

ARQUITETURA PARA NAVEGAÇÃO AUTÔNOMA UTILIZANDO DRONES – BCC

Matheus Mahnke, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil
matmahnke@furb.br, dalton@furb.br

Resumo:

Palavras-chave: Drone. Raspberry Pi. Navegação autônoma.

1 INTRODUÇÃO

Todos os anos processos antes feitos por humanos são automatizados, em qualquer área e qualquer campo da ciência, a tecnologia vem avançando para reduzir esforços humanos. Segundo Corrêa (2020, p. 16), “essa tecnologia trouxe mudanças à sociedade, tal como novos modelos de trabalho e novas profissões”.

Diante do crescente uso da tecnologia, a popularização do drone chama atenção, pois com ele é possível acessar locais inacessíveis de forma terrestre. Por conta disso, a cada ano os drones vêm ganhando mais popularidade. Dentre as opções, os quadrotoros são os mais comuns. De acordo com Lugo e Zell (2014), quadrotoros são uma escolha muito popular por conta de sua robustez, mecânica simples, baixo peso e tamanho pequeno.

Os drones geralmente são equipados com diversos sensores, como altímetro, acelerômetro e giroscópio, porém, apesar de bastante preciso, durante uma navegação baseada nesses sensores pode acontecer acúmulo de erros, aumentando o desvio no decorrer da navegação. Sensores mais modernos ajudam a solucionar esse problema, porém acabam sendo inviabilizados pelo seu alto valor comercial. Segundo Mur-artal e Tardós (2017), dentre as diferentes modalidades de sensores, as câmeras são baratas e podem fornecer informações valiosas do ambiente permitindo o reconhecimento robusto e preciso do local.

De acordo com Martins, Ramos e Mora-Camino (2018), a vantagem de se trabalhar somente com navegação baseada em Visão Computacional é que a solução é simples e tem um baixo custo. Sendo assim, a navegação baseada em câmeras se torna uma opção viável, sendo possível utilizar técnicas de processamento de imagem para detectar e desviar de obstáculos.

Entretanto, ao se utilizar técnicas de processamento de imagem em tempo real, o drone se torna dependente da conexão com um equipamento capaz de fazer esse processamento. Segundo Lugo e Zell (2014), a comunicação por meio de uma rede sem fio limita a distância de trabalho do sistema e introduz um atraso entre as informações de dados dos sensores e comandos de controle. Em contrapartida, novos modelos de microcontroladores possuem essa capacidade de processamento e podem ser levados a bordo do drone, de forma portátil, independente de conexão.

O drone pode executar um plano de voo, tanto controlado manualmente, com uma rota fixa baseada em sensores quanto baseado em reconhecimento de imagens. Contudo, nesses casos, ainda é necessário a ação humana na criação das rotas. De acordo com Corrêa (2020, p. 20), para voar autonomamente para uma determinada coordenada geográfica, é necessário um dispositivo receptor de sistemas que fornecem posicionamento geográfico espacial. Nesse contexto, o Global Positioning System (GPS) é uma importante ferramenta para a navegação autônoma. Com ele é possível determinar o destino e definir planos de voo em ambientes não conhecidos previamente, aumentando sua aplicabilidade comercial.

Diante disso, este trabalho **propõe** uma arquitetura de navegação baseada em GPS, aliado a imagens de câmera para o reconhecimento e desvio de obstáculos em voo, com processamento a bordo resultando em uma navegação totalmente autônoma.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado os conceitos para o entendimento do projeto realizado. A seção 2.1 descreve as principais características do Raspberry Pi modelo 3 A+. Na seção 2.2 é dado um breve resumo sobre a biblioteca Pyarrow. A seção 2.3 aborda a Lei dos cossenos. Na seção 2.4 é apresentado o modelo MiDaS. Por fim, na seção 2.5 são listados os trabalhos correlatos.

2.1 RASPBERRY PI 3 A+

O Raspberry Pi é um computador de baixo custo, do tamanho de um cartão de crédito, que pode ser conectado a um monitor ou televisão. É um pequeno dispositivo capaz que permite que pessoas de todas as idades explorem a computação e aprendam a programar em linguagens como Scratch e Python. Ele é capaz de fazer tudo o que você espera de um computador desktop, desde navegar na Internet e reproduzir vídeos de alta definição até criar planilhas, processamento de texto e jogos (RASPBERRY PI FOUNDATION, s.d., p. 1).

Este dispositivo é capaz de executar sistemas operacionais baseados em Linux como o Raspbian. De acordo com Raspbian (s.d., p. 1) o Raspbian é um sistema operacional gratuito baseado no Debian otimizado para o hardware Raspberry Pi. Um sistema operacional é o conjunto de programas e utilitários básicos que fazem seu Raspberry Pi funcionar.

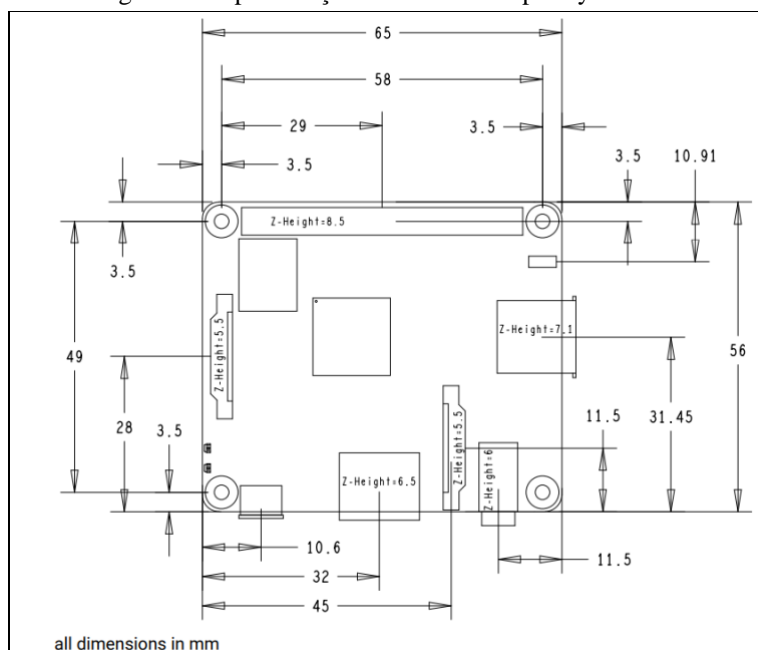
“O Raspberry Pi 3 modelo A+ é o mais recente produto da linha Raspberry Pi 3. Assim como o modelo B+, ele possui um processador quad core de 64 bits rodando a 1,4 GHz, LAN sem fio de banda dupla de 2,4 GHz e 5 GHz e Bluetooth 4.2” (RASPBERRY PI FOUNDATION, 2018, p. 2, Tradução nossa). A Figura 1 mostra o Raspberry Pi 3 A+ e a Figura 2 demonstra as especificações físicas do aparelho

Figura 1 – Raspberry Pi 3 A+



Fonte: Raspberry pi foundation (2018, p. 2).

Figura 2 - Especificações físicas do Raspberry Pi 3 A+



Fonte: Raspberry pi foundation (2018, p. 4).

2.2 PYPARROT

PyParrot é uma biblioteca desenvolvida em Python, sob licença MIT, projetada para programar drones Parrot Minidrone e Parrot Bebop. Como descrito por McGovern (s.d., p. 1), esta interface foi desenvolvida para ensinar crianças de todas as idades e qualquer pessoa interessada em programação autônoma de drones pode usá-la. A biblioteca se conecta com o drone por meio da rede *wireless* criada pelo drone e fornece métodos para controlar o drone e obter dados dos sensores.

O Quadro 1 demonstra um exemplo de código utilizando o drone Parrot Bebop 2 no qual contém a sequência de comandos para conectar-se ao drone, definir a função `sensors_update` para ser chamada quando houver atualização nos sensores, aguardar um segundo, mover-se para a direita por dois segundos. Em paralelo, o a função `sensors_update` exibe dados da bateria, estado de voo, latitude, longitude, altitude e guinada.

Quadro 1 – Comandos de voo utilizando a biblioteca PyParrot

```
bebop = Bebop(drone_type="Bebop2", ip_address="192.168.42.1")

class Server:
    def sensors_update(self, args):
        print("Battery: " + str(bebop.sensors.battery))
        print("Flying state: " + str(bebop.sensors.flying_state))
        if "GpsLocationChanged_latitude" in bebop.sensors.sensors_dict:
            lat = bebop.sensors.sensors_dict["GpsLocationChanged_latitude"]
            print("Lat: " + str(lat))
        if "GpsLocationChanged_longitude" in bebop.sensors.sensors_dict:
            lon = bebop.sensors.sensors_dict["GpsLocationChanged_longitude"]
            print("Lon: " + str(lon))
        if "GpsLocationChanged_altitude" in bebop.sensors.sensors_dict:
            print("Alt: " +
                  str(bebop.sensors.sensors_dict["GpsLocationChanged_altitude"]))
        if "AttitudeChanged_yaw" in bebop.sensors.sensors_dict:
            yaw = bebop.sensors.sensors_dict["AttitudeChanged_yaw"]
            print("Yaw: " + str(yaw))

    def start(self):
        success = bebop.connect(10)
        if (success):
            bebop.set_user_sensor_callback(self.sensors_update, None)
            bebop.ask_for_state_update()
            bebop.smart_sleep(1)
            bebop.disconnect()

server = Server()
server.start()
```

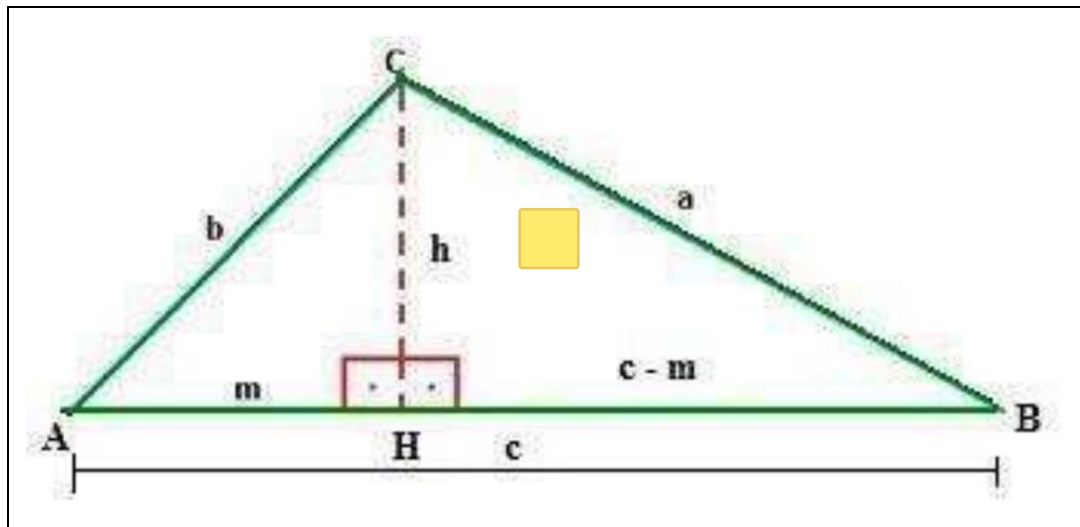
Fonte: Elaborado pelo autor.

2.3 LEI DOS COSSENOS



Em todo triângulo, o quadrado de qualquer um dos lados é igual à soma dos quadrados dos outros dois, diminuída do dobro do produto desses lados pelo cosseno do ângulo por eles formado. Seja o triângulo $\triangle ABC$, acutângulo conforme Figura 3, e $\overline{CH} = h$ a altura relativa ao lado \overline{AB} . Considerando o triângulo $\triangle BCH$, em que $a^2 = h^2 + (c - m)^2$ e o triângulo $\triangle ACH$, em que $h^2 = b^2 - m^2$, obtém-se que $a^2 = b^2 + c^2 - 2 \cdot c \cdot m$. Considerando que $m = b \cdot \cos \hat{A}$, chega-se a $a^2 = b^2 + c^2 - 2 \cdot b \cdot c \cdot \cos \hat{A}$. Com raciocínio análogo, pode-se escrever $b^2 = a^2 + c^2 - 2 \cdot a \cdot c \cdot \cos \hat{B}$ e $c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos \hat{C}$. (PEREIRA, Mariana Barreto; FERREIRA, Guttenberg Sergistótanés Santos, s.d., p. 7)

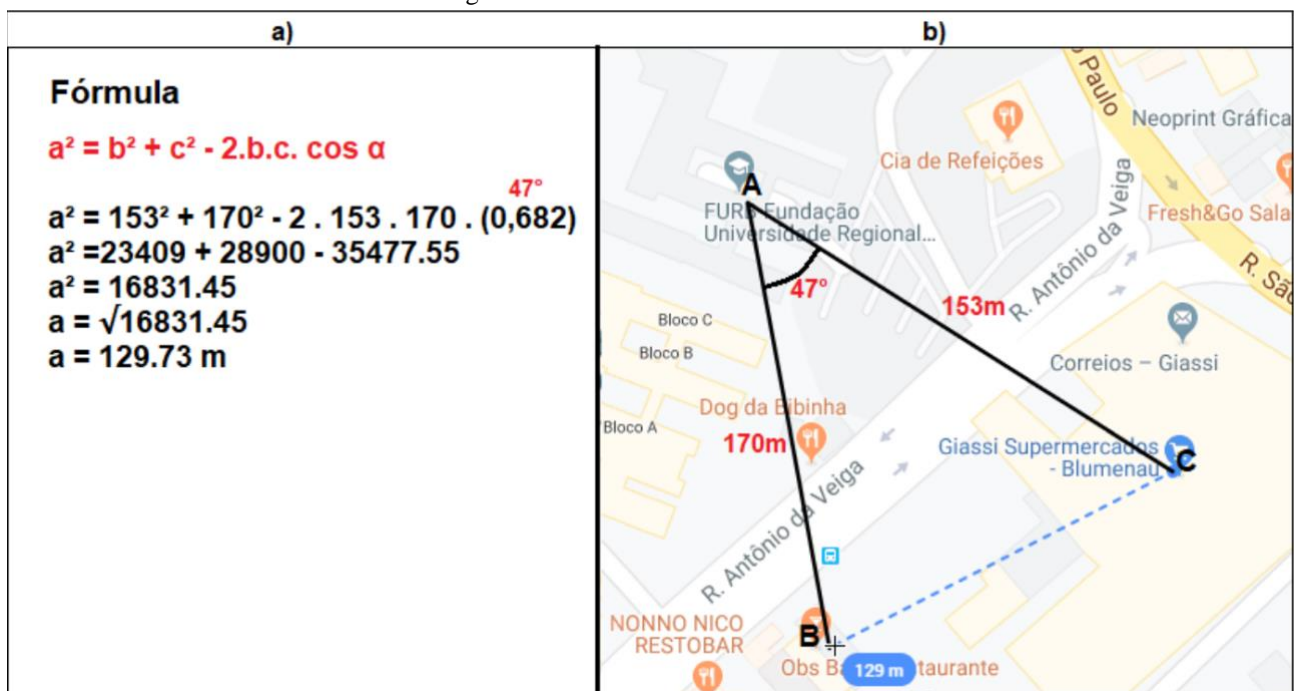
Figura 3 - Triângulo Retângulo



Fonte: Pereira, Mariana Barreto; Ferreira, Guttenberg [Sergistótanés Santos](#) (s.d., p. 6).

Para entender melhor como funciona a fórmula da lei dos cossenos serão utilizados exemplos do mundo real mostrados na Figura 4 (b). Sendo assim serão dados 3 pontos no mapa: A, B e C que formam um triângulo, porém são conhecidas apenas as distâncias de A para C (b), e de A para B (c), formando entre si um ângulo ($\cos \alpha$). Para calcular a distância desconhecida entre B e C será utilizado a lei dos cossenos. Para isso deve ser considerado que $b = 153\text{m}$, $c = 170\text{m}$ e $\cos \alpha = \cos 47^\circ = 0,682$ (esse valor convertido de graus para decimais é encontrado em tabelas trigonométricas). Substituindo esses valores na fórmula e calculando conforme Figura 4 (a), será obtido a distância entre B e C conforme Figura 4 (b). (SILVA, 2019, p. 4).

Figura 4 - Fórmula da lei dos cossenos

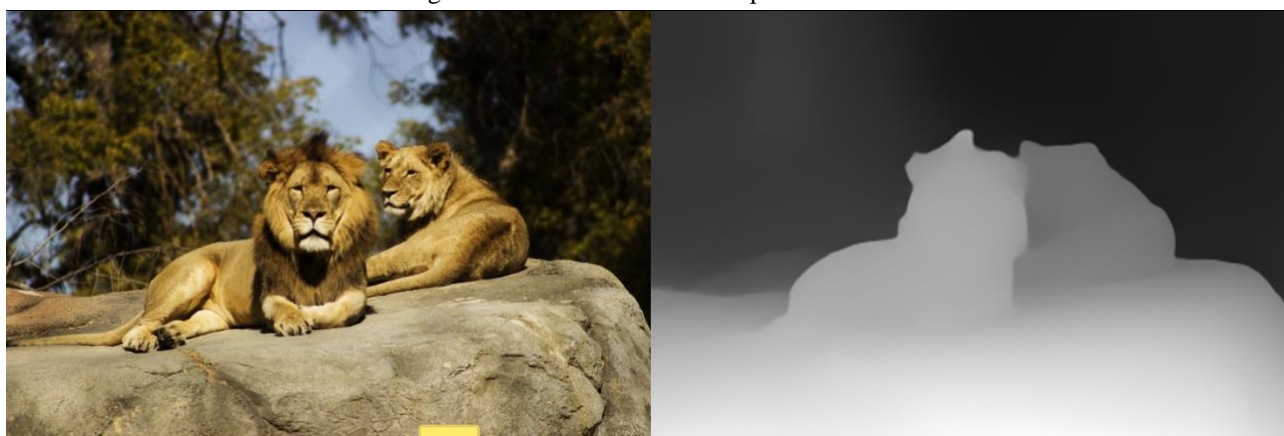


Fonte: Silva (2019, p. 4).

2.4 MIDAS

MiDaS é um modelo robusto de inferência de profundidade com imagem monocular (RANFTL *et al.*, 2020). Propomos um treinamento que é invariável a mudanças na amplitude e escala de profundidade, defendemos o uso de aprendizagem multi-objetivo baseada em princípios para combinar dados de diferentes fontes e destacar a importância de codificadores de pré-treinamento em tarefas auxiliares (RANFTL *et al.*, 2020, p. 1). A Figura 5 demonstra a esquerda a imagem colorida e a direita a profundidade estimada em escala de cinza.

Figura 5 - MiDaS estimativa de profundidade



Fonte: Hugging Face (s.d., p.1).

De acordo com Ranftl, René *et al.* (2020, p. 3), foi realizado um experimento com cinco conjuntos de dados diferentes e complementares para treinar o modelo. ReDWeb (WS), um conjunto de dados bastante preciso que apresenta cenas diversas e dinâmicas com validação, adquiridas de forma estérea. MegaDepth (MD), é maior, mas mostra predominantemente cenas estáticas, com maior precisão em regiões de fundo. WSVD (WS), consiste em vídeos estéreos obtidos da internet, com diversas características, sem validação, foi recriado uma validação de acordo com o procedimento descrito pelos autores originais. DIML Indoor (DL), é um conjunto de dados RGB-D, predominantemente com cenas estáticas em ambiente fechado, obtidas com um Kinect v2. Para comparar a performance do modelo, foi escolhido 6 conjuntos de dados baseados na diversidade e precisão do *ground-truth*.

2.5 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos correlatos. No Quadro 2 será descrito o trabalho de Corrêa (2020), que consiste em um sistema de navegação autônoma de drone baseando-se em GPS. Por fim, no Quadro 3 é apresentado o trabalho de Martins, Ramos e Mora-Camino (2018), que utiliza técnicas de processamento de imagem para reconhecimento e desvio de obstáculos em voo.

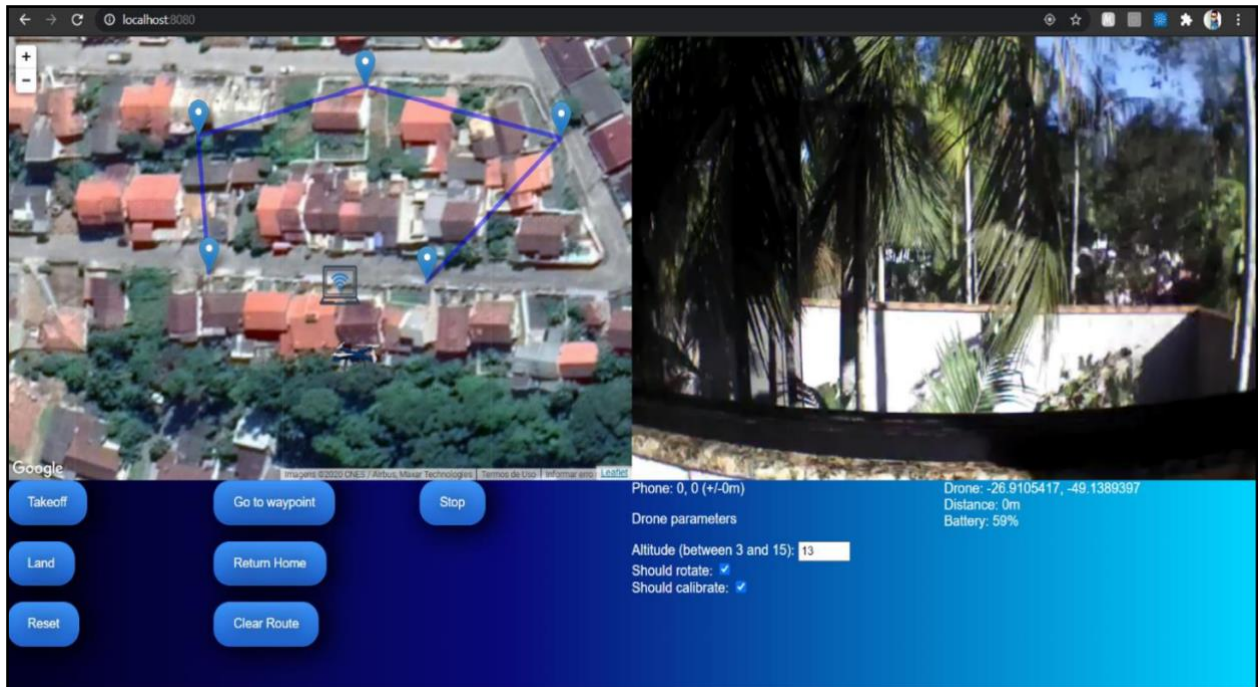
Quadro 2 – Drone autônomo: Vigilância aérea de espaços externos

Referência	<u>Corrêa (2020)</u>
Objetivos	<u>O trabalho tem como objetivo propor uma arquitetura para sistema de vigilância</u>
Principais funcionalidades	<u>O sistema permite cadastro de base e rotas através de uma aplicação web, oferece a possibilidade de percorrer rotas de forma autônoma baseado em GPS, registrando e disponibilizando dados</u>
Ferramentas de desenvolvimento	<u>A arquitetura utiliza o drone Parrot AR.Drone 2.0, foi implementado um servidor utilizando Node.js integrado a bibliotecas NPM para a comunicação com o drone</u>
Resultados e conclusões	<u>Foi disponibilizado a interface web com os comandos de voo, visualização por satélite e vídeo em tempo real. Foram realizados 3 cenários de testes para a navegação autônoma, nos quais tiveram um erro de precisão em até 5 metros. Conclui-se que a arquitetura proposta atendeu os objetivos</u>

Fonte: elaborado pelo autor.

A Figura 6 apresenta a interface web desenvolvida contendo os comandos de voo, dados de sensores, terreno em formato de visualização por satélite e as imagens da câmera em tempo real.

Figura 6 - Interface web da arquitetura



Corrêa (2020, p. 52).

Quadro 3 – A computer vision based algorithm for obstacle avoidance (outdoor flight)

Referência	Martins, Ramos e Mora-Camino (2018, p. 1)
Objetivos	O trabalho tem como objetivo implementar técnicas de processamento de imagem para reconhecer e desviar de obstáculos durante o voo de um Unmanned Aerial Vehicle (UAV), usando apenas a câmera frontal
Principais funcionalidades	Obter imagem de câmera em tempo real, detectar e desviar de obstáculos.
Ferramentas de desenvolvimento	Foi utilizado um quadrotor utilizando o controlador Pixhawk. O algoritmo de processamento de imagem foi implementado em C++ utilizando a plataforma Robot Operational System (ROS). As imagens foram simuladas com o simulador Gazebo
Resultados e conclusões	Nos testes aos quais foi submetido o drone desviou de todos os obstáculos que encontrou em seu caminho, porém o projeto limitou-se a desviar de obstáculos e não a seguir um plano de voo, e, dessa forma, cada desvio alterou o destino do drone. O algoritmo se mostrou eficaz para evitar colisões em voos externos, mas, por se basear em uma câmera, o algoritmo se torna sensível a mudanças de iluminação.

Fonte: elaborado pelo autor.

Para exemplificar o funcionamento do projeto a Figura 7 apresenta uma imagem a céu aberto com obstáculos, em seguida a Figura 8 demonstra a imagem já processada e dividida em partes iguais, identificado os blocos com obstáculos e as opções de áreas livres para o desvio.

Figura 7 - Foto capturada, um drone voando



Fonte: Martins, Ramos e Mora-Camino (2018, p. 1).

Figura 8 – Áreas livres



Fonte: Martins, Ramos e Mora-Camino (2018, p. 2).

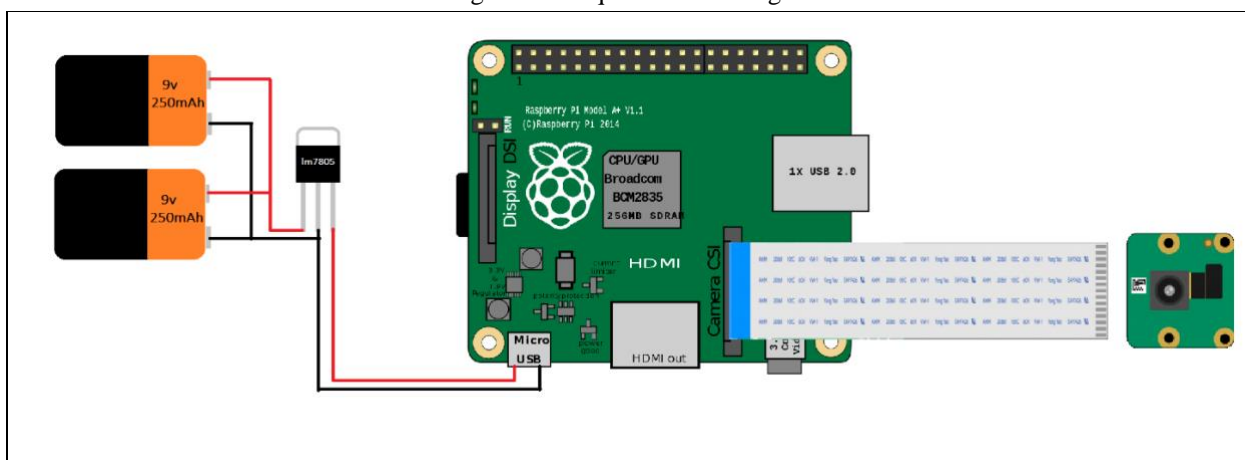
3 DESCRIÇÃO

Neste capítulo será apresentado detalhes de implementação de hardware e software, bem como a descrição do funcionamento da arquitetura. Na seção 3.1 é apresentada as especificações de hardware. Já na seção 3.2 aborda detalhes de implementação e funcionalidades do projeto.

3.1 ESPECIFICAÇÃO

O fornecimento de energia para o Raspberry Pi foi feito adicionando duas baterias de 9v 250mAh ligadas em paralelo com um controlador de tensão Lm7805 que converte a tensão de 9v para 5v. Também foi utilizado uma câmera de 5 Megapixels (MP) ligada ao Raspberry Pi por meio de um cabo flat. A Figura 9 representa o esquema de montagem e a Figura 10 demonstra os itens adicionados ao Parrot Bebop 2.

Figura 9 – Esquema de montagem



Fonte: Elaborado pelo autor.

Figura 10 - Disposição dos itens no drone



Fonte: Elaborado pelo autor.

3.2 IMPLEMENTAÇÃO

Esta seção apresenta o desenvolvimento da arquitetura para a navegação autônoma do drone Parrot Bebop 2 carregando o Raspberry Pi modelo 3 A+ a bordo. A Quadro 4 apresenta os Requisitos Funcionais (RF), já a Quadro 5 apresenta os Requisitos Não Funcionais (RNF).

Quadro 4 - Requisitos funcionais

Requisitos funcionais
RF01: o drone deverá seguir um plano de voo baseado em GPS
RF02: o drone deverá possuir uma câmera frontal
RF03: o drone deverá possuir um sistema de estabilização com base em sensores
RF04: o processamento em voo deverá ser totalmente a bordo, sendo dependente apenas da conexão com GPS
RF05: o drone deverá desviar de objetos em voo

Fonte: Elaborado pelo autor.

Quadro 5 - Requisitos não funcionais

Requisitos não funcionais
RF01: a detecção de obstáculos deve ser feita com o framework de aprendizado de máquina MiDaS
RF02: o drone deverá possuir um Raspberry Pi III modelo A+ para processamento de imagem a bordo

Fonte: Elaborado pelo autor.

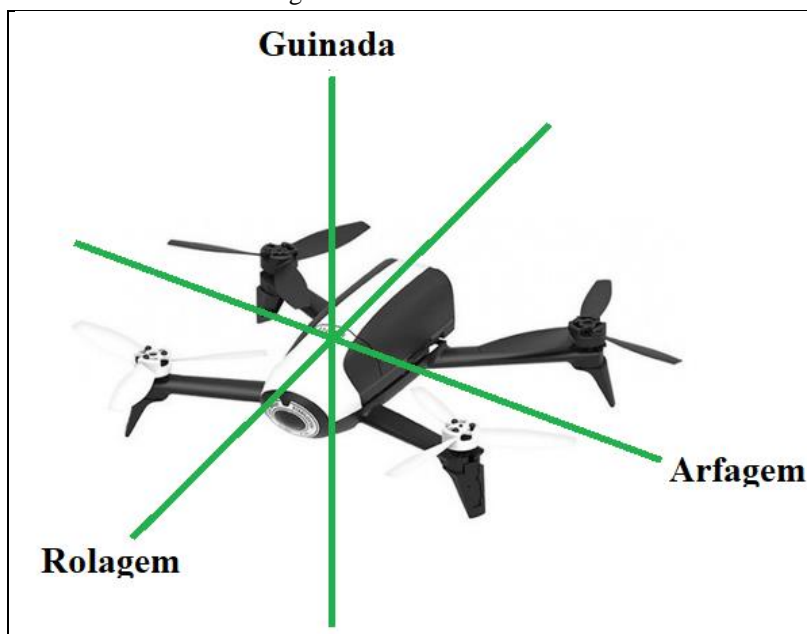
Para a implementação da rotina de navegação foi utilizada a linguagem Python na versão 3.7, juntamente com a biblioteca Pyparrot para os comandos de voo e acesso a dados de sensores. O desenvolvimento foi dividido em três partes, 1) Acesso aos comandos de voo; 2) cálculo do ângulo e rotina para direcionar o drone para a coordenada destino; 3) Rotina de estimativa de profundidade do modelo MiDaS.

Os comandos de voo são executados a partir da biblioteca Pyparrot através da classe `Bebop`, com a função `connect` é feita a conexão com o drone, definindo a quantidade de tentativas por parâmetro. A função `safe_takeoff`

envia comandos para o drone decolar e só termina de executar com a confirmação da decolagem através dos sensores, garantindo a decolagem **segura, já** a função `safe_land` envia comandos para pousar e termina a execução garantindo que o drone está no solo.

A movimentação do drone no ar é feita pela função `fly_direct`, sendo o parâmetro `pitch` o movimento de arfagem, com valor positivo para se movimentar para frente, com o `pitch` negativo o drone se movimenta para trás, o parâmetro `roll` representando o movimento de rolagem, com valor positivo o drone se movimenta para a direita, com o `roll` negativo o movimento é para a esquerda. Ainda existem os parâmetros `yaw` para o movimento de guinada, `vertical_movement` para o movimento vertical e o parâmetro `duration` onde é definido o tempo de duração de cada movimento. A Figura 11 ilustra a relação de rotação de eixos e parâmetros da função.

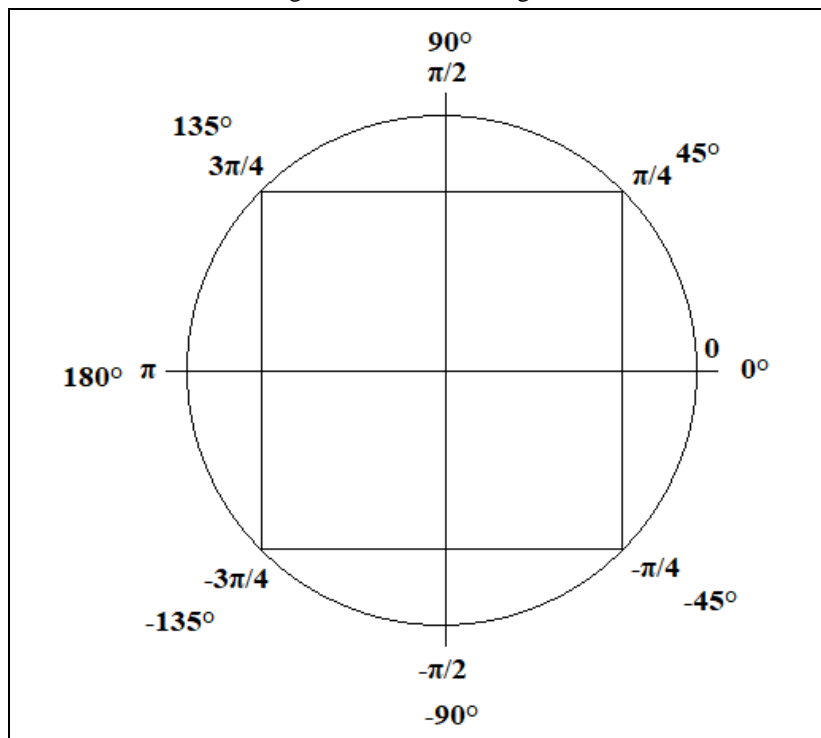
Figura 11 - Eixos do drone



Fonte: Elaborado pelo autor.

Para direcionar o drone para as coordenadas de destino foi utilizada a lei dos cossenos através da função `atan2` da biblioteca `math` padrão do Python, enviando como parâmetro a diferença das distâncias de latitude e longitude, resultando em um valor em radianos, conforme a Figura 12.

Figura 12 - Radianos e graus



Fonte: Elaborado pelo autor.

Como o dado da bússola do Pyparrot, encontrado no dicionário de sensores, `sensors.sensors_dict["AttitudeChanged_yaw"]` retorna o valor em radianos no mesmo padrão apresentado na Figura 12. A rotina de direcionar o drone até a coordenada consiste em arredondar os ângulos de guinada para uma casa após a vírgula, chamar a função `fly_direct` até que o valor de guinada do sensor seja o mesmo calculado pela função `atan2`. O Quadro 6 apresenta o trecho de código que direciona o drone para a coordenada de destino.

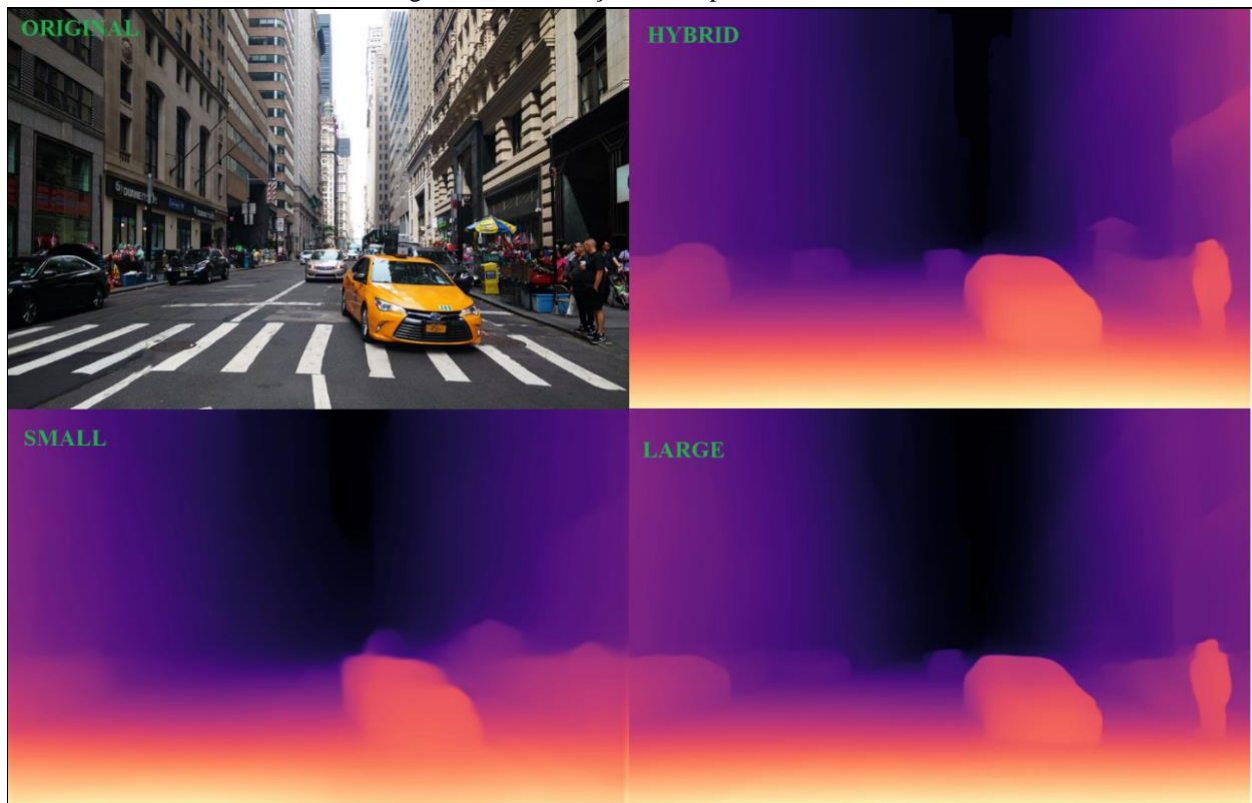
Quadro 6 - Direcinando o drone para o destino

```
def turn_to_coordinates(self):
    current_yaw = \
        self.get_yaw_angle_from_coordinates(
            self.lat, self.lon, self.lat_dest, self.lon_dest)
    self.yaw_angle = current_yaw
    rounded_yaw = round(current_yaw, 1)
    rounded_current_yaw = round(self.yaw, 1)
    while(rounded_yaw != rounded_current_yaw):
        bebop.fly_direct(roll=0, pitch=0, yaw=10,
            vertical_movement=0, duration=0.5)
```

Fonte: Elaborado pelo autor.

Para o reconhecimento de obstáculos foi utilizado o Pytorch MiDaS que faz a estimativa de profundidade dos objetos na imagem. O algoritmo da a possibilidade de escolher um tipo de modelo dentre três, `DPT_Large`, com alta acurácia e baixa velocidade de inferência, `DPT_Hybrid`, com acurácia média e velocidade de inferência também mediana e por fim, o modelo escolhido para esse projeto, `MiDaS_small`, com baixa acurácia e alta velocidade de inferência. A Figura 13 exibe imagens comparativas entre os modelos `DPT_Large` e `MiDaS_small`.

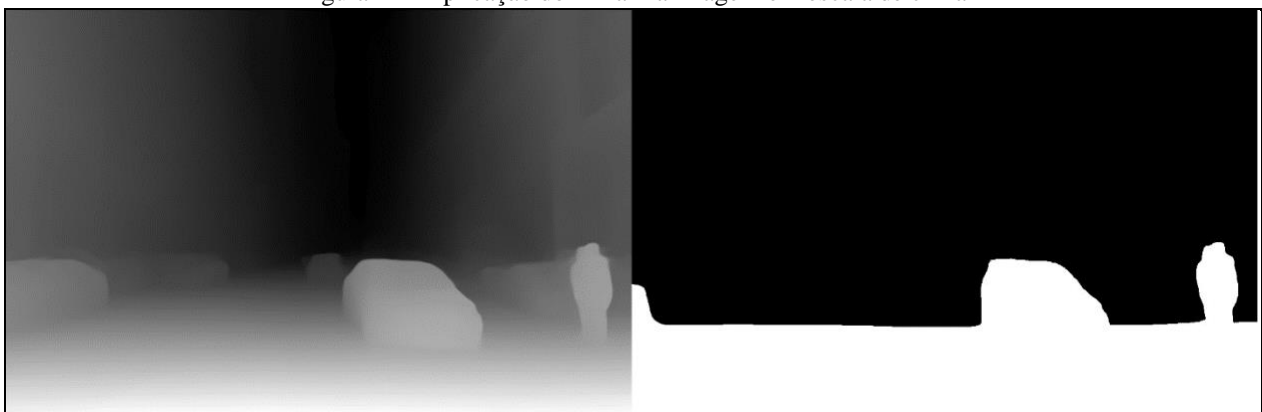
Figura 13 - Diferenças entre tipos de modelos



Fonte: Elaborado pelo autor.

Após o processamento da imagem, é retornado uma imagem em escala de cinza, sendo o mais claro mais próximo, então é aplicado um limite na imagem para deixar branco o que está perto e o restante preto, conforme exibido na Figura 14. Para identificar os obstáculos a imagem é dividida em três partes e é contado a quantidade de pixels brancos em cada parte. O algoritmo para identificação do obstáculo se dá a partir da quantidade de pixels brancos, o Quadro 7 exibe a regra para identificação e desvio do obstáculo.

Figura 14 - Aplicação do limiar na imagem em escala de cinza



Fonte: Elaborado pelo autor.

Quadro 7 - Pseudocódigo de identificação e desvio de obstáculo

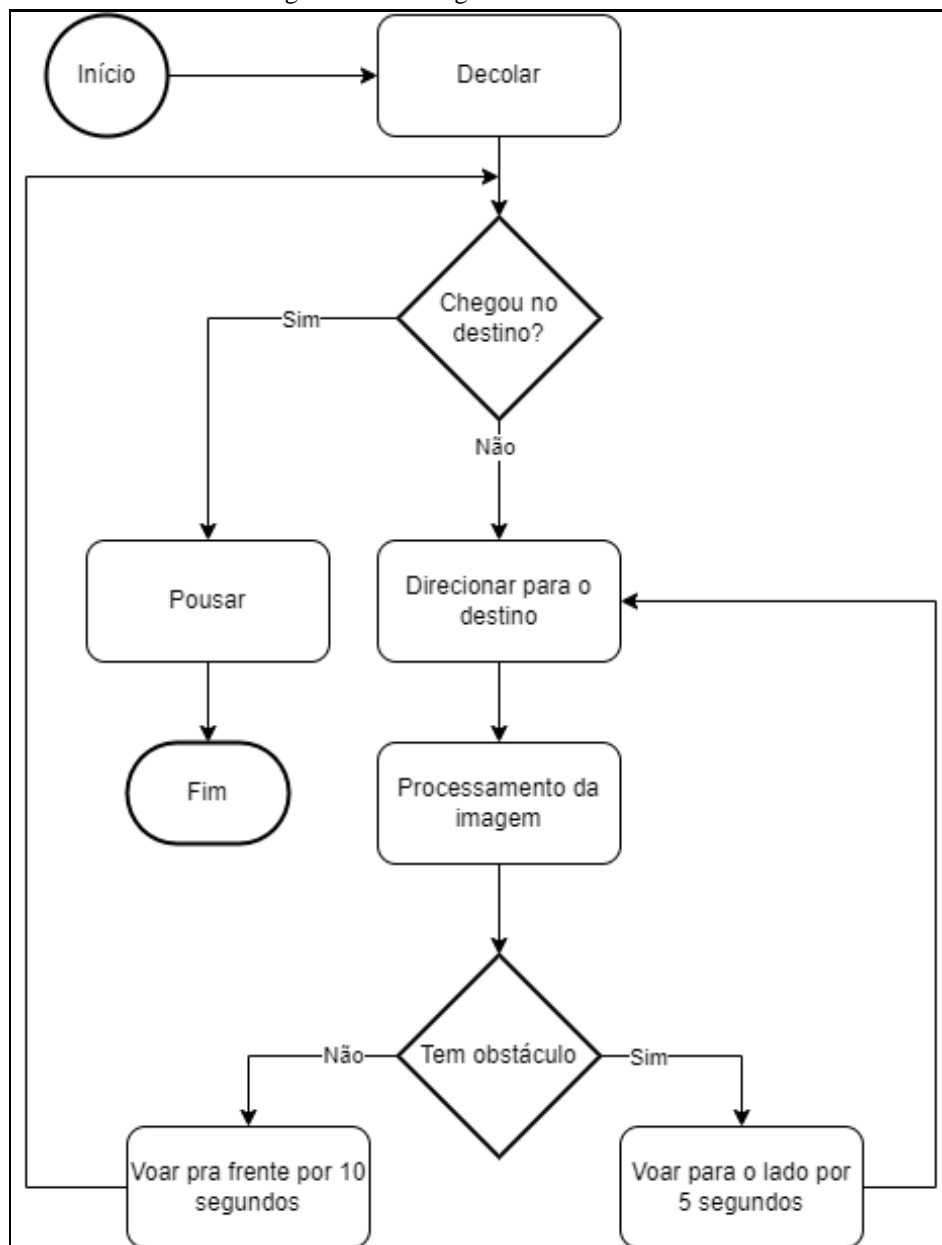
```
if (non_zero_center > 1000):
    if (left_count < 1000 and left_count <= right_count):
        print("mover left")
    else:
        print("move right")
```

Fonte: Elaborado pelo autor.

O fluxo da execução do algoritmo é representado na Figura 15 e se inicia pela decolagem, em seguida o valor do GPS arredondado é comparado ao valor arredondado das coordenadas de destino, caso sejam iguais, significa que o drone chegou no destino, então é feito o pouso e o fluxo finaliza. Contudo se os valores divergirem, entende-se que o drone não atingiu o destino, com isso é iniciada a rotina de direcionar o drone para o destino, em seguida o Raspberry Pi captura

uma imagem e aplica o modelo MiDaS para estimativa de profundidade, com a imagem processada verifica-se se há obstáculos, caso não tenha, é enviada a ação de voar para frente por 10 segundos e o fluxo volta para a verificação das coordenadas, contudo, se existe um obstáculo é enviada a ação de voar para o lado oposto do obstáculo por 5 segundos e o fluxo retorna para o direcionamento do drone.

Figura 15 – Fluxograma da rotina de voo



Fonte: Elaborado pelo autor.

4 RESULTADOS

5 CONCLUSÕES

REFERÊNCIAS

CORRÊA, Diego F. **Drone autônomo: Vigilância aérea de espaços externos**. 2020. 67 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2020. Disponível em: http://dsc.inf.furb.br/arquivos/tccs/monografias/2020_1_diego-fachinello-correa_monografia.pdf. Acesso em: 20 jun. 2022.

HUGGING FACE. **MiDaS**. [S. l.], [s.d.]. Disponível em <https://huggingface.co/spaces/pytorch/MiDaS>. Acesso em 20 de junho de 2022.

LUGO, Jacobo J.; ZELL, Andreas. Framework for Autonomous On-board Navigation with the AR.Drone. **J Intell Robot Syst**, [s. l.], v. 73, p. 401–412, 19 out. 2013. DOI <https://doi.org/10.1007/s10846-013-9969-5>. Disponível em: <http://www.ra.cs.uni-tuebingen.de/publikationen/2013/jimenezJINT2013.pdf>. Acesso em: 20 jun. 2022.

MARTINS, Wander M. *et al.* A Computer Vision Based Algorithm for Obstacle Avoidance. **Springer International Publishing**, [s. l.], ano 2018, p. 569-575, 13 abr. 2018. DOI https://doi.org/10.1007/978-3-319-77028-4_73. Disponível em: https://www.researchgate.net/profile/Alexandre-C-B-Ramos/publication/324731184_A_Computer_Vision_Based_Algorithm_for_Obstacle_Avoidance/links/5adf8183458515c60f63c28f/A-Computer-Vision-Based-Algorithm-for-Obstacle-Avoidance.pdf. Acesso em: 20 jun. 2022.

MCGOVERN, Amy. **Bebop Commands and Sensors**. [S. l.], [s.d.]. Disponível em: <https://pyarrow.readthedocs.io/en/latest/bebopcommands.html>. Acesso em 20 jun 2022.

MUR-ARTAL, Raúl; TARDÓS, Juan D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. **IEEE Transactions on Robotics**, [s. l.], ano 2017, v. 33, ed. 5, p. 1255-1262, 12 jun. 2017. DOI 10.1109/TRO.2017.2705103. Disponível em: <https://arxiv.org/pdf/1610.06475.pdf>. Acesso em: 20 jun. 2022.

PEREIRA, Mariana Barreto; FERREIRA, Guttenberg Sergistótes Santos. **SOBRE RELAÇÕES MÉTRICAS, LEIS DE SENOS E COSSENOS NO ENEM: ESTUDO DE INICIAÇÃO CIENTÍFICA**. [s. l.], [s.d.]. Disponível em: https://www.academia.edu/download/38963782/SOBRE_RELACOES_METRICAS_LEIS_DE_SENOS_E_COSENOS_NO_ENEM_UM_ESTUDO_DE_INICIACAO_CIENTIFICA.pdf. Acesso em: 24 jun. 2022.

RANFTL, René *et al.* Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [s. l.], ano 2022, v. 44, n. 3, p. 1623-1637, 27 ago. 2020. DOI 10.1109/TPAMI.2020.3019967. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9178977>. Acesso em: 20 jun. 2022.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model A+**. [S. l.], nov. 2018. Disponível em: https://www.raspberrypi.org/app/uploads/2018/11/Raspberry_Pi_3A_product_brief.pdf. Acesso em: 20 jun 2022.

RASPBERRY PI FOUNDATION. **What is a Raspberry Pi?**. [S. l.], [s.d.]. Disponível em: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. Acesso em: 20 jun 2022.

RASPIBIAN. **Welcome to Raspbian**. [S. l.], [s.d.]. Disponível em: <https://www.raspbian.org/>. Acesso em: 20 jun 2022.

SILVA, William Lopes da. **Black glasses**: Assistente para deficientes visuais via geolocalização. Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau. Blumenau, dez. 2019. Disponível em: http://dsc.inf.furb.br/arquivos/tccs/monografias/2019_2_william-lopes-da-silva_monografia.pdf. Acesso em: 20 jun 2022.