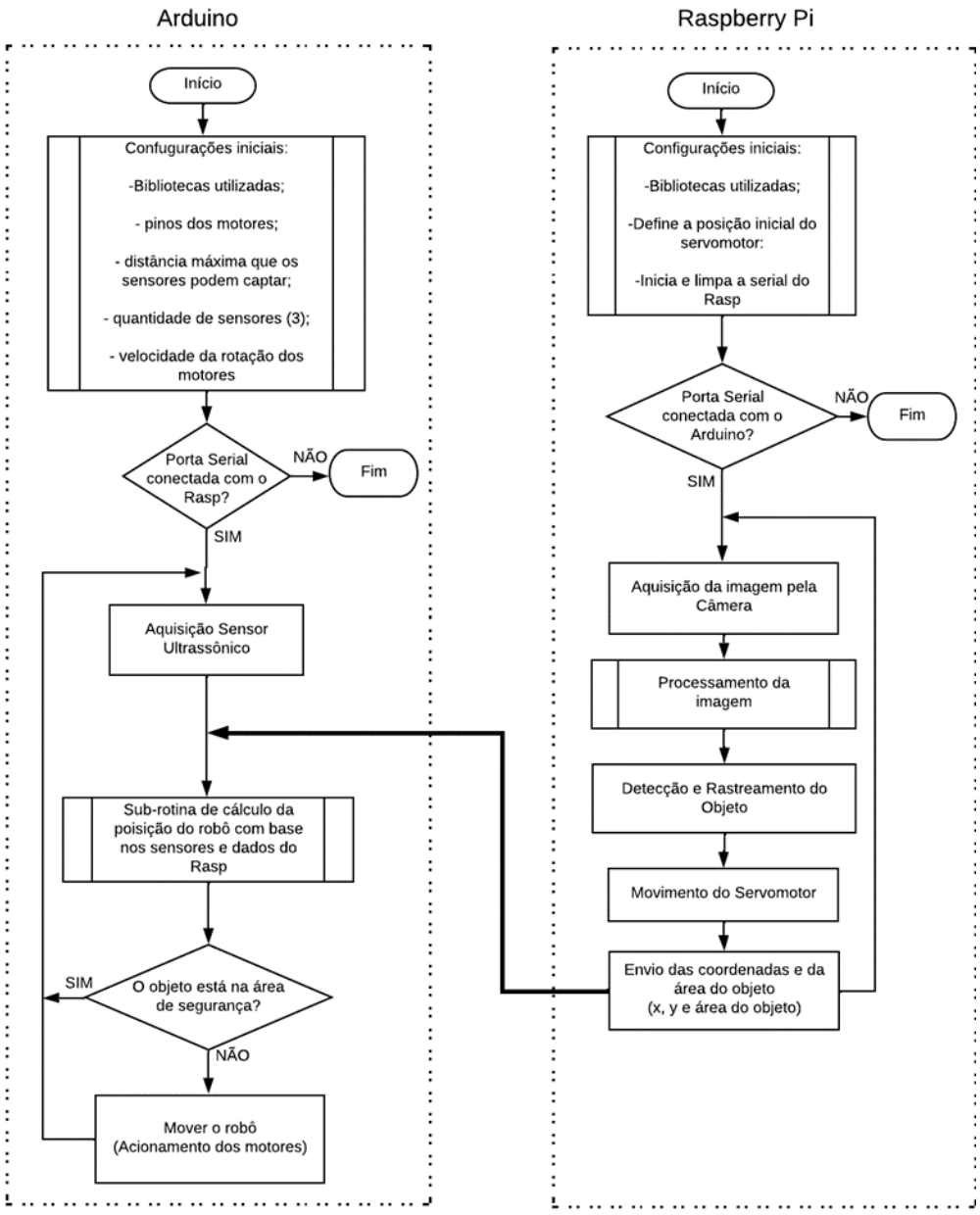
Para o desenvolvimento da programação dos sensores ultrassônicos e visando desviar dos obstáculos foi estabelecida uma distância de segurança de 10 centímetros, então, caso algum obstáculo se encontre a uma distância $\leq 11cm$, um nível lógico alto é recebido pelo Arduino, caso o valor seja $> 11cm$, um nível lógico baixo é recebido pelo Arduino. Com isso é possível criar uma tabela verdade, Tabela 8, com todas as combinações possíveis de saídas dos sensores e para cada combinação tem-se uma ação realizada pelo robô. Ressalta-se que $Cm[0]$ corresponde ao Sensor 0 (esquerda do robô), $Cm[1]$ corresponde ao Sensor 1, na frente e $Cm[2]$ corresponde ao Sensor 2, (direita do robô). O fluxograma do algoritmo implementado nas duas plataformas, é apresentado na Figura 38.

Tabela 8 – Tabela verdade com as saídas dos sensores e a ação realizada pelo robô.

Cm[0]	Cm[1]	Cm[2]	Ação Realizada
0	0	0	Leitura da Serial
0	0	1	Aciona os motores para a esquerda
0	1	0	Dá ré
0	1	1	Aciona os motores para a esquerda
1	0	0	Aciona os motores para a direita
1	0	1	Leitura da Serial
1	1	0	Aciona os motores para a direita
1	1	1	Dá ré

Fonte: Próprio autor.

Figura 38 – Fluxograma da comunicação serial entre Raspberry Pi e Arduino.



Fonte: Próprio autor.

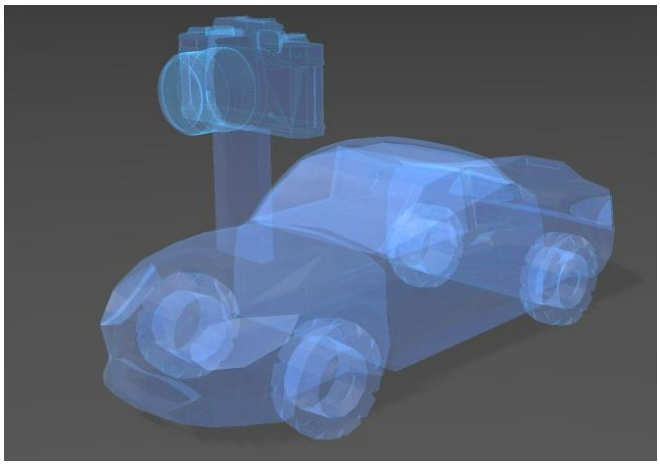
No fluxograma tem-se a programação do Arduino, que executa o cálculo dos sensores com base na posição do robô e também do acionamento dos motores, depois segue fazendo uma nova varredura de dados dos sensores ultrassônicos.

No caso do fluxograma implementado para o Raspberry Pi tem-se o processamento da imagem e a detecção e rastreamento do objeto desejado, dado no capítulo 5. Quando detectado o objeto, o servo motor passa a seguir o objeto, e a partir do momento em que o objeto atinge

um limite determinado do frame, o servomotor/câmera gira na direção especifica, para que o objeto volte à área de segurança.

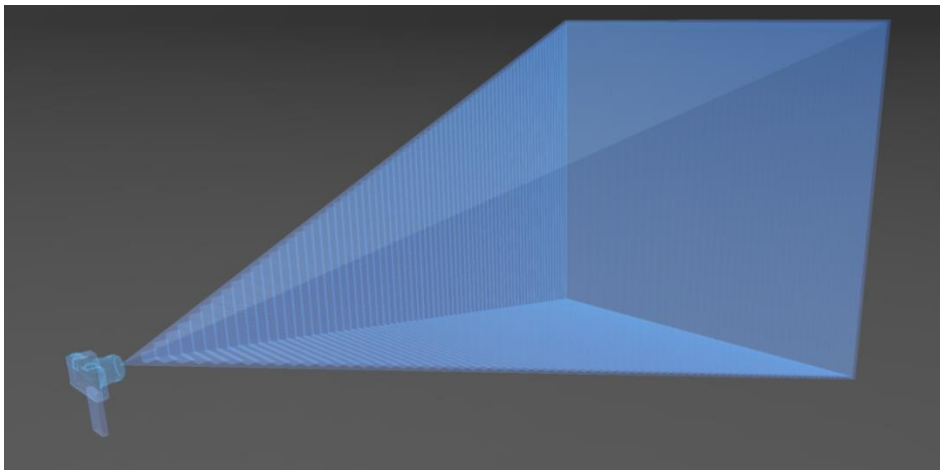
Para ilustrar melhor o funcionamento do servo motor, o posicionamento do objeto e o conceito de área de segurança, utilizou-se do software 3D Builder (MICROSOFT, 2023) para simular um ambiente em que é elaborado um modelo 3D do robô e da câmera, conforme mostrado na Figura 39. A visão do alcance visual da câmera, vista na tela do notebook é piramidal, como representado na Figura 40.

Figura 39 – Representação 3D do robô acoplado com a câmera.



Fonte: Próprio autor.

Figura 40 – Representação do alcance visual da câmera

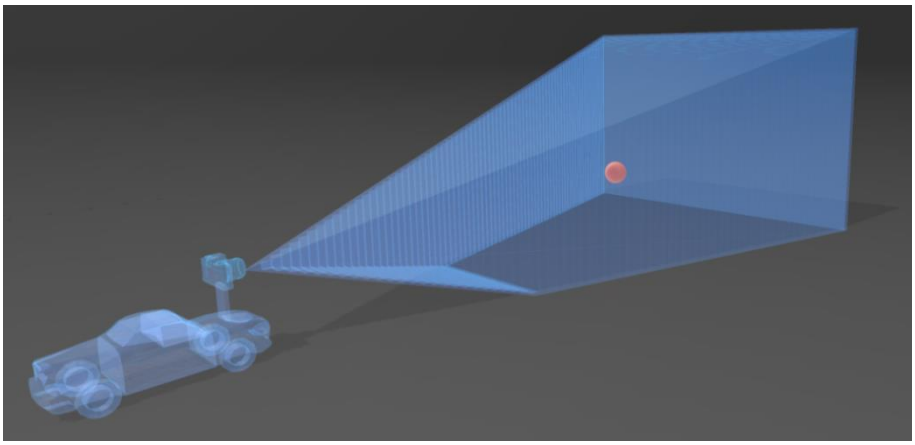


Fonte: Próprio autor.

Considerando a altura da câmera acoplada ao robô em relação ao solo, parte do alcance visual da câmera é limitada, por isso foi feita, na Figura 41, uma aproximação da diminuição

desse alcance usando o mesmo modelo 3D e para isso é adicionado no desenho uma bolinha na cor laranja, prevendo dar melhor contraste em relação ao ambiente 3D.

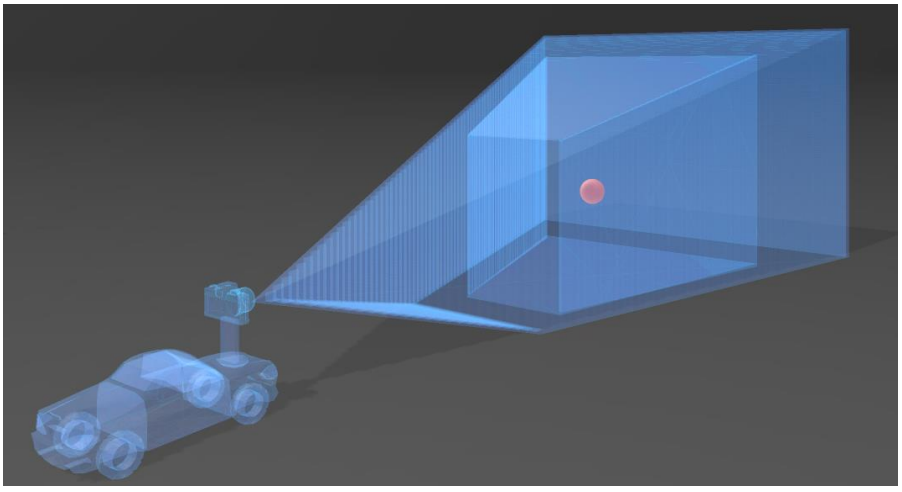
Figura 41 - Redução do alcance visual da câmera e a adição do objeto.



Fonte: Próprio autor.

Dentro desta área de visualização da câmera, foi estabelecida uma área de segurança para o robô perceber o objeto (objetivo principal é manter o objeto nesta área). Na Figura 42 ilustra-se, esta área de segurança.

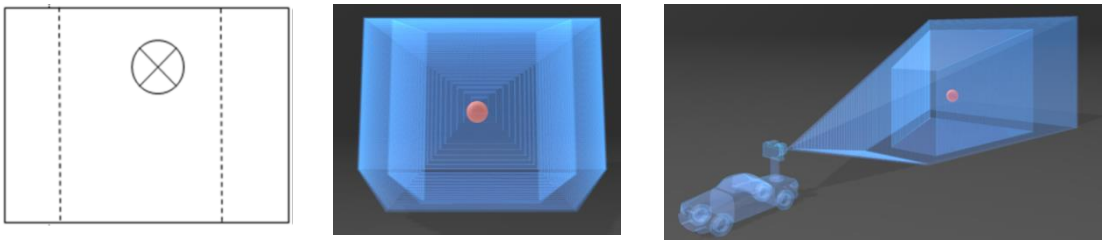
Figura 42 - Representação da área de segurança do objeto.



Fonte: Próprio autor.

Como forma de representar a movimentação do objeto em relação a área de segurança definida, faz-se três diferentes tipos de análise, sendo o objeto dentro da área de segurança, o objeto perto do limite da área de segurança e o objeto fora da área de segurança. Na forma visual têm-se as seguintes interpretações, apresentadas nas Figuras 43, 44 e 45.

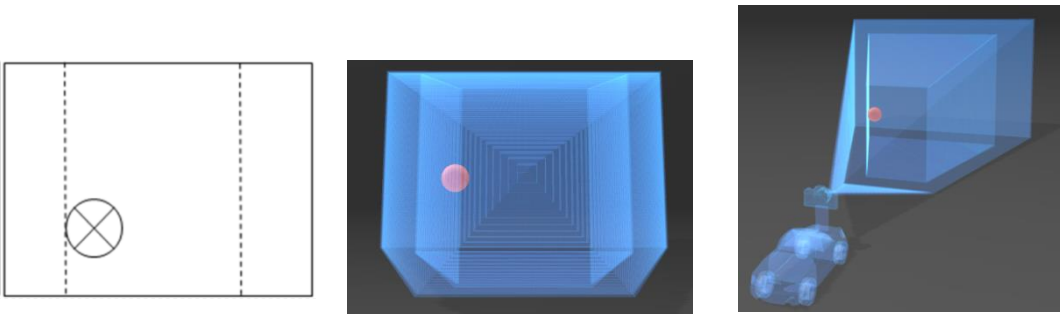
Figura 43 - Objeto dentro da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D.



(a) (b) (c)

Fonte: Próprio autor.

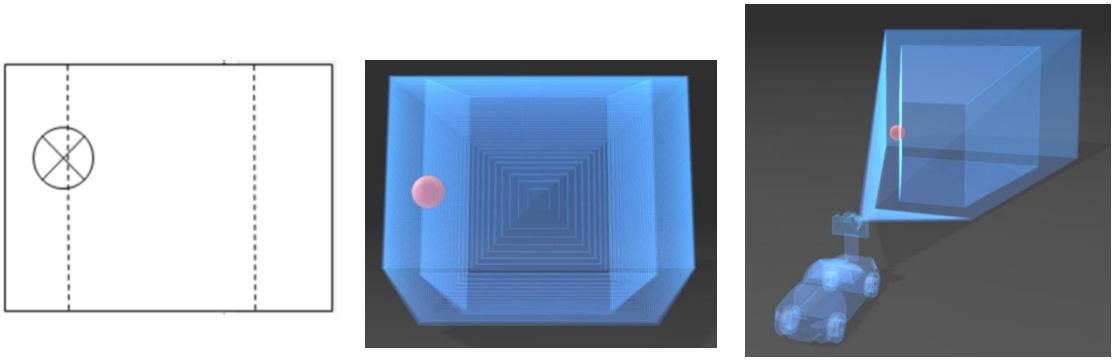
Figura 44 – Objeto perto do limite da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D.



(a) (b) (c)

Fonte: Próprio autor.

Figura 45 – Objeto fora da área de segurança (a) Bidimensional simples. (b) Bidimensional no ambiente 3D. (c) Tridimensional no ambiente 3D.



(a) (b) (c)

Fonte: Próprio autor.

Nas ilustrações, as linhas pontilhadas representam os limites da área de segurança do frame da imagem adquirida pela câmera. Somente quando o objeto ultrapassa esses limites de segurança, é que o Raspberry Pi envia um comando ao servo motor para girar na direção que vai colocar o objeto de volta nos limites de segurança pré-estabelecidos.

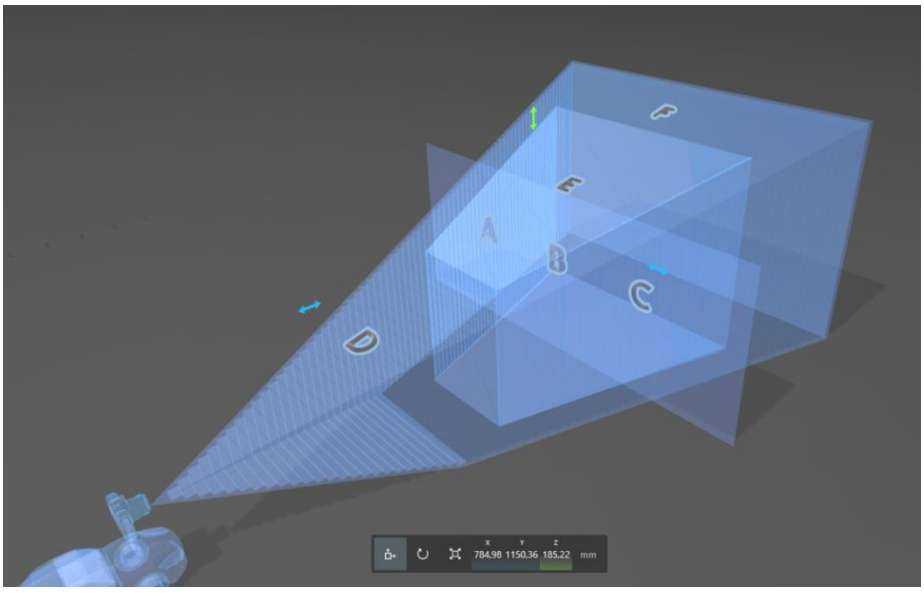
No próximo bloco do fluxograma para o Raspberry Pi, ocorre o envio das coordenadas e área do objeto. Depois disso, o Raspberry Pi volta a fazer nova aquisição de imagem pela câmera e assim sucessivamente.

7.5 Mapeamento do ambiente

Para construir o algoritmo de comunicação serial é necessário fazer o mapeamento do ambiente em que o robô está se deslocando.

Com o resultado da posição do objeto na tela e a dimensão da área obtida no processamento da imagem, deve-se determinar qual a distância que o objeto se encontra da tela, uma vez que quanto maior a área do objeto, mais perto ele vai estar, e ao contrário (quanto menor a área), mais distante ele vai estar. Para isso foi utilizado apenas um objeto para os testes com o robô, por isso suas dimensões permanecem constantes, e a varredura da câmera é considerada somente na horizontal. Sendo assim, é possível ilustrar o esquema de mapeamento do ambiente pela Figura 46.

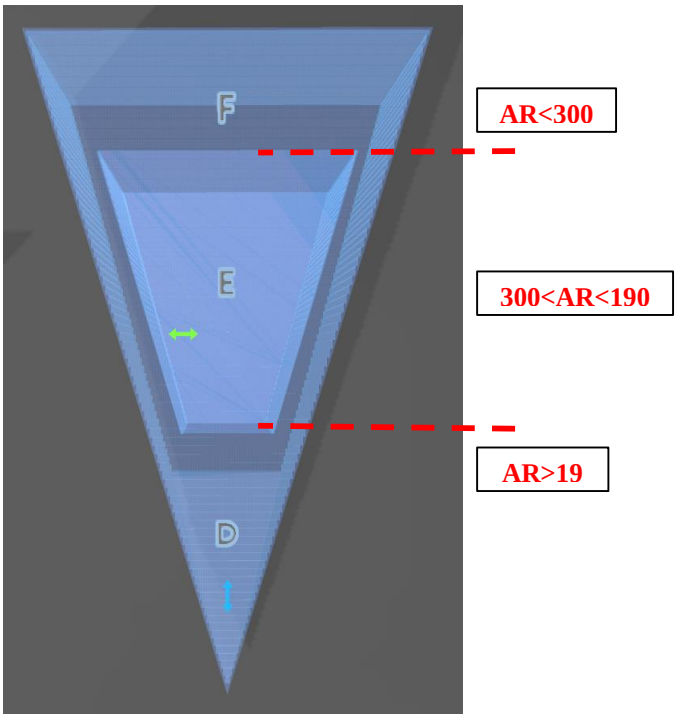
Figura 46 – Mapeamento do ambiente.



Fonte: Próprio autor.

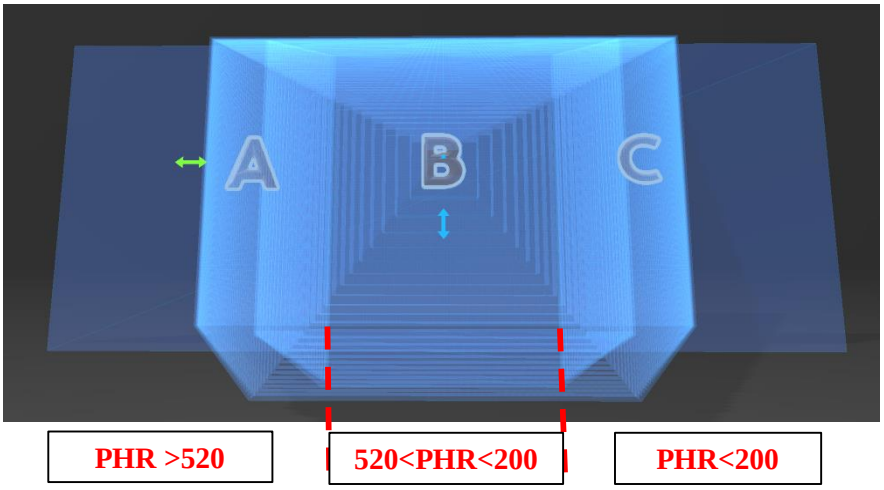
Para facilitar a compreensão, divide-se em mapeamento da área e o mapeamento da posição horizontal do objeto. O mapeamento da área do objeto é representado pela variação da profundidade em que o objeto se encontra em relação à câmera, Figura 47, e o mapeamento da posição horizontal é representado pela variação da distância horizontal do objeto, apresentado na Figura 48, sendo AR: área relativa do objeto e PHR: Posição Horizontal Relativa do objeto. O termo “relativo” se refere aos valores adquiridos por meio dos testes realizados e não remete à unidade de medida como metro ou centímetro.

Figura 47 – Visualização superior do ambiente mapeado.



Fonte: Próprio autor.

Figura 48 – Visualização frontal do ambiente mapeado.



Fonte: Próprio autor.

Analisando matricialmente as dimensões, tanto da área quanto da posição do objeto e transformando as dimensões relativas para facilitar o processamento do algoritmo, chega-se na equação a seguir, Equação (13),

$$\begin{bmatrix} F \\ E \\ D \end{bmatrix} * \begin{bmatrix} A & B & C \end{bmatrix} = \begin{bmatrix} FA & FB & FC \\ EA & EB & EC \\ DA & DB & DC \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (13)$$

Concluindo, os dados adquiridos provenientes do Raspberry Pi, caso sejam lidos os valores 1, 4 ou 7, o robô vira para a esquerda, caso sejam lidos os valores 3, 6 ou 9, vira para a direita, caso o valor lido for 2, o robô se move para frente, caso lido o valor 8 o robô dá ré, conforme resumido na Tabela 9. Caso o valor lido pelo Arduino seja 5 o robô fica parado, pois a sessão da área 5 representa a área de segurança tanto em relação à posição horizontal do objeto, quanto da área dele, enquanto nenhum dos sensores ultrassônicos acusa ter obstáculos por perto.

Tabela 9 – Sintetização das ações do robô.

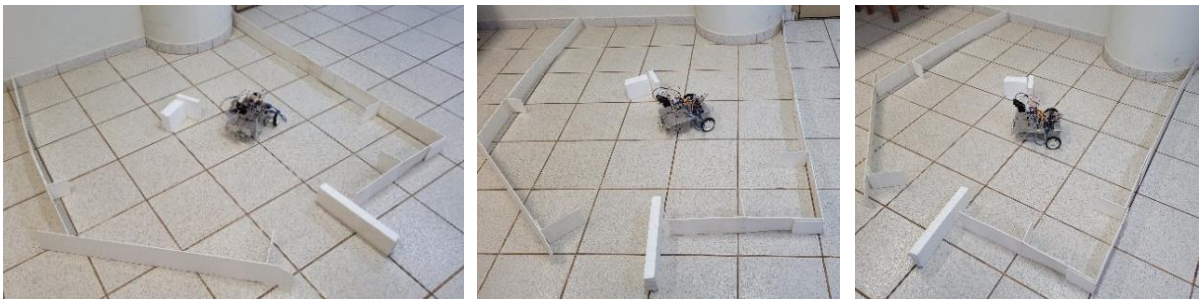
Ação do Robô	Nível lógico dos sensores Ultrassônicos			Valores lidos na Serial		
	Cm[0]	Cm[1]	Cm[2]			
Vira para a esquerda	0	0	1	1	4	7
	0	1	1			
Vira para a direita	1	0	0	3	6	9
	1	1	0			
Dá ré	0	1	0	8		
	1	1	1			
Segue em Frente	-	-	-	2		
Fica parado	-	-	-	5		

Fonte: Próprio autor.

7.6 Resultados dos testes com o protótipo

Os testes com o protótipo do robô foram realizados para um percurso com obstáculos, observando primeiramente, o comportamento dos sensores ultrassônicos, conforme mostrado na Figura 49, vídeo tomado de fora do robô.

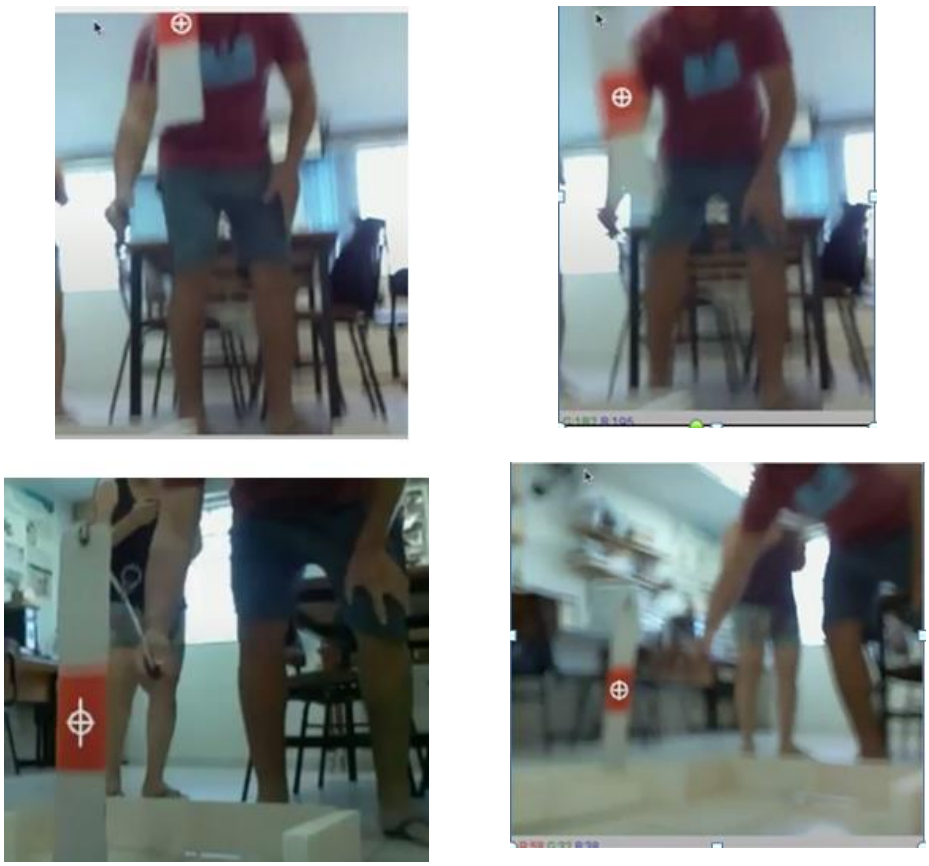
Figura 49 – Testes dos sensores ultrassônicos do robô.



Fonte: Próprio autor.

A visão do robô e o seu processamento foram observados por meio do aplicativo VNC Viewer no celular. A tela do celular passou a apresentar o que estava sendo transmitido no Raspberry Pi, desta forma tem-se na Figura 50 uma sequência de imagens demonstrando a visão do robô em tempo real, e a detecção e o rastreamento do objeto de cor alaranjada que estava sendo movimentado de um lado para o outro.

Figura 50 – Detecção e rastreamento sob a visão do robô.



Fonte: Próprio autor.

Ao invés de utilizar uma bola anti-stress para o teste final, optou-se por utilizar uma régua de cor laranja, pois era o objeto que tinha o maior contraste dentre as outras cores presentes no ambiente do laboratório, e possuía um orifício que permitia conectá-lo a uma vara de pesca e movimentá-lo à distância, facilitando a detecção e o rastreamento pelo robô.

Embora poucos, estes resultados demonstram que houve o domínio da técnica de detecção e rastreamento do objeto de cor e, nos testes com os sensores ultrassônicos, foi observado que a programação estava correta, uma vez que houve o deslocamento do robô e o desvio dos obstáculos e a detecção do objeto e o seu rastreamento. Visando dar maior confiabilidade ao sistema de hardware e software projetado, uma maior quantidade de testes com o robô pode ser realizada.

Por meio dos testes com o robô foi observado também que o algoritmo de detecção e rastreamento, devido à taxa de aquisição de frames de cerca de 30 frames por segundo, no processamento da imagem ocupa a toda a memória do *Raspberry*, problema que foi superado

zerando o acúmulo de informações na memória de tempos em tempos, evitando o travamento da detecção e rastreamento pelo robô.

8 CONCLUSÕES

Neste trabalho foi realizada a detecção e o rastreamento de objeto de determinada cor, em vídeo, e em tempo real, com aplicação para deslocamento em um ambiente de um robô móvel, utilizando para o processamento das imagens a biblioteca OpenCV e a plataforma de controle *Raspberry Pi*.

Inicialmente, para os testes com o algoritmo de processamento de imagem desenvolvido, fez-se a detecção e o rastreamento de dois objetos, para uma bola anti-stress e uma garrafa térmica na cor azul, podendo-se com isso adquirir o aprendizado das diversas técnicas envolvendo o processamento de imagem e dos recursos da biblioteca OpenCV utilizada.

No processamento de imagem foi observado que nem todas as funções da biblioteca OpenCV, tais como a função cv2.CamShift(), não se mostrou uma técnica muito prática uma vez que era necessário determinar a posição inicial do objeto, para o seu uso. Os resultados obtidos também não foram satisfatórios uma vez que o algoritmo não conseguia rastrear o objeto por um período prolongado, perdendo, em pouco tempo, a sua referência.

Pelo fato de a garrafa térmica ter dois pontos na cor azul, foi notado também a detecção de dois centros geométricos de uma vez. Isso gerou um problema no momento do rastreamento, uma vez que dificulta o cálculo de forma exata, da posição do objeto na tela por conta da existência de mais de um contorno simultâneo. O uso da garrafa térmica demonstra que é possível a detecção de mais de um objeto simultaneamente, mas não é recomendado.

Na construção do protótipo do robô foi necessário o desenvolvimento do hardware embarcado em duas plataformas de controle, uma vez que o processamento da imagem exige uma plataforma com bons recursos computacionais, como o *Raspberry Pi* e na parte de aquisição de dados e acionamento dos motores foi usado o Arduino Uno R3. As duas plataformas se comunicam por meio de um programa elaborado na linguagem Python.

Visando os testes práticos com o protótipo do robô para a detecção e o rastreamento de um objeto no ambiente e consequentemente o seu deslocamento, foi usado um objeto de cor laranja, distinta das cores existentes no ambiente do laboratório, de forma a facilitar a detecção e o rastreamento (pelo robô). Com isso obteve-se resultados de detecção e rastreamento satisfatórios, apresentando menor perda de referência do objeto, ou seja, o robô foi capaz de seguir o objeto e desviar do obstáculo quando necessário. Também foi observado que respeitou

a prioridade dos sensores ultrassônicos sobre a câmera, estabelecida no programa, no caso de estar na frente de obstáculos, para depois realizar o acompanhamento do objeto.

Nos testes com o robô foram obtidas algumas fotografias tremidas indicando que a causa deve ter sido a vibração dos motores. Mas, este problema não interferiu na detecção e o rastreamento do objeto, uma vez que a metodologia para detecção e rastreamento baseava-se na cor do objeto, não dependendo do formato ou distorção do frame. O algoritmo ainda é capaz de calcular o seu centro geométrico de forma a realizar todos os processos com a imagem.

Observando os resultados, conclui-se que os objetivos da detecção e o rastreamento do objeto de determinada cor foram alcançados, e também o deslocamento do robô sem bater em obstáculos, proporcionando um bom conhecimento no tema de visão computacional que podem ser aprofundados, por meio da aplicação no algoritmo, de outras funções da biblioteca OpenCV. Um resumo deste trabalho foi publicado no XXXII Congresso de Iniciação Científica da Unesp - CIC 2020.

Para futuro trabalhos e pensando em aperfeiçoar o hardware embarcado no robô, pode-se usar a Inteligência Artificial para determinar as melhores entradas para o espaço de cores no momento da detecção do objeto, podendo com isso oferecer resultados mais eficazes para a detecção e rastreamento pelo protótipo.

9 REFERÊNCIAS

ALFAMÍDIA, Modelos de cor: HSB ou HSV. Porto Alegre, 2019 Disponível em: http://www.alfamidia.com.br/15-01-2019-modelos-de-cor-hsb-ou-hsv/. Acesso em: 10 fev. 2022

ARDUINO. What Is Arduino?, 2018. Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 10 fev. 2022

ARIAS, A. O que você deve saber sobre baterias Lipo, o combustível do VANT. **DRONENG**, 2016. Disponível em: https://blog.droneng.com.br/baterias-lipo/. Acesso em: 10 fev. 2022

BRAGA, E. S.; COSTA, R. M. R. Imagem e Movimento: Implementando o Movimento Autônomo Através do Arduino e OpenCV. **Caderno de Estudos em Sistemas de Informação**, Juiz de Fora, v. 2, n. 2. 2015. Disponível em: https://seer.uniacademia.edu.br/index.php/cesi/article/view/522 Acesso em: 4 nov. 2023

BALLARD, D. H.; BROWN, C. M. **Computer Vision.** Prentice-Hall *Inc.*: New Jersey, 1982. p. 539. ISBN 0131653164

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. [S.l.]: O'Reilly Media, Inc., 2008 ISBN: 9780596554040

BLOG ELETROGATE. Guia definitivo de uso da ponte H L298n. 2020. Disponível em: <https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n/>. Acesso em: 02 nov 2022.

CAVALCANTI, M. Desvendando o mistério: sRGB , Adobe RGB , ProPhoto RGB. 2020 Disponível em: https://marcellocavalcanti.com.br/desvendando-o-misterio-srgb-adobe-rgb-prophoto-rgb-quarentenatalks/. Acesso em: 20 dez 2023.

E-GIZMO. Ultrasonic sonar module Disponível em https://www.e-gizmo.net/oc/kits%20documents/US-100/ultrasonic%20sonar%20module%201r0.pdf. Acesso em: 10 fev. 2022.

ENCYCOLORPEDIA. FFA500 Disponível em: https://encycolorpedia.pt/ffa500#:~:text=A%20cor%20orange%20%2F%20Laranja%2C%20com,%25%20verde%20e%200%25%20azul. Acesso em: 03 nov. 2023

ESTADÃO, 2020. Conheça o robô inteligente que combate lesmas nas plantações. Disponível em: https://summitagro.estadao.com.br/tendencias-e-tecnologia/robo-inteligente-combate-lesmas-plantacoes/#:~:text=A%20iniciativa%2C%20batizada%20de%20Slugbot%20e%20resultado%20da%20união%20de,o%20valor%20comercial%20das%20culturas. Acesso em: 10 fev. 2022

FORSYTH, D. A.; PONCE, J**. Computer Vision: A Modern Approach**. Prentice Hall Professional Technical Reference, 2002. ISBN: 978013085198-7. Acesso em: 03 nov. 2023

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3ed. [s.l.]: Pearson, 2009. p. 624. ISBN: 9788576054016.

HEINEN, F.; OSÓRIO, F. "HyCAR - A Robust Hybrid Control Architecture for Autonomous Robots". In: HIS 2002 – Proc. Of Hybrid Intelligent Systems, Santiago, Chile. SoftComputing Systems - Amsterdam: IOS Press. V. 87, p. 830-840. 2002.

HELAL, L.2015. Disponível em: https://lucasghelal.medium.com/disco-crom%C3%A1tico-hsv-db3cd81c9a80. Acesso em: 20 jan.2021

HELERBROCK, Rafael. "Espectro eletromagnético"; Brasil Escola. Disponível em: https://brasilescola.uol.com.br/fisica/espectro-eletromagnetico.htm. Acesso em: 10 nov 2023.

HSV. WIKIPÉDIA. Disponível em: https://pt.wikipedia.org/wiki/HSV. Acesso em: 30 mar. 2020.

MASUTANI, H.V.; ARTERO, A. O. ; DE ALMEIDA, L. L; DA SILVA, F.Assis. RECONHECIMENTO DE OBJETOS COLORIDOS E MÃOS USANDO CORES E FORMAS. (2013). *Colloquium Exactarum. ISSN: 2178-8332*, *5*(1), 01 -11. Disponível em: https://revistas.unoeste.br/index.php/ce/article/view/862. Acesso em: 30 mar. 2020.

MAKERHERO Regulador de Tensão LM2596 Conversor DC-DC Step Down., [s.d.]. Disponível em: https://www.filipeflop.com/produto/regulador-de-tensao-lm2596-conversor-dc-dc-step-down/. Acesso em: 10 fev. 2022

MICRORED MICROREDUTORES E MOTORES LTDA. **MICROREDUTOR Modelo P-VDC.** [S. l.]: MICRORED MICROREDUTORES E MOTORES LTDA., 2010. Disponível em: https://microred.com.br/CATALOGO-MOD_P-VDC.pdf. Acesso em: 15 nov. 2019

MICROSOFT . 3D Builder . Disponível em: https://apps.microsoft.com/detail/9WZDNCRFJ3T6?hl=pt-br&gl=BR . Acesso em: 04 mar 2023

NIKU, S. B. **Introduction to Robotics**: Analysis, Control, Applications. 2 ed. New Jersey: Wiley, 2010. p. 480. ISBN: 9780470604465.

OLIVEIRA, B. A. S.; ASSIS, S. S.; RAMOS, I. A.; MARCON, M. Detecção e Rastreamento de objetos coloridos em vídeo utilizando o OpenCV. Bambuí. **Anais de Resumos Expandidos**, 2014. p. 01-05. Disponível em: https://www.bambui.ifmg.edu.br/jornada_cientifica/2014/resumos/Info/Detecção%20e%20Rastreamento%20de%20Objetos%20coloridos%20em%20vídeo%20utiliz.pdf. Acesso em: 25 mar. 2020

OLIVEIRA, B. A. S.; ASSIS, S. S.; RAMOS, I. A.; MARCON, M. Detecção e Rastreamento de objetos coloridos em vídeo utilizando o OpenCV. Bambuí. **Anais de Resumos Expandidos**, 2014. p. 01-05. Disponível em:

https://www.bambui.ifmg.edu.br/jornada_cientifica/2014/resumos/Info/Detecção%20e%20Rastreamento%20de%20Objetos%20coloridos%20em%20vídeo%20utiliz.pdf. Acesso em: 25 mar. 2020

OLIVEIRA, S. C. A.; OLIVEIRA, S. C. A; MOREIRA, H. R.. Construção de um robô móvel utilizando Raspberry Pi integrado a um sistema de visão computacional para a movimentação baseada no reconhecimento de cor. In: MOSTRA NACIONAL DE ROBÓTICA, 4. 2014, São Carlos -SP. **Anais [...].**: MNR, 2014. v. 01, p. 419-422. Disponível em: http://www.mnr.org.br/wp-content/uploads/2018/02/Anais_2014.pdf. Acesso em: 06 abr. 2020

OPENCV. **Open Computer Vision Library**. 2019. About Disponível em: https://opencv.org/about/. Acesso em: 06 abr. 2020

OSÓRIO, F. S.; BORDIGNON, G. N. Extração de informações relevantes de imagens voltadas para aplicações em robótica móvel. In: MOSTRA NACIONAL DE ROBÓTICA, 04, 2014, São Carlos - SP. **Anais [...]:** Mnr, 2014. v. 01, p. 440-443. Disponível em: http://www.mnr.org.br/wp-content/uploads/2018/02/Anais_2014.pdf. Acesso em: 01 abr. 2020

PASSARELLI, L. Aplicação de visão computacional com OpenCV. **EMBARCADOS**, 2017 Disponível em: https://www.embarcados.com.br/aplicacao-de-visao-computacional-com-opencv/. Acesso em: 30 mar. 2020

RAPIDTABLES. **HSV to RGB color conversion.** Disponível em: https://www.rapidtables.com/convert/color/hsv-to-rgb.html. Acesso em: 03 nov. 2023

RASPBERRY PI 4 pinout. **DYI LIFE**, [s.d.]**.** Disponível em: https://www.the-diy-life.com/wp-content/uploads/2021/05/Raspberry-Pi-4-Pinout.png. Acesso em: 10 fev. 2022

RASPBERRY PI Documentation. About the Camera Module. Disponível em: https://www.raspberrypi.com/documentation/accessories/camera.html. Acesso em: 10 fev. 2021

RASPBERRY PI FOUNDATION. **Teach, Learn, and Make With Raspberry Pi**, Raspberry Pi, 2019. Disponível em: https://www.raspberrypi.org/. Acesso em: 11 out. 2019

SANTOS, J. P. S. **Sistema embarcado para rastreamento de imagens**, 2009. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica) - Faculdade de Tecnologia-UNB, Brasília, 2009. Disponível em: https://bdm.unb.br/handle/10483/982 . Acesso em: 03 mar. 2020

SCHUEROFF, J. **Desenvolvimento de Um Sistema de Visão Artificial Para Um Robô Explorador**. Trabalho de Conclusão de Curso (Bacharelado em Tecnologia da Informação e Comunicação) – Universidade Federal de Santa Catarina, Araranguá, 2013. Disponível em: https://repositorio.ufsc.br/handle/123456789/131173. Acesso em: 03 mar. 2020

SICILIANO, B.; KHATIB, O.; KRÖGER, T. **Springer Handbook of Robotics**. Berlin: springer, 2008. ISBN 9783540239574. Acesso em: 03 mar. 2021

SILVA, B.R. de A. **Sistema de contagem automática de objetos utilizando processamento digital de Imagens em dispositivos móveis** Dissertação de Mestrado**.** Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido. PPGCC – Mossoró-RGN. 2014

STMICROELETRONICS**.** L298 Dual Full Bridge Driver, [s.d.].2000 Disponível em: https://www.st.com/en/motor-drivers/l298.html. Acesso em 10 fev. 2022

SZELISKI, R. **Computer Vision**: Algorithms and Applications. 1 ed. Londres: Springer, 2010. p. 812. Doi: https://doi.org/10.1007/978-1-84882-935-0ISBN: 978-1-84882-935-0. Acesso em: 03 nov. 2023

SUNFOUNDER, Ultrassonic Module. 2019. Disponível em: http://wiki.sunfounder.cc/index.php?title=Ultrasonic_Module. Acesso em: 10 fev. 2022.

VASCONCELOS, P. Quais são as melhores IDE para Python? Confira a lista, 2017. Disponível em: https://paulovasconcellos.com.br/quais-s%C3%A3o-as-melhores-ide-para-python-confira-a-lista-279b54bef301.Acesso em: 10 fev. 2022

VIEIRA, M. A. C.; GONZAGA, A. **SEL0339 Introdução à Visão Computacional-2020.** [s.d.]. Apresentação do Power Point. Disponível em: https://edisciplinas.usp.br/pluginfile.php/5549058/mod_resource/content/1/Aula%201%20-%20Introducao%20a%20Visao%20Computacional.pdf Acesso em: 10 fev. 2022

YILMAZ, A.; JAVED, O.; SHAH, M**. Object tracking:** A Surveys, Florida: University of Central Florida, v. 38, n. 4, p. 45, dez. 2006. Association for Computing Machinery (ACM). Disponível em: http://doi.acm.org/10.1145/1177352.1177355. Acesso em: 25 mar. 2020

WARELINE, 2018. Entenda como a Robótica na Medicina vai revolucionar os procedimentos cirúrgicos. Disponível em: https://www.wareline.com.br/wareline/noticias/como-funciona-a-robotica-na-medicina/. Acesso em: 20 fev. 2022

APÊNDICE A- Conexões ao Arduino

Quadro 2 - Conexões sob o controle do Arduino.

Componente	Saída	Destino (Arduino)
Ponte H	In1	Pcint22 = Pino 6 Arduino
	In2	Pcint21 = Pino 5 Arduino
	In3	Pcint1 = Pino 9 Arduino
	In4	Pcint2 = Pino 10 Arduino
	Out1	Motor 1
	Out2	Motor 1
	Out3	Motor 2
	Out4	Motor 2
	Vs	Positivo da Bateria
	VCC	Barramento VCC
	GND	Barramento GND
Sensor 0	Echo	Pcint23 = Pino 7 Arduino
	Trigger	Pcint5 = Pino 13 Arduino
	VCC	Barramento VCC
	GND	Barramento GND
Sensor 1	Echo	Pcint3 = Pino 11 Arduino
	Trigger	Pcint4 = Pino 12 Arduino
	VCC	Barramento VCC
	GND	Barramento GND
Sensor 2	Echo	Pcint19 = Pino 3 Arduino
	Trigger	Pcint20 = Pino 4 Arduino
	VCC	Barramento VCC
	GND	Barramento GND

Fonte: Próprio autor.

APÊNDICE B – Resumo publicado no XXXII Congresso de Iniciação Científica da Unesp - CIC 2020.

DETECÇÃO E RASTREAMENTO DE OBJETOS DE COR USANDO OPENCV E RASPBERRY PI VISANDO APLICAÇÃO EM ROBÔS MÓVEIS.

Vinícius Monzem Garcia, Suely Cunha Amaro Mantovani, Câmpus de Ilha Solteira, Curso de Engenharia Elétrica, viniciusmonzemg@gmail.com, Bolsista PET-MEC.

Palavras Chave: Detecção de objeto, Rastreamento, Robôs Móveis.

Introdução

A detecção e o rastreamento de objetos fazem parte dos Sistemas de Visão Computacional muito utilizado na área da robótica e navegação de veículos autônomos. Com este recurso um robô, ou mesmo um veículo, "aprende a enxergar" usando receptores artificiais e, com isso, fazer tarefas, cada vez mais complexas, como a sua movimentação sem bater em obstáculos. O aumento do interesse em algoritmos de rastreamento[2] advém da disponibilidade do alto poder computacional, de vídeos de câmeras com alta qualidade e, principalmente, do custo reduzido. A análise de vídeo é realizada pela detecção e rastreamento de objetos em movimento, no quadro, pela cor (ou forma), visando o seu reconhecimento.

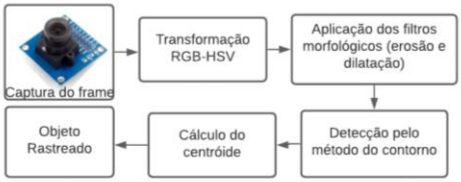
Objetivo

Realizar a detecção e rastreamento de objetos de cor, em tempo real, embarcado em uma plataforma Raspberry Pi, em um robô móvel visando sua movimentação autônoma em um ambiente.

Material e Métodos

Baseado no diagrama de blocos da Figura 1 e para sistemas em tempo real, foi utilizada a OpenCV[1] (Open Source Computer Vision), biblioteca com uma diversidade de funções genéricas e específicas para tratamento de imagens, para realizar o processo de detecção e rastreamento de objetos coloridos em frames (ou quadros) adquiridos por uma webcam de 0,92 megapixels e resolução de 1280 x 720, de um notebook.

Figura 1. Diagrama de blocos da detecção e rastreamento.

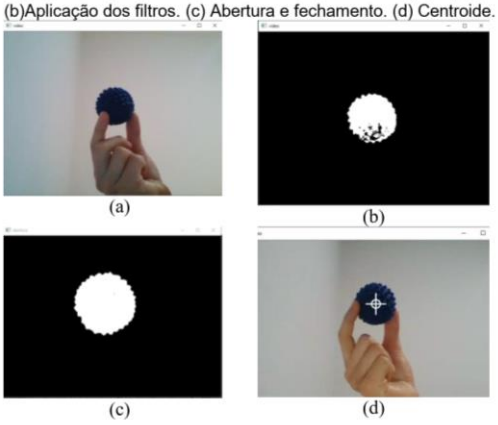


Para realizar este processo as imagens capturadas (frames) foram convertidas do sistema de cores RGB para o HSV, na segmentação faz-se uso de técnicas como limiarização, dilatação, erosão, e bibliotecas OpenCV. Posterior ao método da erosão e dilatação tem-se a detecção dos contornos dos objetos que aparecem na imagem, calculando o seu centro geométrico, e a adição de um marcador.

Resultados e Discussão

O algoritmo de detecção e rastreamento foi programado na linguagem Python. Foram feitos os testes para a detecção de diferentes objetos e cores, mas neste mostram-se na Figura 2, os melhores resultados obtidos para uma sequência de quadros para o rastreamento de uma bolinha anti-stress na cor azul. Foram testados mais 2 métodos de rastreamento, o CamShift e o Moments, com resultados inferiores ao método dos contornos.

Figura 2. Detecção e o rastreamento em vídeo (a)Captura. (b)Aplicação dos filtros. (c) Abertura e fechamento. (d) Centroide.



Conclusões

O uso de visão artificial nos sistemas robóticos proporciona a percepção de ambientes externos, mas deve-se atentar para o uso do computador embarcado no robô, neste caso o Raspberry Pi, a taxa de frames por segundo (FPS) usada no processamento, e a resolução da webcam que podem limitar algoritmos mais complexos. Os resultados da detecção serão utilizados no controle de navegação de um protótipo de um robô.

[1]OPENCV. Library. 2019. Disponível em: <https://github.com/opencv/opencv/wiki>. Acesso em: 16 Set. 2020.
[2] OLIVEIRA, B. A. S.; ASSIS, S. S. de; RAMOS, I. A.; MARCON, M.. Detecção e rastreamento de objetos coloridos em vídeo utilizando o OpenCV. Disponível em: https://www.bambui.ifmg.edu.br/jornada_cientifica/2014/resumos/Info/. >Acesso em: 25 mar. 2020.