UNIFEOB
ESCOLA DE NEGÓCIOS

2024

PROJETO INTEGRADO

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

AUTOMAÇÃO ROBÓTICA: SOLUÇÕES SUSTENTÁVEIS E INCLUSIVAS

GRUPO MALPA

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2024

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

AUTOMAÇÃO ROBÓTICA: SOLUÇÕES SUSTENTÁVEIS E INCLUSIVAS

GRUPO MALPA

MÓDULO DE ROBÓTICA

Cálculo Diferencial e Integral – Prof. Carlos Alberto Collozzo de Souza

Robótica – Prof. Marcelo Ciacco de Almeida

Machine Learning – Prof. Rodrigo Marudi de Oliveira

Álgebra Linear e Geometria Analítica – Prof. Carlos Alberto Collozzo de Souza

Projeto de Robótica – Profª. Mariângela M. Santos

Estudantes:

Bruno Cardoso Silva, RA 22000657
Igor Gabriel Moraes Gaspar, RA 22001517
Igor Guilherme dos Reis Melo, RA 24001497
Luciano Aimon Ottoni Santos, RA 24001813
Matheus Souza Pinto, RA 22000502
Vinycius Gabriel Oblonczyk, RA 22001650

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2024

SUMÁRIO

1 INTRODUÇÃO

O projeto “Automação Robótica: Soluções Sustentáveis e Inclusivas” tem como objetivo desenvolver um robô autônomo que contribua para os Objetivos de Desenvolvimento Sustentável (ODS) da ONU. Utilizando um Raspberry Pi 4 como plataforma base, o robô será equipado com um sistema de hover para mobilidade eficiente e um braço robótico para manipulação de objetos. Integrando técnicas de machine learning e visão computacional, ele será capaz de reconhecer diferentes tipos de terreno e operar de forma autônoma em ambientes variados.

A principal função do robô será identificar e recolher pequenos resíduos, promovendo a limpeza de espaços urbanos e naturais, contribuindo para a sustentabilidade ambiental (ODS 11). O projeto também contou com o apoio da empresa **3D-Aimon**, que patrocinou a fabricação de peças essenciais para o robô por meio de impressão 3D, garantindo componentes leves, resistentes e personalizados para a solução. Essa parceria permitiu a aplicação de conhecimentos em Robótica, Machine Learning, Álgebra Linear, Geometria Analítica e Cálculo Diferencial, alinhando inovação tecnológica com soluções práticas para desafios globais, como o cuidado com o meio ambiente e a promoção de cidades mais sustentáveis.

2 DESCRIÇÃO DA EMPRESA

A Małpa é uma empresa fictícia criada para a elaboração do projeto integrado. Ela atua principalmente na área de tecnologia, e seu surgimento foi motivado pela constatação dos desafios recorrentes enfrentados pela comunidade em relação aos ODS (Objetivos de Desenvolvimento Sustentável da ONU). Esses desafios, muitas vezes, envolvem a poluição ambiental, e a empresa emergiu como uma solução para esse problema.

Seus serviços estão focados na área de tecnologia, com projetos voltados para sustentabilidade e inovação. Em resumo, a principal contribuição da Małpa é a criação de um robô que auxilia seus usuários no descarte correto e consciente de latas de refrigerante.

Com uma abordagem voltada para a conscientização e praticidade, a Małpa visa não apenas reduzir os impactos negativos do descarte incorreto de resíduos, mas também educar os usuários por meio de interações amigáveis, incentivando uma contribuição mais significativa para o descarte adequado.

3 PROJETO INTEGRADO

3.1 CÁLCULO DIFERENCIAL E INTEGRAL

O cálculo diferencial e integral desempenha um papel fundamental em robótica, permitindo descrever e controlar o movimento e o equilíbrio do robô. Essas ferramentas matemáticas são utilizadas para definir e ajustar a trajetória e a dinâmica, otimizando seu desempenho em diversas aplicações. Como Sira-Ramírez e Agrawal (2004) afirmam, "o uso de métodos de cálculo diferencial permite entender as leis de movimento de sistemas robóticos e definir respostas adequadas para controle".

3.1.1 CÁLCULO DE VELOCIDADE E ACELERAÇÃO

Nesta etapa, o cálculo diferencial é aplicado para determinar a taxa de variação da posição em relação ao tempo, que representa a velocidade, e a taxa de variação da velocidade, que é a aceleração. Como afirmado por Kelly (2013), "a derivada da função de posição em relação ao tempo permite calcular a velocidade, e a segunda derivada fornece a aceleração, ambos essenciais para o controle em tempo real de robôs" .

Para o carrinho hover, a derivada da função posição permite calcular a velocidade de cada motor, garantindo que o robô se mova suavemente. Por exemplo, se a posição $s(t)$ é dada em função do tempo, a velocidade $v(t) = \frac{ds}{dt}$ e a aceleração $a(t) = \frac{d^2s}{dt^2}$ podem ser usadas para ajustar a resposta dos motores a cada instante .

3.1.2 PLANEJANDO CAMINHOS

As integrais são fundamentais para calcular a trajetória ótima que o robô deve seguir. Segundo LaValle (2006), "a integração da função de velocidade em relação ao tempo permite calcular distâncias percorridas e planejar trajetórias eficientes em sistemas autônomos" .

A aplicação no projeto de integrais permite definir a trajetória ideal do carrinho hover, minimizando o consumo de energia. No planejamento de caminhos, as integrais

ajudam a somar pequenas variações de posição e a ajustar a direção para que o robô se desloque de forma eficiente até o objeto desejado.

3.1.3 ESTABILIDADE

A estabilidade de um sistema robótico pode ser garantida por meio da análise de equações diferenciais, que permitem prever o comportamento do sistema sob várias condições. Craig (2005) destaca que "o uso de sistemas diferenciais em robótica auxilia na avaliação de estabilidade, garantindo que o robô mantenha seu equilíbrio em movimento e operação" .

A estabilidade aplicada ao carrinho hover ajuda a controlar o centro de massa e ajustar a resposta dos motores ao manipular objetos, garantindo que o robô e a garra não vire. Métodos de cálculo de estabilidade, como o cálculo de pontos de equilíbrio, são importantes para manter o robô seguro e eficiente em condições variadas.

3.2 ROBÓTICA

3.2.1 CONCEITO DO SISTEMA

A robótica, enquanto área interdisciplinar, envolve a criação de sistemas automatizados e inteligentes que podem interagir com o ambiente de maneira autônoma. Como descrevem Romero, Prestes e Osório (2014), a robótica móvel se destaca por estudar e desenvolver robôs capazes de se deslocarem no espaço, ampliando suas capacidades de percepção e interação em ambientes dinâmicos e complexos. Esse campo de estudo abrange não apenas o desenvolvimento de hardware e software, mas também a integração de sensores, atuadores e algoritmos avançados para navegação, reconhecimento de objetos e tomada de decisão autônoma. E no projeto, estamos aplicando essas inovações para criar um sistema robótico que inclui um rover e um braço robótico.

O rover será capaz de explorar diferentes tipos de terreno e é equipado com uma câmera para identificar e um braço robótico para coletar objetos. Ele se move com a ajuda de esteiras, que facilitam o deslocamento em terrenos irregulares. A inteligência artificial (IA) será usada para coordenar o rover e o braço robótico, garantindo que os

objetos sejam coletados e levados para os locais certos.
O braço robótico permitirá manipular objetos com precisão. Combinando essas tecnologias, nosso sistema será capaz de realizar tarefas de forma autônoma e eficiente, enfrentando os desafios técnicos do projeto.

A arquitetura do robô é composta pelos seguintes principais componentes:

1. Placa Principal:

Raspberry Pi 4 Model B: Atua como a unidade de controle central, coordenando as ações dos motores e servos, além de processar os dados recebidos dos sensores e tomar decisões.

2. Sistema de Controle dos Motores:

Driver de Servos PCA9685 (16 canais, 12 bits): Responsável pelo controle dos servos do braço robótico e dos motores de locomoção do rover, garantindo movimentos precisos e coordenados. Alimentação: 5V compartilhada com os servos conectados.

3. Rover:

Motores DC (3-6V): Quatro motores DC são utilizados para movimentar o rover através das esteiras, proporcionando tração e mobilidade em diversos tipos de terreno. Alimentação: 6V (proveniente de uma fonte de 4 pilhas AA, conectada à Ponte H).

Ponte H L298N: Controla a direção e a velocidade dos motores DC, permitindo que o rover se mova para frente, para trás e execute manobras em diferentes direções. Alimentação: 6V, fornecida pela bateria de 4 pilhas AA.

4. Braço Robótico:

Servos MG996R (Metal, TowerPro): Três servos MG996R são utilizados para movimentar a base do braço com rotação de 180º e para as juntas do braço, oferecendo flexibilidade e precisão na manipulação de objetos. Alimentação: 5V, com consumo de corrente em torno de 2A para movimentos com torque elevado, fornecida pelo driver PCA9685.

Micro Servos MG90 (TowerPro): Três micro servos controlam o punho do braço robótico, permitindo a rotação nos eixos X, Y e Z, além de equipar a garra para a coleta de objetos. Alimentação: 5V (mesma alimentação do PCA9685, mas com menor consumo de corrente devido ao porte menor).

5. Câmera:

Logitech G920: Equipado com uma câmera para a verificação do terreno e identificação de objetos. Essa câmera fornece imagens em tempo real, permitindo ao sistema realizar análises e tomar decisões sobre a navegação e a coleta de itens. Alimentação: via USB diretamente do Raspberry Pi, com consumo aproximado de 500mA.

O funcionamento autônomo do robô envolve a capacidade do rover de navegar por diversos tipos de terreno, como terra e grama, e lidar com diferentes níveis de

relevo. A inteligência artificial será responsável por processar os dados capturados pelos sensores e tomar decisões para a navegação e coleta de objetos. O sistema autônomo permitirá ao rover explorar o ambiente e ao braço robótico realizar tarefas de manipulação com precisão, sem a necessidade de supervisão contínua.

O sistema robótico será capaz de realizar a coleta e transporte de objetos de forma autônoma, utilizando um conjunto de sensores e atuadores para interação com o ambiente. A combinação da arquitetura robusta e da inteligência artificial permitirá ao robô operar de maneira eficiente em uma variedade de cenários e condições, atendendo aos requisitos do projeto.

3.2.2 DESENVOLVIMENTO DAS SOLUÇÕES

O hover robótico será utilizado para fornecer mobilidade autônoma e eficiente ao sistema, permitindo que o robô se desloque facilmente por diversos tipos de terrenos. Com sua capacidade de flutuar e se mover sem contato direto com o solo, o hover possibilitará o monitoramento e a inspeção de áreas urbanas e naturais, o acesso a locais de difícil acesso, a entrega de pequenos suprimentos e a execução de intervenções em situações de emergência. Essa plataforma de mobilidade é fundamental para a operação sustentável do robô, garantindo eficiência energética e capacidade de operação prolongada em ambientes variados.

Um braço manipulador é um mecanismo sofisticado para mover materiais, ferramentas, entre outros. Em geral, ele é programado usando teoria de controle que permite calcular a velocidade e a aceleração para cada junta do robô Romero et al. (2014). O braço robótico, equipado com uma pinça de precisão, será integrado ao hover para executar tarefas complexas que exijam manipulação delicada. Essa pinça pode ser adaptada para diferentes finalidades, como a coleta de pequenos resíduos (papel, plástico, etc.), a coleta de amostras ambientais, ou até a manipulação de objetos específicos, dependendo das necessidades da missão. Além disso, o braço robótico poderá ser equipado com outras pinças, como modelos para manipulação de objetos maiores ou ferramentas especializadas, aumentando sua versatilidade para diferentes aplicações.

No projeto “Automação Robótica: Soluções Sustentáveis e Inclusivas”, o hover robótico proporciona mobilidade eficiente, enquanto o braço robótico realiza a manipulação precisa de objetos, focando na coleta de resíduos e amostras ambientais para promover a sustentabilidade (ODS 11). A estrutura do braço permite a adaptação para diferentes tipos de pinças, ampliando seu campo de aplicação e tornando-o adequado para diversas tarefas, como assistência a pessoas com deficiência ou intervenções em locais de difícil acesso. Essa combinação é gerenciada por um Raspberry Pi 4, que utiliza técnicas de machine learning e visão computacional para reconhecimento do terreno e otimização da operação autônoma do robô.

3.2.3 INTEGRAÇÃO E CONTROLE

A integração do hover e do braço robótico será realizada através de um sistema de controle centralizado, baseado em um Raspberry Pi 4b. Este controlador central gerencia todas as operações do robô, desde o processamento de dados de sensores para reconhecimento de terreno, até o controle de movimento do hover e a manipulação de objetos pelo braço robótico. O sistema de controle utilizará técnicas de machine learning e visão computacional para garantir a tomada de decisões autônomas e a adaptação do robô a diferentes ambientes e condições operacionais.

A coordenação das operações do hover e do braço robótico será feita de forma integrada, permitindo que ambos os componentes funcionem em harmonia para a execução de tarefas complexas. O sistema centralizado monitora continuamente o terreno, ajustando a velocidade e a direção do hover enquanto o braço robótico se posiciona e manipula objetos. Isso inclui a adaptação do tipo de pinça utilizada no braço, conforme a tarefa a ser realizada, seja para recolher pequenos resíduos, amostras ambientais ou realizar tarefas de assistência em locais de difícil acesso.

A utilização combinada do hover e do braço robótico no projeto permitirá uma abordagem versátil e eficiente para a realização de atividades de coleta de resíduos, monitoramento ambiental e apoio a pessoas em áreas de difícil acesso. A mobilidade fornecida pelo hover permitirá que o robô se desloque facilmente por diversos

terrenos, enquanto o braço robótico executa a manipulação precisa de objetos. O controle centralizado assegura que essa combinação funcione de maneira coordenada e eficaz, promovendo a sustentabilidade (ODS 11) e contribuindo para a criação de soluções robóticas inovadoras e inclusivas.

Conforme abordado na literatura, o desenvolvimento de sistemas robóticos autônomos e adaptáveis tem se mostrado fundamental para a implementação de tecnologias que atendam a demandas sociais e ambientais contemporâneas (JUNIOR et al., 2019). O uso de plataformas como o Raspberry Pi, além de acessível, proporciona a flexibilidade necessária para integrar diferentes tecnologias e aprimorar a eficiência do robô em cenários variados.

3.3 MACHINE LEARNING

Este projeto tem como objetivo não apenas a criação de um sistema automatizado, mas também a promoção de soluções tecnológicas que impulsionam um futuro mais sustentável. Com o uso de inteligência artificial (IA), análise de dados e ferramentas avançadas como Python e OpenCV, nossa equipe desenvolveu um robô capaz de identificar, coletar e descartar corretamente objetos, como latas de refrigerante, em lixeiras adequadas. Essa automação inteligente busca contribuir para a redução de resíduos indevidamente descartados, promovendo práticas mais responsáveis e ambientalmente conscientes.

O projeto se destaca pelo aprendizado prático de técnicas avançadas, como o treinamento de modelos com plataformas como o Roboflow, que oferece uma base sólida para estudantes e profissionais aplicarem algoritmos de visão computacional em cenários reais. A integração de bibliotecas como OpenCV permite que o robô “enxergue” e entenda o ambiente ao seu redor, possibilitando ações precisas e eficientes no processo de coleta.

Além disso, o desenvolvimento desse robô reforça a importância da inovação aplicada ao contexto da sustentabilidade, demonstrando como a IA pode contribuir diretamente para resolver problemas ambientais e simplificar processos do cotidiano. Ao promover o avanço no desenvolvimento de soluções robóticas inteligentes, o projeto incentiva a inclusão de tecnologias modernas na formação acadêmica, preparando os participantes para desafios futuros e despertando neles uma consciência sustentável que pode impactar a sociedade como um todo.

3.3.1 IMPLEMENTANDO ALGORITMOS DE MACHINE LEARNING EM SOLUÇÕES SUSTENTÁVEIS

A integração da inteligência artificial e do machine learning em tecnologias sustentáveis emergiu como uma esperança para enfrentar os desafios ambientais globais. Essas tecnologias estão transformando setores ao otimizar o uso de energia, melhorar a eficiência de processos e reduzir emissões de carbono (FULL SCALE, 2024).

No desenvolvimento da Inteligência Artificial para o robô coletor de latinhas, iniciamos o treinamento do modelo utilizando a plataforma Roboflow. Apesar de não termos conseguido baixar o modelo devido aos termos de uso, a plataforma nos proporcionou um ambiente intuitivo para treinar a IA com nossos próprios dados de imagens. Visando otimizar o processo, migramos para o uso da biblioteca OpenCV (cv2) em conjunto com o modelo pré-treinado YOLOv8, uma rede neural especializada em detecção de objetos em tempo real. Também utilizamos a biblioteca Numpy para processar os dados e realizar cálculos matemáticos. Essa abordagem facilitou a implementação, uma vez que o modelo YOLOv8 já possui uma base de reconhecimento visual robusta, exigindo menos ajustes para o nosso propósito.

Utilizando Python como linguagem principal, integramos essas bibliotecas para garantir que o robô pudesse executar tarefas de visão computacional de forma rápida e precisa. Esse método exemplifica a aplicação prática de técnicas de aprendizado supervisionado, aproveitando uma rede previamente treinada para adaptar o modelo ao nosso projeto. Técnicas de regressão e classificação ajudam o robô a tomar decisões eficientes durante o processo de coleta, enquanto o clustering permite agrupar os

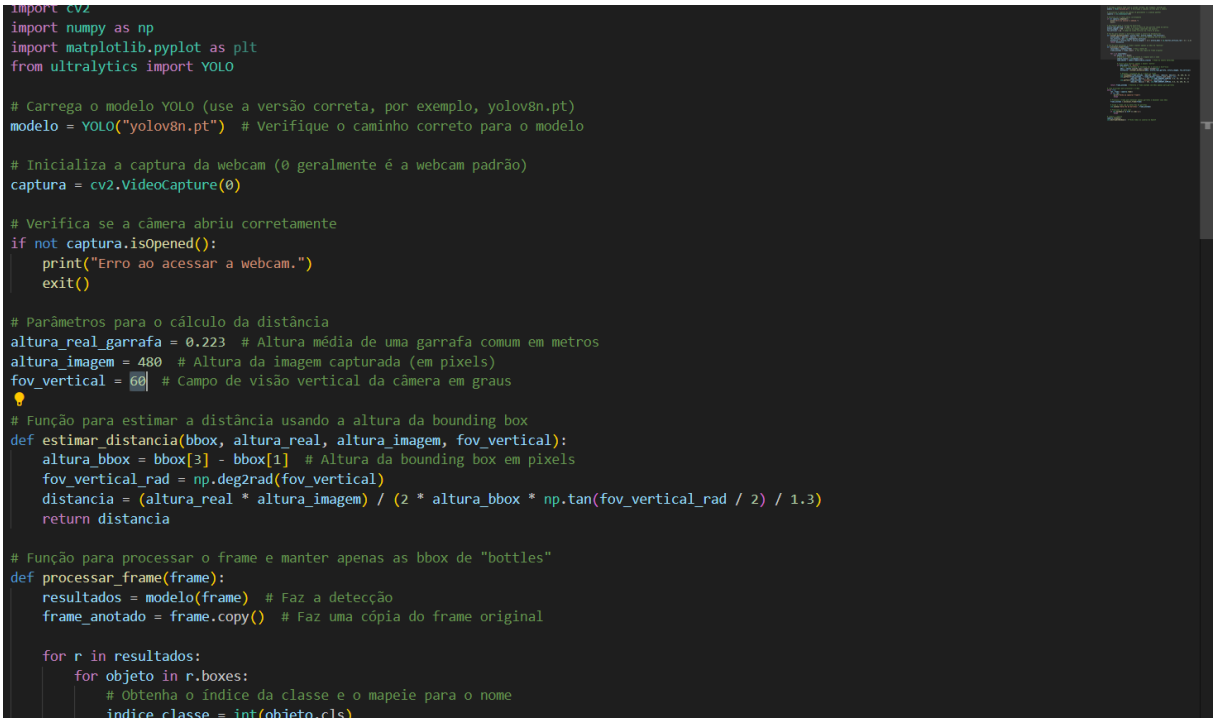
objetos em diferentes classes, otimizando a organização e descarte de resíduos. Essa solução automatizada contribui para um impacto ambiental positivo.

Além disso, devemos destacar como o uso dessas técnicas e algoritmos contribui diretamente para a eficiência e sustentabilidade do projeto. A automação proposta tem como foco principal a otimização da coleta seletiva de resíduos, promovendo a redução de erros e o aumento da velocidade e precisão do processo, o que se traduz em um impacto ambiental positivo. Ao mesmo tempo, deve-se considerar o impacto social da solução, uma vez que ela pode auxiliar na conscientização sobre a importância da reciclagem e melhorar as práticas de descarte de lixo, tanto em ambientes públicos quanto privados.

Com a implementação bem-sucedida do YOLOv8 e a execução dos treinamentos e ajustes necessários, a Inteligência Artificial do robô demonstrou resultados promissores na identificação e coleta de garrafas plásticas. A IA não só conseguiu reconhecer com precisão os objetos, mas também calculou com eficiência as distâncias entre o robô e as garrafas, permitindo a aproximação e a captura de forma coordenada e autônoma. Esse processo assegura que o robô seja capaz de realizar as tarefas propostas com precisão e confiabilidade, atendendo plenamente aos requisitos definidos pelos professores para o projeto.

O Raspberry Pi, em conjunto com a versatilidade dos microcontroladores Arduino e da inteligência artificial, formaram uma plataforma para a criação de um sistema inteligente. O Raspberry, atua como o cérebro da operação, processa as imagens coletadas pela câmera, permitindo que a IA aprenda e tome decisões autônomas.

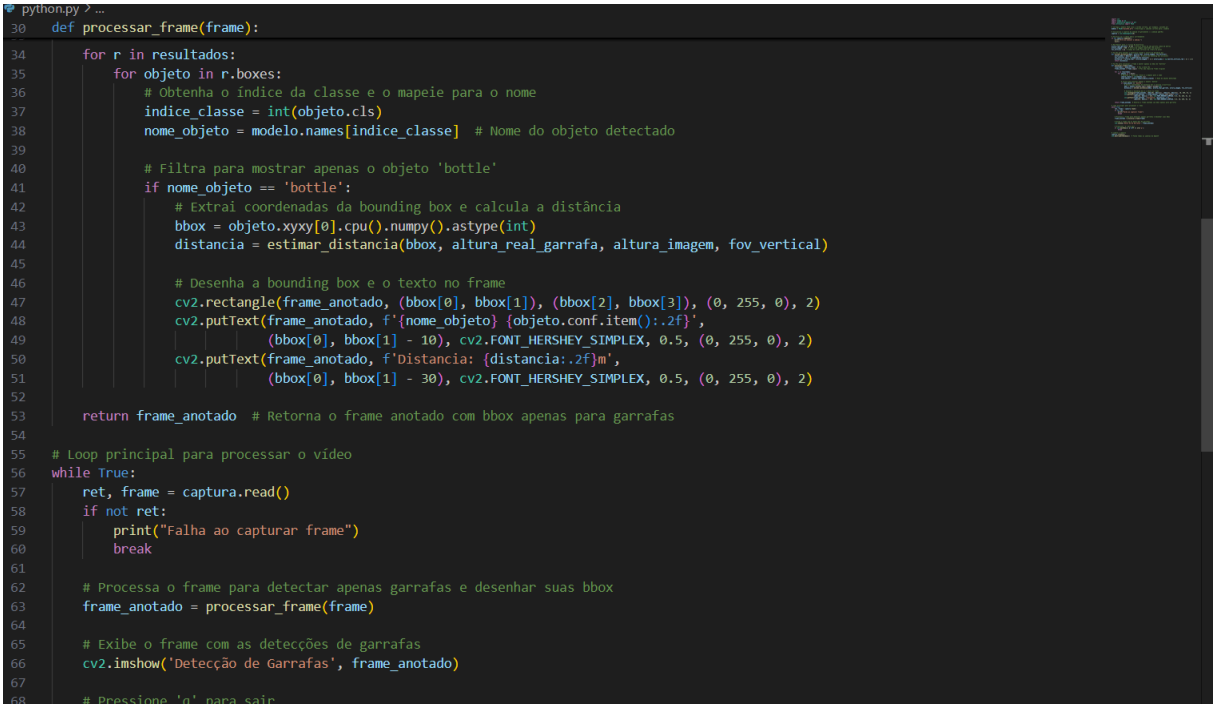
Figura 1: Primeira parte do código



```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from ultralytics import YOLO

# Carrega o modelo YOLO (use a versão correta, por exemplo, yolov8n.pt)
modelo = YOLO("yolov8n.pt")  # Verifique o caminho correto para o modelo

# Inicializa a captura da webcam (0 geralmente é a webcam padrão)
captura = cv2.VideoCapture(0)

# Verifica se a câmera abriu corretamente
if not captura.isOpened():
    print("Erro ao acessar a webcam.")
    exit()

# Parâmetros para o cálculo da distância
altura_real_garrafa = 0.223  # Altura média de uma garrafa comum em metros
altura_imagem = 480  # Altura da imagem capturada (em pixels)
fov_vertical = 60  # Campo de visão vertical da câmera em graus

# Função para estimar a distância usando a altura da bounding box
def estimar_distancia(bbox, altura_real, altura_imagem, fov_vertical):
    altura_bbox = bbox[3] - bbox[1]  # Altura da bounding box em pixels
    fov_vertical_rad = np.deg2rad(fov_vertical)
    distancia = (altura_real * altura_imagem) / (2 * altura_bbox * np.tan(fov_vertical_rad / 2) / 1.3)
    return distancia

# Função para processar o frame e manter apenas as bbox de "bottles"
def processar_frame(frame):
    resultados = modelo(frame)  # Faz a detecção
    frame_anotado = frame.copy()  # Faz uma cópia do frame original

    for r in resultados:
        for objeto in r.boxes:
            # Obtenha o índice da classe e o mapeie para o nome
            indice_classe = int(objeto.cls)
```

Fonte: Autoria própria

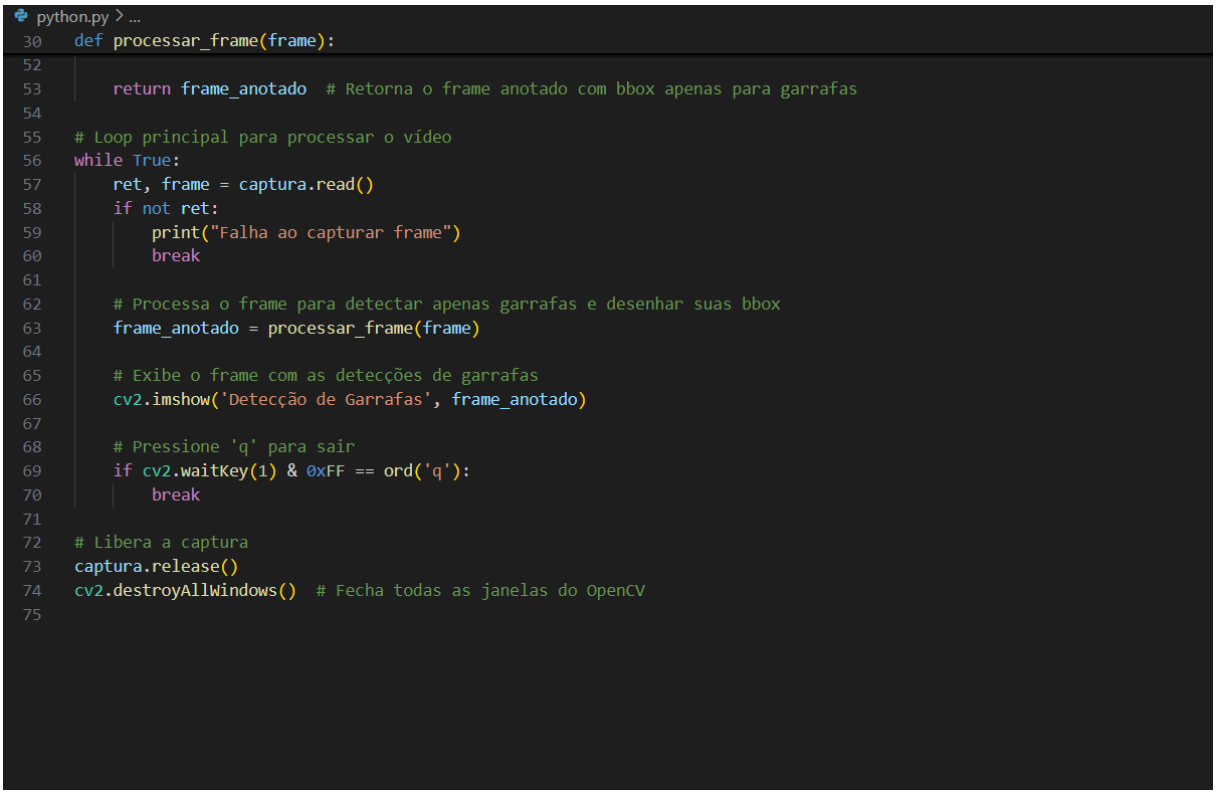
Figura 2: Segunda parte do código



```
python.py > ...
30  def processar_frame(frame):
34      for r in resultados:
35          for objeto in r.boxes:
36              # Obtenha o índice da classe e o mapeie para o nome
37              indice_classe = int(objeto.cls)
38              nome_objeto = modelo.names[indice_classe]  # Nome do objeto detectado
39
40              # Filtra para mostrar apenas o objeto 'bottle'
41              if nome_objeto == 'bottle':
42                  # Extrai coordenadas da bounding box e calcula a distância
43                  bbox = objeto.xyxy[0].cpu().numpy().astype(int)
44                  distancia = estimar_distancia(bbox, altura_real_garrafa, altura_imagem, fov_vertical)
45
46                  # Desenha a bounding box e o texto no frame
47                  cv2.rectangle(frame_anotado, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (0, 255, 0), 2)
48                  cv2.putText(frame_anotado, f'{nome_objeto} {objeto.conf.item():.2f}',
49                              (bbox[0], bbox[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
50                  cv2.putText(frame_anotado, f'Distancia: {distancia:.2f}m',
51                              (bbox[0], bbox[1] - 30), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
52
53      return frame_anotado  # Retorna o frame anotado com bbox apenas para garrafas
54
55  # Loop principal para processar o vídeo
56  while True:
57      ret, frame = captura.read()
58      if not ret:
59          print("Falha ao capturar frame")
60          break
61
62      # Processa o frame para detectar apenas garrafas e desenhar suas bbox
63      frame_anotado = processar_frame(frame)
64
65      # Exibe o frame com as detecções de garrafas
66      cv2.imshow('Detecção de Garrafas', frame_anotado)
67
68      # Pressione 'q' para sair
```

Fonte: Autoria própria

Figura 3: Terceira parte do código



```
python.py > ...
30  def processar_frame(frame):
52
53      return frame_anotado  # Retorna o frame anotado com bbox apenas para garrafas
54
55  # Loop principal para processar o vídeo
56  while True:
57      ret, frame = captura.read()
58      if not ret:
59          print("Falha ao capturar frame")
60          break
61
62      # Processa o frame para detectar apenas garrafas e desenhar suas bbox
63      frame_anotado = processar_frame(frame)
64
65      # Exibe o frame com as detecções de garrafas
66      cv2.imshow('Detecção de Garrafas', frame_anotado)
67
68      # Pressione 'q' para sair
69      if cv2.waitKey(1) & 0xFF == ord('q'):
70          break
71
72  # Libera a captura
73  captura.release()
74  cv2.destroyAllWindows()  # Fecha todas as janelas do OpenCV
75
```

Fonte: Autoria própria

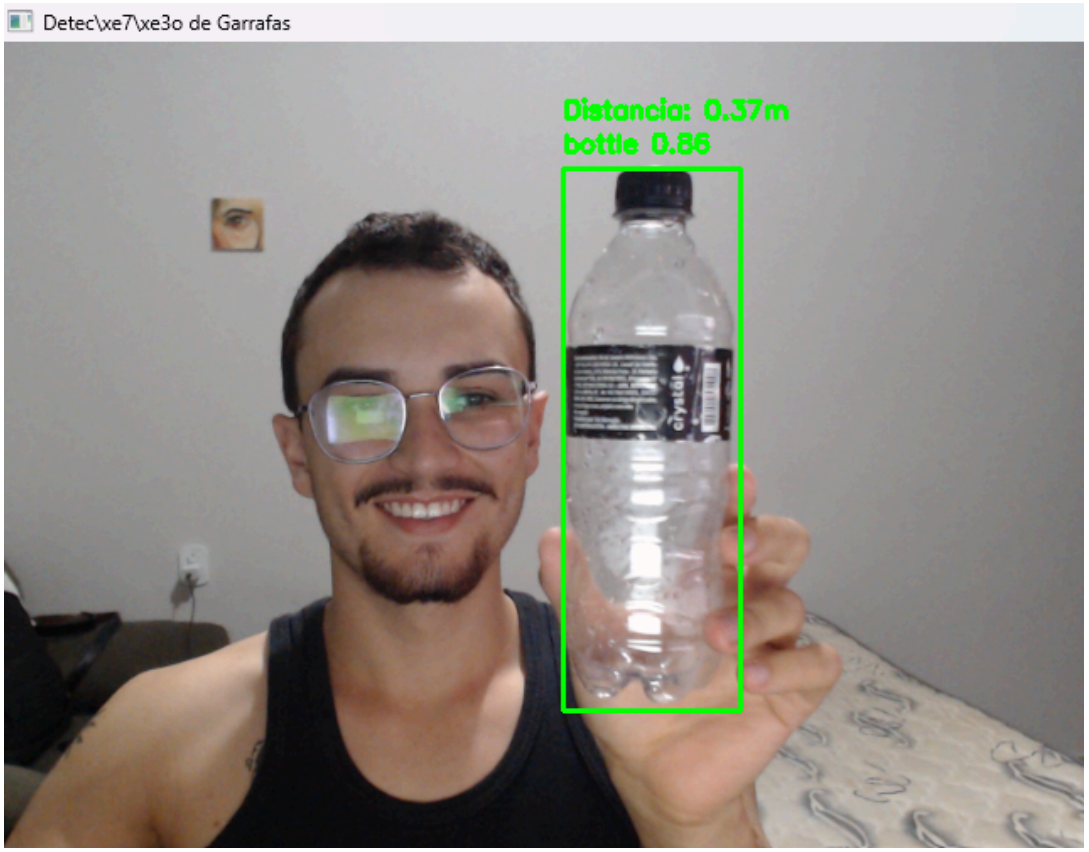
3.3.2 PROTÓTIPO ROBÓTICO: INTEGRAÇÃO DE MACHINE LEARNING E VISÃO COMPUTACIONAL

A integração de machine learning com visão computacional oferece uma abordagem poderosa para o desenvolvimento de protótipos robóticos, aplicando técnicas de aprendizado de máquina como classificação e regressão em tarefas de reconhecimento visual e interação com o ambiente (FULL SCALE, 2024).

Primeiramente, foi necessário compreender os conceitos de Regressão e Classificação, que se tornaram os pilares da proposta do projeto, conforme citado no tópico 3.3.1. Após essa base teórica, exploramos bibliotecas de visão computacional, identificando o OpenCV como a melhor escolha para captura de frames e o YOLO como a biblioteca mais eficiente para detectar objetos específicos, como garrafas, nas imagens de vídeo. A precisão no cálculo de distâncias foi alcançada utilizando princípios matemáticos de Geometria Analítica, como o Teorema de Pitágoras.

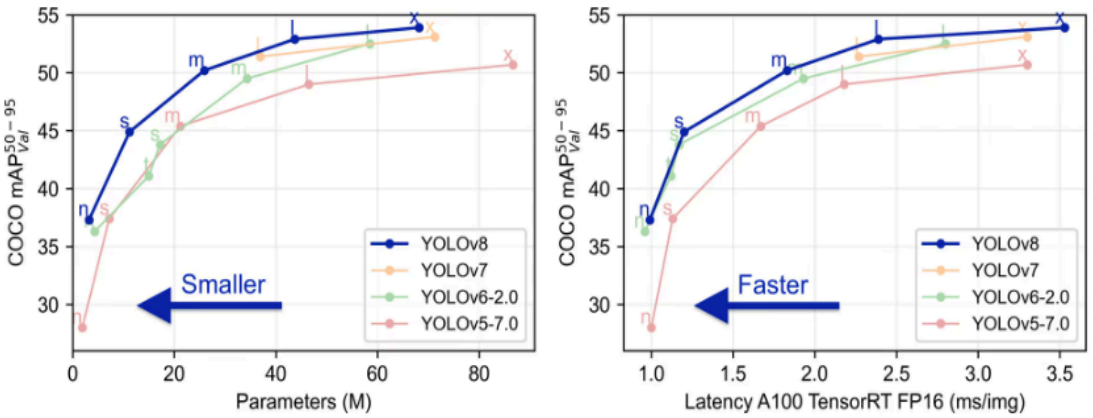
Os primeiros testes mostraram resultados limitados, com excesso de informações processadas, o que gerava lentidão no código. Com suporte de documentações, orientação de professores e IAs, encontramos soluções para problemas técnicos, levando a um aumento significativo na precisão e à redução de erro nos cálculos de distância para cerca de 2 cm, melhorando a eficiência do sistema.

Figura 4: Inteligência artificial identificando uma garrafa, e a distância da mesma utilizando uma câmera.



Fonte: Autoria própria

Figura 5: Gráfico da diferença entre as versões do algoritmo YOLO



Fonte: YOLOv8 - Ultralytics YOLO Docs

3.3.3 ABORDAGENS DE VALIDAÇÃO E OTIMIZAÇÃO DE MODELOS DE MACHINE LEARNING

Para validar e otimizar os modelos de machine learning desenvolvidos no projeto, foram implementadas abordagens sistemáticas que garantem não só a robustez, mas também a adaptabilidade da IA frente a novos cenários. A divisão dos dados em conjuntos de treinamento e teste permitiu avaliar a capacidade de generalização do modelo, enquanto a otimização de hiperparâmetros foi essencial para aprimorar a performance do YOLOv8, especialmente nas tarefas de detecção em tempo real. Com o uso de validação cruzada, verificamos a consistência dos resultados em diversos subconjuntos de dados, aprimorando a precisão e reduzindo significativamente falsos positivos e negativos. Esse processo detalhado de validação e ajuste é fundamental para que a IA alcance uma precisão elevada e se comporte de maneira confiável, qualidades indispensáveis para um sistema de automação robótica que precisa executar operações com precisão e segurança em ambiente dinâmico.

No desenvolvimento do código, o uso da biblioteca OpenCV para captura de frames em tempo real, aliado ao YOLOv8, forma uma combinação eficaz que permite

ao robô detectar e reagir a objetos de forma rápida e precisa. Ao inicializar com a versão leve yolov8n.pt, otimizamos o processamento em ambientes de hardware restrito, como o do robô autônomo. A função estimar distância foi projetada para calcular a distância entre o robô e os objetos detectados com base nas dimensões da bounding box, considerando variáveis como a altura real do objeto e o campo de visão da câmera. Esse cálculo permite ao robô não apenas reconhecer a garrafa, mas também posicionar-se de forma otimizada para a coleta, o que é essencial em uma aplicação prática de automação.

A função processar frame opera como o núcleo do sistema de visão, filtrando as detecções em tempo real e desenhando bounding boxes ao redor das latinhas. Acompanhadas de informações sobre o tipo de objeto, o grau de confiança e a distância, essas bounding boxes são atualizadas a cada frame, o que proporciona um monitoramento contínuo e facilita ajustes dinâmicos no ambiente. Esse fluxo de operações é encerrado com a liberação de recursos de memória e a finalização segura do sistema, prevenindo sobrecarga e garantindo a estabilidade do robô durante a execução. Dessa forma, a integração entre OpenCV e YOLOv8 fornece uma solução robusta e eficiente para detecção e coleta de objetos em tempo real, atendendo aos requisitos de precisão e confiabilidade estabelecidos para o projeto.

3.4 ÁLGEBRA LINEAR E GEOMETRIA ANALÍTICA

As transformações lineares são usadas para realizar cálculos sobre a posição do robô e dos objetos em seu ambiente. Esses cálculos vão incluir a rotação, translação e escala de vetores de posição, facilitando a determinação de onde o lixo ou os obstáculos estão em relação ao robô. A Álgebra Linear fornece ferramentas para lidar com essas transformações de maneira eficiente. Ponto de partida: $P = (x1, y1)$, Ponto de chegada: $C = (x2, y2)$

Para mover o robô ou determinar a movimentação necessária até o lixo, vamos usar a matriz de translação.

A rotação será utilizada para ajustar a direção das esteiras e calcular o movimento do braço robótico. As matrizes de rotação serão usadas para determinar como o braço precisa se mover ou girar em torno de um ponto fixo.

Para interpretar corretamente os dados dos sensores, a projeção de coordenadas tridimensionais para a tela bidimensional será feita usando matrizes de transformação linear. Isso é útil ao interpretar as distâncias e tamanhos dos objetos detectados.

A Geometria Analítica ajuda a descrever matematicamente as posições e trajetórias dos objetos no espaço. A localização do robô e dos obstáculos será representada por coordenadas, e vamos utilizar vetores para calcular o deslocamento e rota do robô até o alvo.

Cálculo de Distâncias: A fórmula da distância entre dois pontos:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Como nosso robô usa sensores (como sensores de distância, câmeras), ele recebe dados que precisam ser convertidos em informações úteis sobre o ambiente. A Álgebra Linear é usada para processar esses dados, aplicando transformações para determinar a posição dos obstáculos e do lixo em tempo real.

A Álgebra Linear também é aplicada no controle dos motores e do braço robótico, já que os ângulos e forças precisam ser cuidadosamente calculados para posicionar corretamente o braço e a garra.

3.4.1 VETORES

Em um sistema de coordenadas cartesianas, nós vamos representar a posição de ponto no espaço como um vetor. Para calcular a distância entre dois pontos Primeiro precisamos encontrar o ponto de partida e de chegada.

$P = (x1, y1)$

$C = (x2, y2)$

Segundo passo é **Calcular a magnitude do vetor diferença** A distância entre os pontos P e C é a magnitude (ou comprimento) desse vetor

$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$

O torque (ou momento de força) é uma medida da força que causa rotação em torno de um ponto de rotação ou eixo.

$\tau = r \, x \, F$

Onde τ (tau) é o torque.

r é o vetor posição do ponto onde a força é aplicada em relação ao eixo de rotação.

F é o vetor da força aplicada.

x representa o produto vetorial.

3.4.2 TRANSFORMAÇÕES LINEARES

A câmera que foi implantada em nosso robô vai capturar imagens do ambiente. Essas imagens são projeções 2D de um mundo tridimensional. Para detectar e identificar objetos como lixo, as transformações lineares desempenham um papel importante em **mapear as coordenadas da imagem** (2D) para o **espaço 3D** do robô. Isso inclui a **calibração da câmera**, **transformações de perspectiva** e o

reconhecimento de padrões. Calibração de Câmera: Para que o robô consiga saber onde está o objeto no mundo real a partir de uma imagem, o mesmo precisa **calibrar** a câmera. Isso envolve a aplicação de **transformações de perspectiva** para corrigir a distorção da lente e mapear as coordenadas de pixel da imagem para coordenadas reais do mundo 3D.

A transformação que ajusta a perspectiva é uma **transformação projetiva**, que será expressa por uma matriz de projeção 3D para 2D:

$$F = [fx\ 0\ cx,\ 0\ fy\ cy,\ 0\ 0\ 1]$$

Onde $fx\ e\ fy$ são os fatores de escala da câmera, e $cx\ e\ cy$ são as coordenadas do centro da imagem.

O robô se locomove com esteiras e motores de 180°, o que significa que ele precisa ajustar sua posição no espaço 2D. A movimentação envolve translações e rotações no plano, calculadas a partir das leituras da câmera e sensores. As transformações lineares aqui garantem que o robô mova-se eficientemente até o lixo detectado.

A locomoção do robô envolve movimentos rotacionais e translacionais. Para calcular a posição do robô em relação ao lixo, nós vamos aplicar uma matriz de rotação e translação:

Translação para mover o robô até a posição do lixo:

$$T = [1\ 0\ \Delta x,\ 0\ 1\ \Delta y,\ 0\ 0\ 1]$$

Onde Δx e Δy são os deslocamentos nas direções x e y.

Rotação para ajustar a direção do robô para alinhar com o objeto:

$$R(\theta) = [cos(\theta)\ (-\ sin(\theta))\ \theta,\ sin(\theta)\ cos(\theta)\ 0,\ 0\ 0\ 1]$$

Onde θ é o ângulo de rotação necessário.

Para que o **braço robótico com servos motores** possa pegar o lixo, nós usaremos **cinemática direta** e **cinemática inversa**, que são amplamente baseadas em transformações lineares e trigonometria.

A cinemática direta é usada para calcular a posição da ponta do braço (a garra) a partir dos ângulos de cada articulação (controlados pelos servos). Isso envolve uma série de multiplicações de matrizes de rotação e translação que descrevem cada elo do braço robótico.

Para um braço com 2 elos (segmentos), por exemplo:

Primeiro elo (comprimento $l1$, ângulo $\theta 1$)

$$T1 = [cos(\theta 1)\ (-sin(\theta 1))\ l1cos(\theta 1),\ sin(\theta 1)\ cos(\theta 1)\ l1sin(\theta 1),\ 0\ 0\ 1]$$

Segundo elo (comprimento $l2$, ângulo $\theta 2$)

$$T2 = [cos(\theta 2)\ (-sin(\theta 2))\ l2cos(\theta 2),\ sin(\theta 2)\ cos(\theta 2)\ l2sin(\theta 2),\ 0\ 0\ 1]$$

A posição final da garra seria $T = T1 \,.\, T2$

3.4.3 TRANSFORMAÇÕES RÍGIDAS

Em nosso projeto, que envolve uma câmera para detecção de objetos, um braço robótico com servos motores, e a locomoção com duas esteiras, as transformações rígidas são essenciais para:

Controlar os movimentos do robô no espaço enquanto ele se move e gira para se posicionar corretamente.

Alinhar o braço robótico para pegar objetos.

Integrar os dados da câmera e outros sensores ao sistema de coordenadas do robô.

Quando o robô se move em um plano (2D), ele realiza translações (movimento para frente/trás) e rotações (mudança de direção). Esses movimentos serão representados por **transformações rígidas** no espaço 2D.

A transformação rígida em 2D será descrita como a combinação de uma **translação** e uma **rotação**:

$$T = (\theta, d) = [cos(\theta)\ (-\ sin(\theta))\ dx,\ sin(\theta)\ cos(\theta)\ dy,\ 0\ 0\ 1]$$

Onde θ o ângulo de rotação, dx e dy é o vetor de translação (deslocamento), e o último valor na matriz é 1 porque estamos trabalhando em **coordenadas homogêneas**.

$$\Delta x = x2 - x1,\ \Delta y = y2 - y1$$

Além disso, o robô precisará mudar sua orientação (girar), uma **rotação** θ será aplicada. Juntas, a rotação e a translação formam a transformação rígida que posiciona o robô corretamente no espaço 2D.

Para o braço robótico manipular o lixo, ele também deve realizar movimentos rígidos, como **translações** e **rotações**. em nosso projeto, trabalhamos no espaço 3D, onde a transformação rígida envolve uma matriz de rotação R e um vetor de translação d

No espaço tridimensional, a transformação rígida é representada por uma matriz de 4 x 4 em coordenadas homogêneas, combinando a rotação e a translação:

$$T(R, d) = [R11\ R12\ R13\ dx,\ R21\ R22\ R23\ dy,\ R31\ R32\ R33\ dz,\ 0\ 0\ 1]$$

R é a matriz de rotação $3x3$ que descreve a orientação do braço robótico e $d = (dx, dy, dz)$ é o vetor de translação, que desloca o braço para uma nova posição no espaço.

Cada articulação do braço robótico será modelada com uma matriz de transformação rígida que inclui a rotação e a posição de cada elo.

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: DESENVOLVENDO IDEIAS

A Formação para a Vida é um dos eixos do Projeto Pedagógico de Formação por Competências da UNIFEOB.

Esta parte do projeto está diretamente relacionada com a extensão universitária, ou seja, o objetivo é que seja aplicável e que tenha real utilidade para a sociedade, de um modo geral.

3.5.1 DESENVOLVENDO IDEIAS

Tópico 1: Ideias e oportunidades

- Síntese: Todo projeto começa com uma ideia, e para que isso aconteça, é necessário que haja uma ou mais oportunidades, essas situações surgem a partir de problemas ou necessidades, sendo elas de interesse popular ou não.

- Exemplo prático: Um aplicativo de carona surgiu de um problema comum entre as pessoas: A falta de transporte, a partir disso, foi desenvolvido a ideia, que se transformou em uma oportunidade de negócio.

Tópico 2: Equipe

- Síntese: Uma equipe é indispensável para o desenvolvimento de uma ideia, o agrupamento de habilidades, criatividade, conhecimentos e experiências aumenta em muitos porcentos as chances de sucesso.

- Exemplo prático: Para desenvolver um aplicativo, por exemplo, é necessário que haja uma equipe com conhecimentos sobre programação mobile, que trabalhará diretamente com pessoas especializadas em design e marketing, que darão suporte ao visual e lançamento do produto, respectivamente. Todos são peças fundamentais para que sejam atendidas as necessidades do usuário.

Tópico 3: Fatores-chave de sucesso para o desenvolvimento das ideias

- Síntese: Para que uma ideia tenha sucesso é preciso planejamento, oportunidade, recursos e principalmente necessidade, uma ideia pode ser inovadora e muito boa, mas se não há público para ela, essa ideia se tornará "inútil".

- Exemplo prático: Uma empresa de tecnologia que investe em uma boa equipe, planeja seus projetos e possuem os recursos necessários, tendem a desenvolver projetos e ideias de sucesso, diferentemente de uma empresa que não se adapta ao mercado com qualidade

Tópico 4: Definindo uma ideia empreendedora.

- Síntese: Uma ideia empreendedora pode ser considerada da seguinte forma, uma resolução de um problema ou forma de preencher uma lacuna no mercado em geral, pode ser com intuito de gerar receita. Definir uma ideia desse tipo envolve analisar o mercado, e se a ideia é realmente útil para o momento atual ou viável, tanto financeiramente, quanto tecnicamente.

- Exemplo prático: Uma startup que oferece cursos de programação e inglês de forma criativa, utilizando como formato de aprendizado, filmes e jogos, resolvendo problemas como falta de atenção ou falta de interesse nos estudos, e torna-se uma possível renda financeira.

3.5.2 ESTUDANTES NA PRÁTICA

Figura 6: Banner sobre como usar o Canvas Model

Fonte: Autoria própria

O objetivo desse material é ajudar as pessoas a realizarem o planejamento de forma mais fácil e evitar que essa etapa seja pulada em qualquer projeto ou ideia.

Muitas ideias surgem ao longo da vida das pessoas e não podem ser desperdiçadas pela falta de um bom planejamento.

A metodologia do Canvas pode contribuir muito para a materialização de ideias que podem trazer muitos benefícios para as pessoas.

Caso a equipe escolha pelo vídeo, poderá ser gravado de forma bem simples e ser disponibilizado em algum canal do Youtube de seus integrantes, como “Não Listado”. Se a equipe se sentir à vontade, também pode compartilhar esse vídeo nas redes sociais, por exemplo, no Instagram e marcar a Escola de Negócios usando @ednunifeob.

O objetivo é que todos os integrantes da equipe participem desse material e que possam compartilhá-lo para que as pessoas da comunidade onde estejam inseridos possam a partir deste material construir com maior eficácia o planejamento de projetos e ideias em suas vidas.

Portanto, neste tópico do projeto, a equipe deve elaborar um pequeno texto descrevendo o conteúdo desse material e, em seguida, colocar o link público do arquivo para que possa ser verificado e avaliado.

A divulgação e compartilhamento desse material para a comunidade externa será de responsabilidade dos próprios estudantes, conforme o compromisso social de cada pessoa, pois a UNIFEOB apenas usará esse material para avaliação desta atividade.

OBSERVAÇÃO: A realização do item 3.5 deste projeto é uma atividade que integra parte do conteúdo da unidade de Formação para a Vida com o Projeto Integrado, portanto, **não exclui a obrigatoriedade** do estudante de realizar os desafios e demais atividades disponibilizados para a unidade de estudo de Formação para a Vida.

4 CONCLUSÃO

Conclui-se que o projeto “Automação Robótica: Soluções Sustentáveis e Inclusivas” representa um avanço significativo no campo da robótica aplicada, oferecendo uma solução acessível e prática para o manuseio de objetos em ambientes controlados. Combinando um mini-computador, peças produzidas em impressora 3D e uma câmera integrada a um sistema de inteligência artificial para detecção de objetos, o robô foi projetado para executar tarefas de coleta com precisão, mostrando o potencial da robótica em atividades cotidianas e industriais.

Durante o desenvolvimento, a equipe enfrentou e superou inúmeros desafios técnicos e conceituais. Desde a integração dos componentes eletrônicos, como o sistema Raspberry, até a calibração precisa da câmera e da garra, cada dificuldade proporcionou um aprendizado significativo, evidenciando a importância da colaboração, da precisão e do rigor técnico para alcançar um sistema funcional e eficiente. Esse processo de superação fortaleceu a equipe e agregou valor ao projeto.

A proposta do robô do grupo Małpa é um exemplo de como a inteligência artificial e a automação podem viabilizar soluções robóticas sustentáveis e inclusivas, acessíveis tanto para aplicação prática quanto para fins educacionais. As expectativas de sucesso envolvem a possibilidade de ampliar suas funcionalidades, tornando o robô ainda mais adaptável a diferentes cenários e aprimorando seu desempenho em ambientes complexos.

REFERÊNCIAS

FULL SCALE. *AI and Machine Learning: Sustainable Technologies to a Greener Future*. Disponível em: https://fullscale.io. Acesso em: 1 nov. 2024.

JUNIOR, Flávio L. P.; GOULART, Cleiton S.; TORRES, Fernando E.; et al. *Robótica*. Porto Alegre: SAGAH, 2019. E-book. ISBN 9788595029125. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595029125/. Acesso em: 3 nov. 2024.

KELLY, R. *Introduction to Robotics*. McGraw-Hill Education, 2013. Disponível em: https://highered.mheducation.com/sites. Acesso em: 23 out. 2024.

LaVALLE, S. M. *Planning Algorithms*. Cambridge University Press. Disponível em: <https://www.routledge.com/Differentially-Flat-Systems/Stics: Mechanics and Control*. Pearson>. Acesso em: 30 out. 2024.

ROMERO, Roseli Aparecida F.; PRESTES, Edson; OSÓRIO, Fernando; et al. *Robótica Móvel*. Rio de Janeiro: Grupo GEN, 2014. E-book. ISBN 978-85-216-2642-8. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-2642-8/. Acesso em: 6 set. 2024.

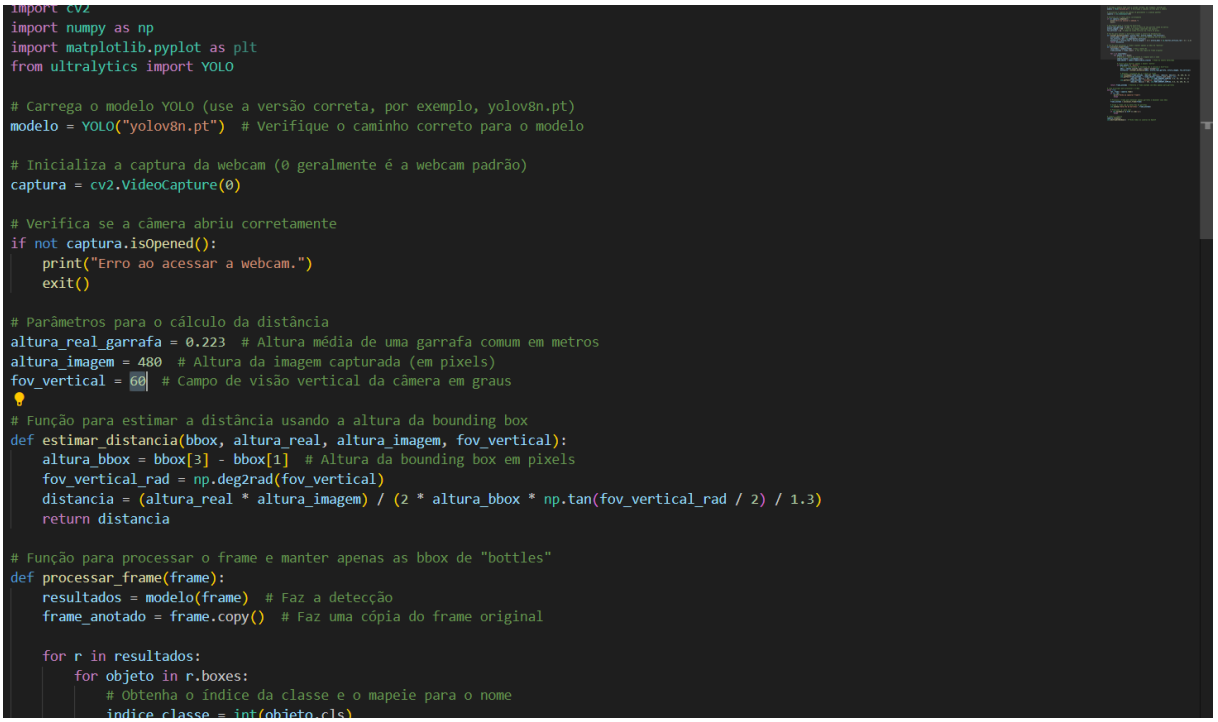
SICILIANO, B.; KHATIB, O.; et al. *Springer Handbook of Robotics*. Disponível em: https://link.springer.com/book/. Acesso em: 30 out. 2024.

SIRA-RAMÍREZ, H.; AGUIRRE, L.; KOROTAEV, S. K. *Differentially Flat Systems*. Marcel Dekker, 2004. Disponível em: https://www.routledge.com/Differentially-Flat-Systems/. Acesso em: 16 out. 2024.

YOSHIKAWA, T. *Foundations of Robotics: Analysis and Control*. MIT Press, 1990. Disponível em: https://mitpress.mit.edu/9780262514652/. Acesso em: 1 nov. 2024.

ANEXOS

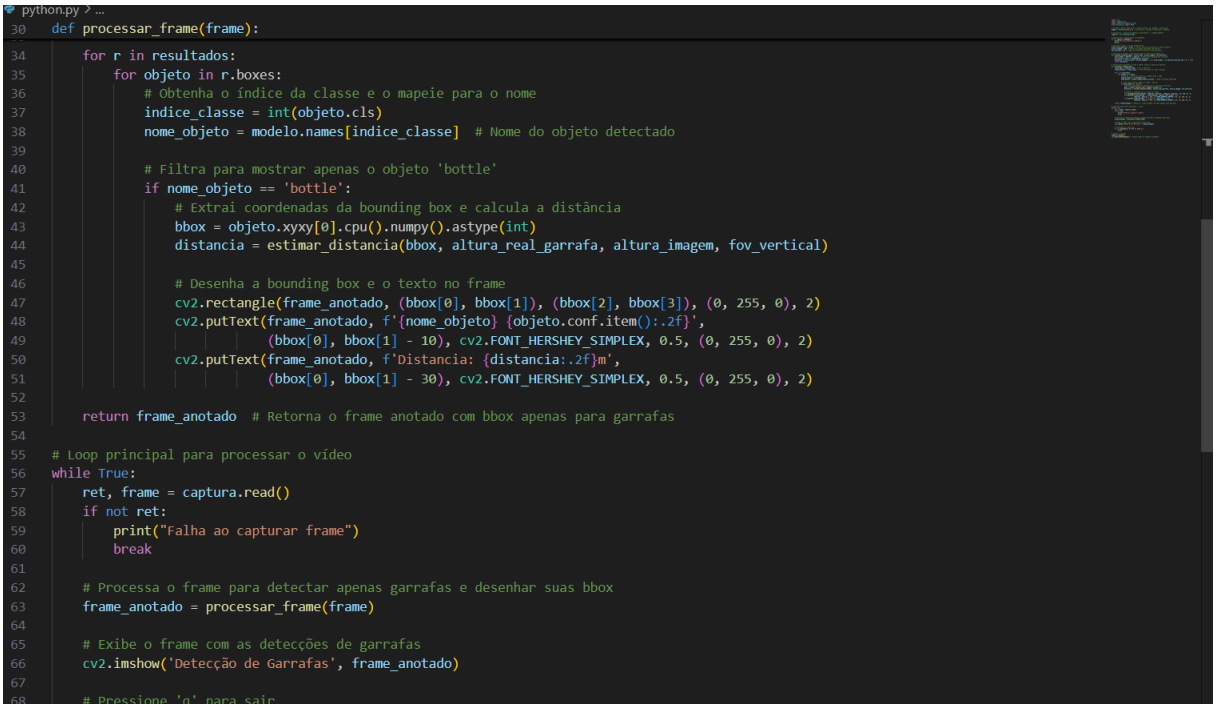
Figura 1: Primeira parte do código



```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from ultralytics import YOLO

# Carrega o modelo YOLO (use a versão correta, por exemplo, yolov8n.pt)
modelo = YOLO("yolov8n.pt")  # Verifique o caminho correto para o modelo

# Inicializa a captura da webcam (0 geralmente é a webcam padrão)
captura = cv2.VideoCapture(0)

# Verifica se a câmera abriu corretamente
if not captura.isOpened():
    print("Erro ao acessar a webcam.")
    exit()

# Parâmetros para o cálculo da distância
altura_real_garrafa = 0.223  # Altura média de uma garrafa comum em metros
altura_imagem = 480  # Altura da imagem capturada (em pixels)
fov_vertical = 60  # Campo de visão vertical da câmera em graus

# Função para estimar a distância usando a altura da bounding box
def estimar_distancia(bbox, altura_real, altura_imagem, fov_vertical):
    altura_bbox = bbox[3] - bbox[1]  # Altura da bounding box em pixels
    fov_vertical_rad = np.deg2rad(fov_vertical)
    distancia = (altura_real * altura_imagem) / (2 * altura_bbox * np.tan(fov_vertical_rad / 2) / 1.3)
    return distancia

# Função para processar o frame e manter apenas as bbox de "bottles"
def processar_frame(frame):
    resultados = modelo(frame)  # Faz a detecção
    frame_anotado = frame.copy()  # Faz uma cópia do frame original

    for r in resultados:
        for objeto in r.boxes:
            # Obtenha o índice da classe e o mapeie para o nome
            indice_classe = int(objeto.cls)
```

Fonte: Autoria própria

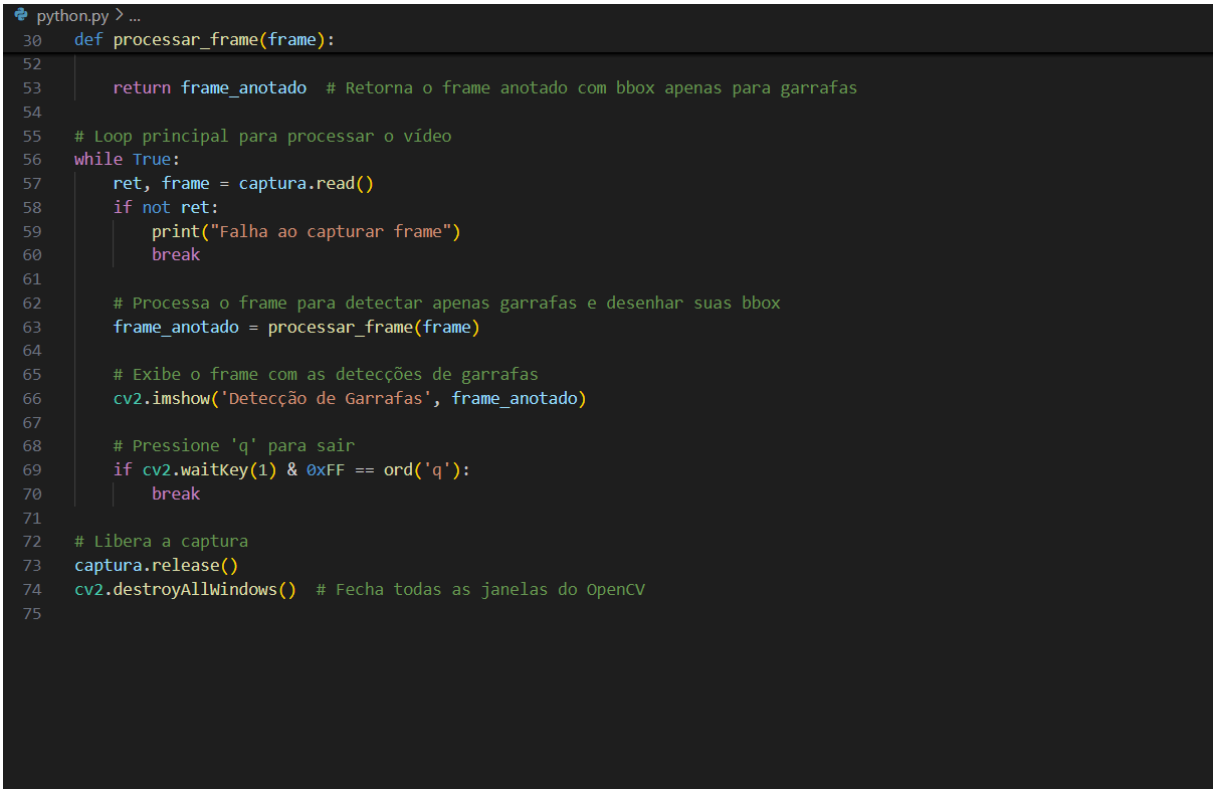
Figura 2: Segunda parte do código



```
python.py > ...
30  def processar_frame(frame):
34      for r in resultados:
35          for objeto in r.boxes:
36              # Obtenha o índice da classe e o mapeie para o nome
37              indice_classe = int(objeto.cls)
38              nome_objeto = modelo.names[indice_classe]  # Nome do objeto detectado
39
40              # Filtra para mostrar apenas o objeto 'bottle'
41              if nome_objeto == 'bottle':
42                  # Extrai coordenadas da bounding box e calcula a distância
43                  bbox = objeto.xyxy[0].cpu().numpy().astype(int)
44                  distancia = estimar_distancia(bbox, altura_real_garrafa, altura_imagem, fov_vertical)
45
46                  # Desenha a bounding box e o texto no frame
47                  cv2.rectangle(frame_anotado, (bbox[0], bbox[1]), (bbox[2], bbox[3]), (0, 255, 0), 2)
48                  cv2.putText(frame_anotado, f'{nome_objeto} {objeto.conf.item():.2f}',
49                              (bbox[0], bbox[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
50                  cv2.putText(frame_anotado, f'Distancia: {distancia:.2f}m',
51                              (bbox[0], bbox[1] - 30), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
52
53      return frame_anotado  # Retorna o frame anotado com bbox apenas para garrafas
54
55  # Loop principal para processar o vídeo
56  while True:
57      ret, frame = captura.read()
58      if not ret:
59          print("Falha ao capturar frame")
60          break
61
62      # Processa o frame para detectar apenas garrafas e desenhar suas bbox
63      frame_anotado = processar_frame(frame)
64
65      # Exibe o frame com as detecções de garrafas
66      cv2.imshow('Detecção de Garrafas', frame_anotado)
67
68      # Pressione 'q' para sair
```

Fonte: Autoria própria

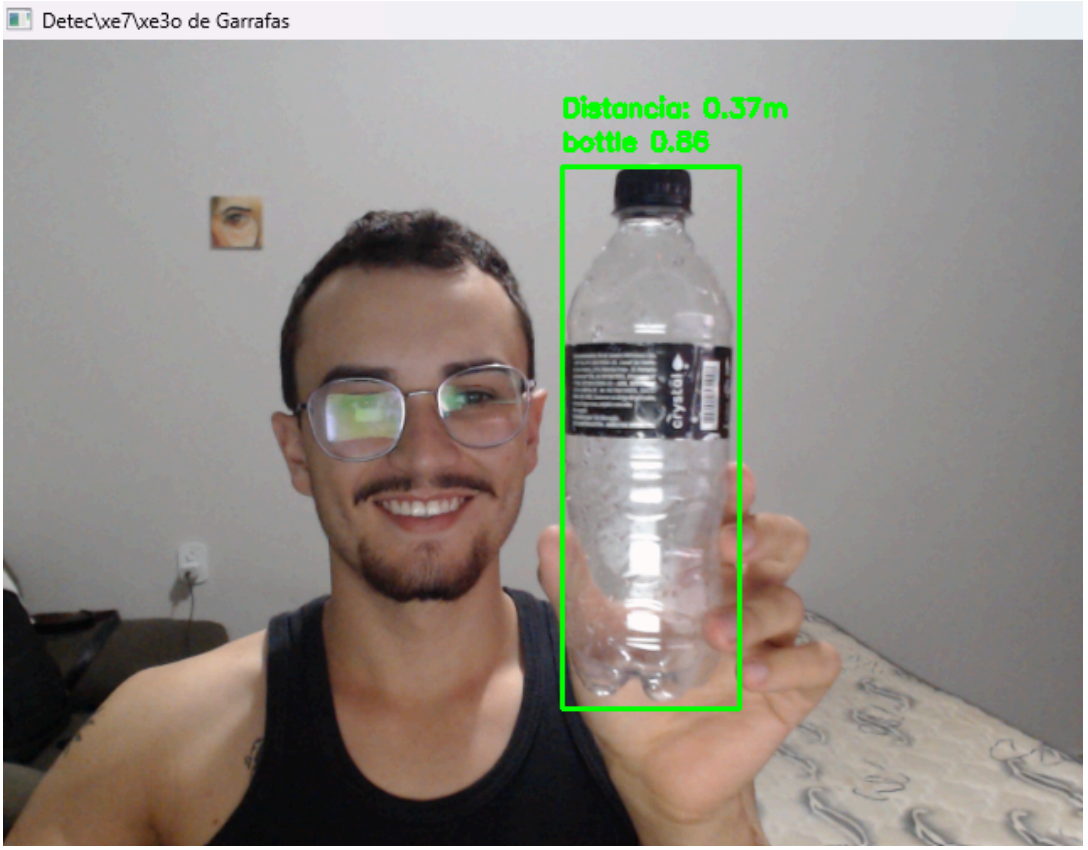
Figura 3: Terceira parte do código



```
python.py > ...
30  def processar_frame(frame):
52
53      return frame_anotado  # Retorna o frame anotado com bbox apenas para garrafas
54
55  # Loop principal para processar o vídeo
56  while True:
57      ret, frame = captura.read()
58      if not ret:
59          print("Falha ao capturar frame")
60          break
61
62      # Processa o frame para detectar apenas garrafas e desenhar suas bbox
63      frame_anotado = processar_frame(frame)
64
65      # Exibe o frame com as detecções de garrafas
66      cv2.imshow('Detecção de Garrafas', frame_anotado)
67
68      # Pressione 'q' para sair
69      if cv2.waitKey(1) & 0xFF == ord('q'):
70          break
71
72  # Libera a captura
73  captura.release()
74  cv2.destroyAllWindows()  # Fecha todas as janelas do OpenCV
75
```

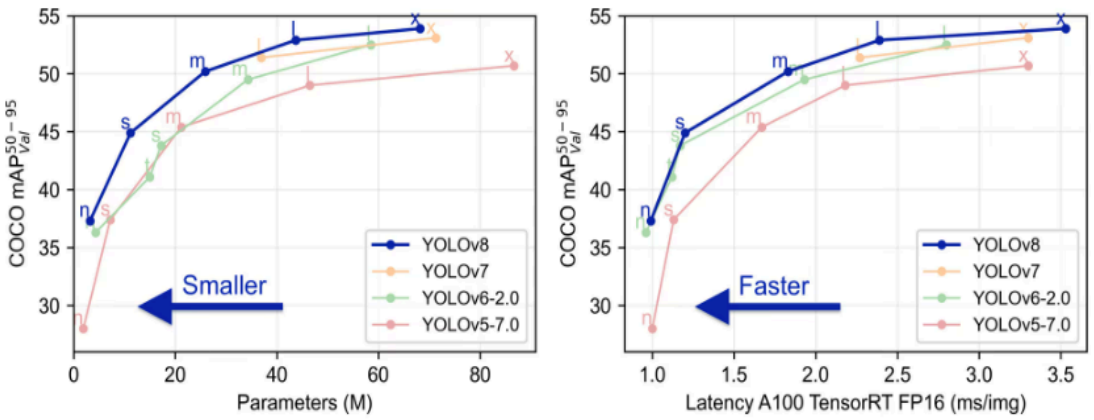
Fonte: Autoria própria

Figura 4: Inteligência artificial identificando uma garrafa, e a distância da mesma utilizando uma câmera.



Fonte: Autoria própria

Figura 5: Gráfico da diferença entre as versões do algoritmo YOLO

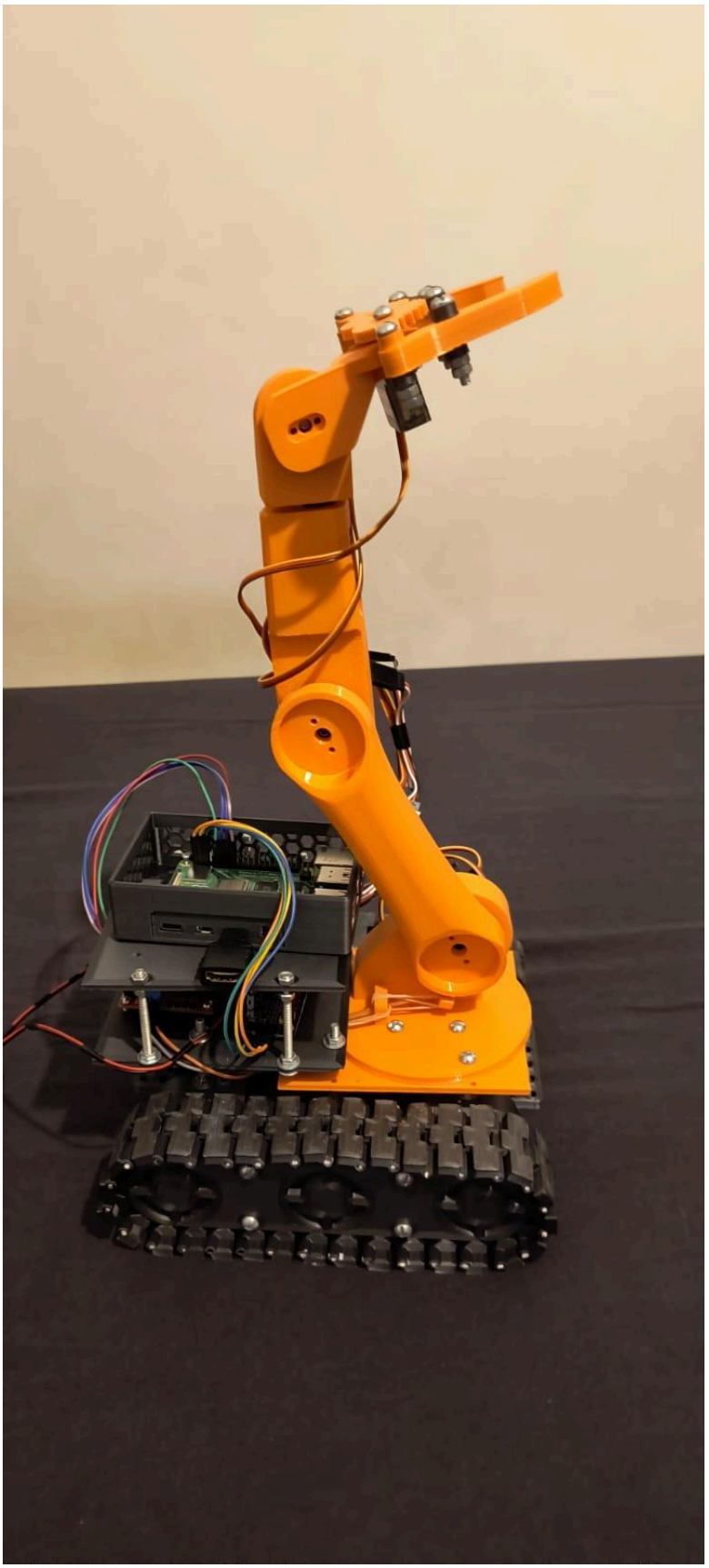


Fonte: YOLOv8 - Ultralytics YOLO Docs

Figura 6: Banner sobre como usar o Canvas Model

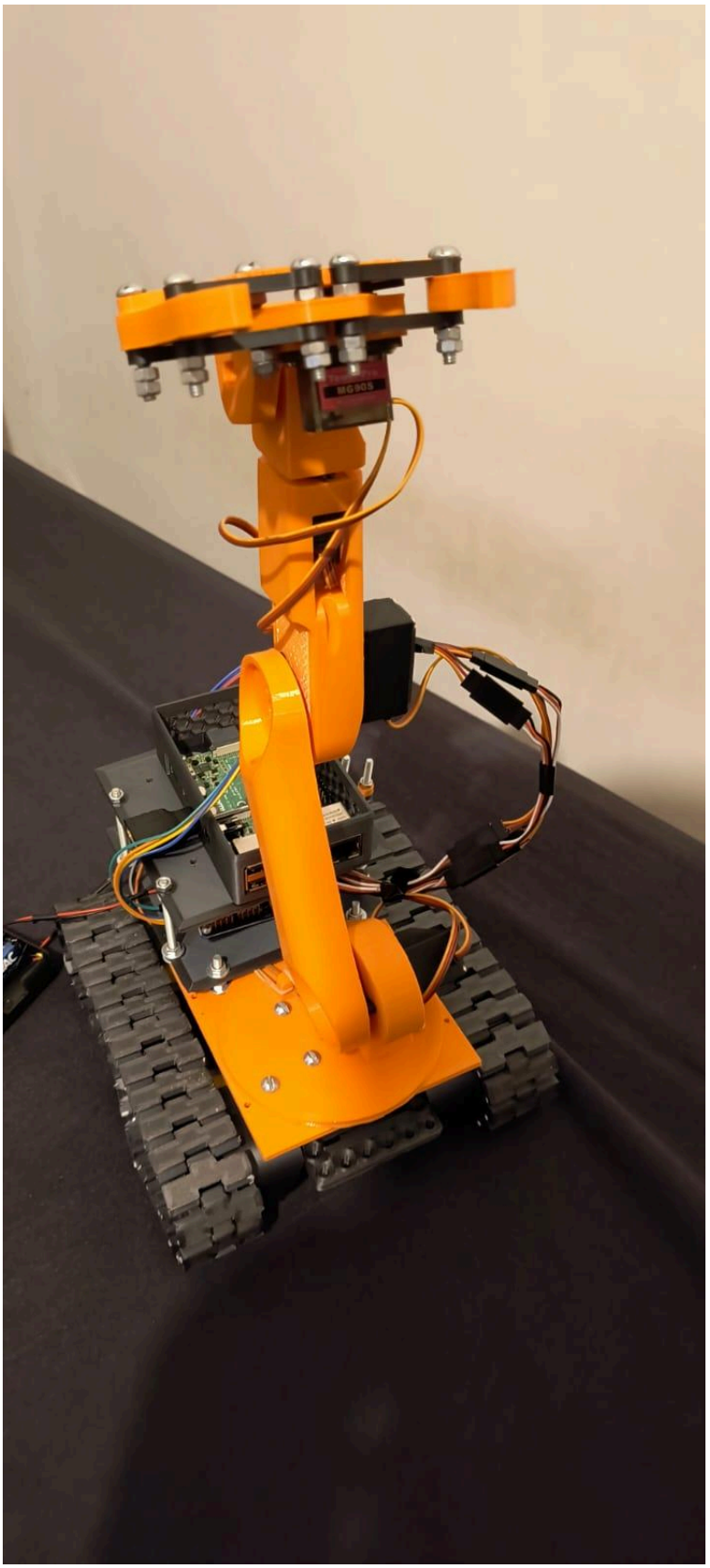
Fonte: Autoria própria

Figura 7: Fotografia de nosso robô pronto visto do lado



Fonte: Autoria própria

Figura 8: Fotografia de nosso robô pronto visto de frente



Fonte: Autoria própria