INSTITUTO FEDERAL DO ESPÍRITO SANTO

MESTRADO PROFISSIONAL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

RAFAEL MENEGUELLI

UTILIZAÇÃO DE VISÃO COMPUTACIONAL E ROBÓTICA COLABORATIVA PARA A COLETA SELETIVA DE LIXO

Serra

2022

RAFAEL MENEGUELLI

UTILIZAÇÃO DE VISÃO COMPUTACIONAL E ROBÓTICA COLABORATIVA PARA A COLETA SELETIVA DE LIXO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo para obtenção do Título de Mestre em Engenharia de Controle e Automação

Orientador: Prof. Dr. Cassius Zanetti Resende

Coorientador: Prof. Dr. Daniel Cruz Cavalieri

Serra

2022

Dados Internacionais de Catalogação na Publicação (CIP)

M541u 2022

Meneguelli, Rafael
Utilização de visão computacional e robótica colaborativa para a coleta seletiva de lixo / Rafael Meneguelli. - 2022.
88 f.; il.; 30 cm

Orientador: Prof. Dr. Cassius Zanetti Resende.
Coorientador: Prof. Dr. Daniel Cruz Cavalieri.

Dissertação (mestrado) - Instituto Federal do Espírito Santo, Programa de Pós-graduação em Engenharia de Controle de Automação, 2022.

1. Robótica. 2. Visão por computador. 3. Coleta seletiva de lixo. 4. Resíduos sólidos - Reaproveitamento. 5. Robôs. I. Resende, Cassius Zanett. II. Cavalieri, Daniel Cruz. III. Instituto Federal do Espírito Santo. IIV Título.

CDD 629.892

Bibliotecária Rogeria Gomes Belchior - CRB6/ES 417

MINISTÉRIO DA EDUCAÇÃO

INSTITUTO FEDERAL DO ESPÍRITO SANTO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

RAFAEL MENEGUELLI

UTILIZAÇÃO DE VISÃO COMPUTACIONAL E ROBÓTICA COLABORATIVA NA COLETA SELETIVA DE LIXO

Dissertação apresentada ao Programa de PósGraduação em Engenharia de Controle e Automação do Instituto Federal do Espírito Santo, como requisito parcial para obtenção de título de Mestre em Engenharia de Controle e Automação.

Aprovado em 13 de julho de 2022

COMISSÃO EXAMINADORA

Prof. Dr. Cassius Zanetti Resende
Instituto Federal do Espírito Santo Orientador

Prof. Dr. Daniel Cruz Cavalieri
Instituto Federal do Espírito Santo
Coorientador

Prof. Dr. Gustavo Maia de Almeida
Instituto Federal do Espírito Santo
Membro Interno

Prof. Dr. Filipe Ieda Fazanaro
Universidade Federal do Espírito Santo
Membro Externo

RESUMO

Um dos maiores problemas ambientais que a humanidade vem enfrentando é a quantidade de resíduos gerados. O Brasil é um dos países do mundo que mais gera resíduos sólidos urbanos e que possui um dos maiores contingentes de pessoas que separam esses resíduos manualmente, muitas vezes em deploráveis condições de trabalho. A coleta seletiva consiste basicamente na separação de resíduos recicláveis. A utilização de sistemas automatizados é uma alternativa para tornar a separação de resíduos mais eficiente e segura. Assim, este trabalho descreve a implementação de um sistema para separação dos materiais, combinando a atuação de um robô colaborativo e o uso de métodos baseados em visão computacional utilizados para detectar 04 tipos principais de resíduos sólidos, a saber, vidro, metal, papel e plástico. Para detectar cada tipo de resíduo sólido, foram utilizadas duas redes neurais convolucionais distintas, a Mask R-CNN e a YoloV5. Os resultados obtidos mostram que a pose dos objetos é obtida com uma precisão adequada para a tarefa utilizando-se ambos *frameworks*, contudo a rede YoloV5 se demostrou mais adequada ao problema, pois foi capaz de classificar objetos na imagem com tempo inferior a 0,01s. Nos experimentos práticos realizados, o robô colaborativo se desloca até a pose calculada, captura o resíduo e o deposita no recipiente correspondente a seu tipo automaticamente, simulando um sistema industrial de separação de material reciclável. Os comandos de controle de movimento e a integração do sistema de visão foram realizadas utilizando-se a linguagem *Python*. Ressalta-se que foi desenvolvida uma biblioteca própria para o envio de comandos de movimentação do robô. O sistema protótipo desenvolvido demonstra-se robusto e cumpre com o objetivo do trabalho, que é realizar a coleta seletiva do lixo de forma autônoma.

Palavras-chave: Robô colaborativo. Deep Learning. Mask-RCNN. YoloV5. UR3

ABSTRACT

One of the biggest environmental problems that humanity has been facing is the amount of waste generated. The disorderly growth of large cities combined with consumption and industrialization are substantially increasing the amount of solid waste generation. Brazil is one of the countries in the world where most generate solid urban waste and which has one of the largest numbers of people who separate this waste manually, many times in deplorable working conditions. Selective waste sorting basically consists of segregation of recyclable waste. However, when performed manually, the practice of segregation may not be followed evenly. The use of automated systems are an alternative to make the waste segregation more efficient and safer. This work describes a system implementation combining a method based on computer vision to detect 4 main types of solid waste, glass, metal, paper and plastic, and a collaborative robot to perform the separation of identified materials. To classify and detect each type of solid waste, Convolutional Neural Networks, the Mask R-CNN and YoloV5, was used. Results obtained by the computer vision models allows the objects detection with high precision and the YoloV5, showed itself as the ideal framework for solving the proposed problem since it's capable of detecting objects with an image with a time of less than 0.01s. Sorting generates a bounding box and object position information in the image. The collaborative robot moves to the determined position at the image, using a library developed to perform its movement, captures the waste, and automatically deposits it in the container corresponding to its type, simulating an industrial system for separating recyclable material. The prototype system developed proves to be robust and fulfills the objective of the work, that is perform the autonomous waste sorting.

Key Words: Collaborative Robots. Deep Learning. Mask-RCNN. YoloV5 UR3

LISTA DE FIGURAS

Figura 1 - Países com a maior quantidade de catadores de recicláveis no mundo e proporção de materiais em todo resíduo sólido urbano gerado no mundo. 13

Figura 2 - Arquitetura básica do trabalho de Kulcke et al. 16

Figura 3 - Representação do protótipo desenvolvido 18

Figura 4 - Modos de interação entre homem e robô. 21

Figura 5 - Robô UR3. 24

Figura 6 - Interface de operação e programação do cobot UR3e. 25

Figura 7 - Interface gráfica de operação e programação do robô. 25

Figura 8 - Exemplo de programação sequencial de movimentação do robô. 26

Figura 9 - Estrutura do programa de *pick and place* utilizando-se script. 27

Figura 10 - Tela de configurações do UR3e e botão responsável por habilitar a comunicação em Ethernet/IP 27

Figura 11 - Representação do sistema de orientação vetor-angulo. O eixo em azul representa a orientação de um corpo antes da rotação desse mesmo eixo em θ graus em torno do vetor u. O eixo vermelho ilustra a orientação final. 29

Figura 12 - Representação das rotações em torno dos eixos no UR3e. 30

Figura 13 - Estrutura padrão de uma rede neural convolucional. 33

Figura 14 - Comparação dos índices de erro de treinamento e teste em redes de 20 e 56 camadas. 34

Figura 15 - Representação básica de uma rede neural plana. 35

Figura 16 - Representação básica de uma ResNet. 36

Figura 17 - Treinamento utilizando o dataset ImageNet. 36

Figura 18 - Resultados da aplicação da Mask RCNN na classificação das imagens do COCO dataset. O modelo utiliza a ResNet-101 em sua implementação. 38

Figura 19 - Arquitetura básica da rede Faster RCNN. 38

Figura 20 - Arquitetura básica do modelo Mask RCNN. 39

Figura 21 – Refinamento da RoI utilização o método da supressão máxima 40

Figura 22 – Diferença entre RoI Pooling e RoI Aligment. 41

Figura 23 - Plotagem das curvas de precisão/recall para cada indice de IoU. A média da sumarização de formato dessas curvas em cada um dos limiares de IoU compõem a mAP. 43

Figura 24 - Exemplo de detecção utilizando a rede YOLOv5 pré treinada com o COCO dataset 44

Figura 26 - Exemplos de imagens da base de dados TrashNet. 47

Figura 27 - Processo de demarcação manual da posição dos objetos na imagem utilizando o *software VGG Image Annotator*. 48

Figura 28 - Processo de demarcação manual utilizando o *software LabelIMG* para o treinamento do modelo YOLOv5 49

Figura 29 - Diagrama de blocos que representa o processo de aprimoramento do modelo. 53

Figura 30 - Ilustração da caixa delimitadora gerada pela MaskRCNN. 54

Figura 31 - Caixa delimitadora a ser gerada pelo algoritmo. 55

Figura 32- Imagem de garrafa retirada do *dataset* TrashNet 55

Figura 33 - Máscara gerada pela classificação utilizando Mask-RCNN. 56

Figura 34 - Utilização dos métodos *findContours e minAreaRect* da biblioteca OpenCV. Com a aplicação dos métodos, é possível resolver o problema da detecção da orientação dos objetos classificados. 57

Figura 35 - Utilização do método de binarização de Otsu para geração de máscara e determinação de orientação de objeto utilizando a YOLOv5 58

Figura 36 - Exemplo de envio de comandos e recebimento de informações do robô via Python 61

Figura 37 - Arquitetura básica do sistema proposto. 61

Figura 38 - Representação do método utilizado para a determinação das coordenadas do objeto na imagem 63

Figura 39 - Representação do método utilizado para determinação da relação pixel/milímetros. 63

Figura 40 - Relação entre o ângulo encontrado pelo sistema de visão computacional e o ângulo da garra do robô. Observa-se o objeto em diferentes disposições em a) e b). 64

Figura 41 - Erro no treinamento dos diferentes modelos 67

Figura 42 - Erro da classificação no processo de validação 68

Figura 43- Valores de mAP para cada modelo testado. 68

Figura 44 - Matrizes de confusão geradas pelos modelos propostos com dados normalizados. 69

Figura 45 - Erros de treinamento e validação obtido sem a classe vidro pelo modelo 3. 70

Figura 46 - Matriz de confusão normalizada gerada pelo modelo 3 sem a classe vidro. 70

Figura 47 - Classificação de imagens utilizando o modelo 3. 71

Figura 48 - Imagens classificadas utilizando imagens de *webcam* como entrada do modelo 3. 72

Figura 49 - Valores de erro obtidos no processo de treinamento da YoloV5 73

Figura 50 - Erro obtido no processo de validação do treinamento da YoloV5 74

Figura 51 - Valores de mAP obtidos pelos modelos treinados 74

Figura 52 - Matrizes de confusão geradas pelos modelos treinados 75

Figura 53 - Matriz de confusão gerada no treinamento do modelo 4 sem a classe vidro 76

Figura 54 - Exemplos de resultados de classificação gerados pelo modelo 4. 76

Figura 55 - Detecção de objeto plástico no plano de trabalho. Sobre a *bounding box* estão representados respectivamente a classe à qual o elemento pertence, a precisão da classificação e as coordenadas em x e y em milímetros. 79

Figura 56- Robô se posicionando sobre o objeto detectado 79

Figura 57 - Robô depositando material na lixeira adequada 80

Figura 58 - Detecção de dois objetos de classes diferentes no plano de trabalho 81

Figura 59 - Representação da priorização da classe plástico quando há a classificação de mais de um elemento na imagem. 81

Figura 60 - Alocação do objeto plástico na lixeira determinada para essa classe de resíduo .. 82

Figura 61 - Aquisição do objeto metálico na sequência do descarte do objeto plástico 82

Figura 62 - Descarte do objeto metálico no local apropriado..... 83

LISTA DE TABELAS

Tabela 1 - Aplicações do cobot na indústria. 22

Tabela 2 - Parâmetros dos *frameworks (*MASK RCNN e YOLOv5) alterados a fim de obter-se o melhor resultado. 51

Tabela 3 - Tipos diferentes de aumentação de dados. 52

Tabela 4 - Mapa de comunicação do UR3, disponibilizado na porta 80 da rede. Parâmetros utilizados no controle do robô. 59

Tabela 5 - Diferentes conjuntos de parâmetros de treinamento testados durante o aprimoramento do modelo da Mask RCNN. 66

Tabela 6 - Diferentes conjuntos de parâmetros de treinamento testados durante o aprimoramento do modelo da YoloV5. 73

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

2D – Bidimensional

AP – *Average Precision*

CNN – *Convolution Neural Network*

COCO – *Common Objects in Context*

COBOT – *Collaborative Robot*

FCN – *Fully Connected Networks*

FPN – *Feature Pyramid Network*

FPS – *Frames Per Second*

IoU – *Intercept Over Union*

mAP – *Minimun Average Precision*

NIR – *Near Infrared*.

QDR – *Quadratic Discriminant Classifier*

ReLU – *Rectified Linear Activation Unit*

RGB – *Red, Green and Blue.*

RoI – *Region of Interest*

RSU – Resíduo Sólido Urbano

SAM – *Spectral Angle Mapper*

SVM – *Support Vector Machine*

RPN – *Region Proposal Network*

TCP – *Tool Center Point*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 TRABALHOS CORRELATOS 15
1.2 OBJETIVOS 18
1.2.1 Objetivo geral 18
1.2.2 Objetivos específicos 18
1.3 ESTRUTURA DA DISSERTAÇÃO 19
2 REFERENCIAL TEÓRICO 20
2.1 ROBÔS COLABORATIVOS 20
2.1.1 Utilização da robótica colaborativa na indústria 20
2.2 ROBÔ UR3E 23
2.2.1 Operação e Possibilidades de integração do robô UR3e 24
2.2.2 Sistema de Coordenadas e Orientação do robô UR3e 28
2.3 DEEP LEARNING NA CLASSIFICAÇÃO DE IMAGENS 32
2.3.1 Redes Neurais Convolucionais 32
2.3.1.1 ResNet 33
2.3.2 Mask-RCNN 37
2.3.3 YOLOv5 43
3 MATERIAIS E MÉTODOS 46
3.1 DATASET: TRASHNET 46
3.1.1 Preparação da base de dados 47
3.1.2 Arranjo da base de dados para o treinamento da Mask R-CNN 48
3.1.3 Arranjo da base de dados para o treinamento da YOLOv5 49
3.2 TRANSFERÊNCIA DE APRENDIZAGEM 50
3.3 APRIMORAMENTO DOS MODELOS 50
3.4 DETERMINAÇÃO DA POSIÇÃO E ORIENTAÇÃO DOS OBJETOS 53
3.4.1 Mask RCNN 54
3.4.2 YOLOv5 57

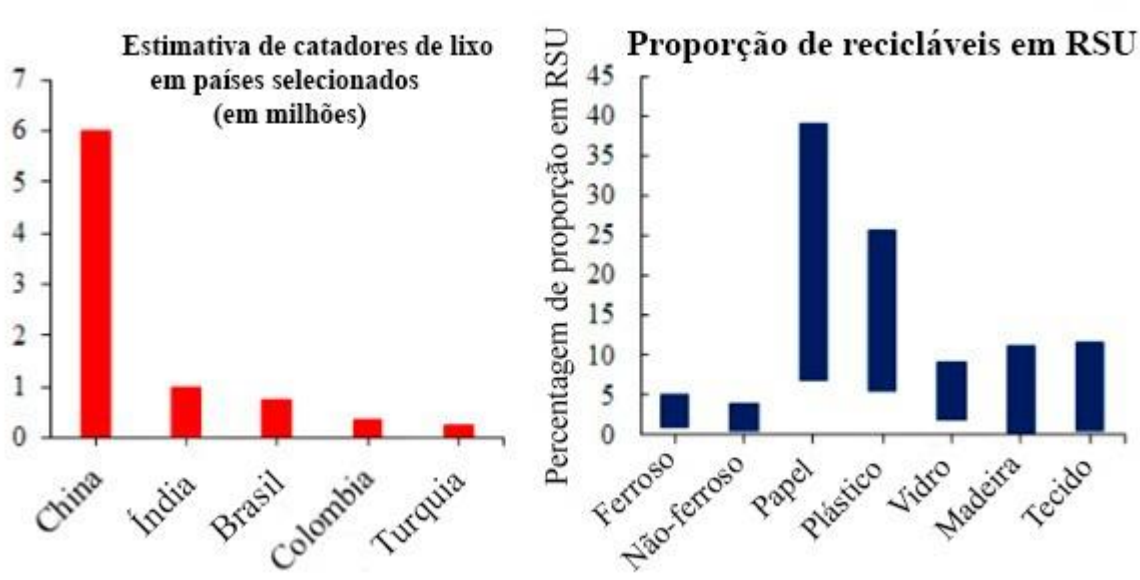
3.5 INTERFACE COM O ROBÔ 58
3.6 INTEGRAÇÃO ENTRE OS SISTEMAS 61
3.6.1 Utilização dos ângulos calculados no sistema de orientação do robô 63
4 RESULTADOS E DISCUSSÕES 66
4.1 RESULTADOS OBTIDOS UTILIZANDO A MASK-RCNN 66
4.2 RESULTADOS OBTIDOS UTILIZANDO A YOLOV5 72
4.3 COMPARAÇÃO ENTRE OS MODELOS DE VISÃO DESENVOLVIDOS 77
4.4 MOVIMENTAÇÃO DO ROBÔ UTILIZANDO PYTHON 77
4.5 RESULTADOS OBTIDOS NA INTEGRAÇÃO DOS SISTEMAS 78
5 CONCLUSÕES E TRABALHOS FUTUROS 84
5.1 TRABALHOS FUTUROS 84
REFERÊNCIAS 86

1 INTRODUÇÃO

Um dos maiores problemas do ponto de vista ambiental enfrentados pela humanidade é a quantidade de resíduos gerada. O crescimento desordenado de grandes cidades aliado ao consumismo e à industrialização vêm gerando o aumento substancial da quantidade de lixo sólido gerado (HOLLOWAY, 1989). A reciclagem é um recurso essencial para o desenvolvimento sustentável das sociedades ao passo que visa a reutilização dos resíduos e, com isso, a diminuição do acúmulo de lixo. Entretanto, os processos convencionais de separação de materiais, que acontecem de forma manual, muitas vezes são ineficientes e geram diversas complicações sociais como o aparecimento de doenças causadas pela condição de trabalho insalubre (SALIMI; BAYU DEWANTARA; WIBOWO, 2019).

A Figura 1 ilustra o *ranking* dos países que possuem a maior quantidade de catadores de recicláveis e os principais materiais coletados (PAULRAJ; HAIT; THAKUR, 2016). Observe que o Brasil é o país que possui o terceiro quantitativo de pessoas que separam esses resíduos manualmente, e que papel e plástico são os dois resíduos recicláveis mais gerados mundialmente.

Figura 1 - Países com a maior quantidade de catadores de recicláveis no mundo e proporção de materiais em todo resíduo sólido urbano gerado no mundo.



Fonte: Adaptado de Paulraj, Hait e Thakur (2016).

Observa-se que a coleta seletiva de lixo consiste basicamente na separação e triagem preliminar de resíduos recicláveis. Contudo, quando realizada de forma manual, a prática da segregação

pode não ser seguida de forma uniforme. Com isso, diversos sistemas de automação foram desenvolvidos para solucionar o problema da separação de lixo. Esses sistemas se dividem entre diretos e indiretos (GUNDUPALLI; HAIT; THAKUR, 2017).

A segregação direta de resíduos se baseia na implementação de sistemas automáticos que visam separar os materiais de acordo com suas propriedades físico/químicas como a suscetibilidade magnética, condutividade elétrica e densidade. A segregação indireta, por outro lado, emprega sensores para detectar a presença e, frequentemente, a localização de materiais recicláveis no lixo, para que máquinas ou robôs automatizados possam ser empregados para classificar os materiais recicláveis detectados (GUNDUPALLI; HAIT; THAKUR, 2017).

Neste trabalho, a solução proposta para o problema levantado é utilizar uma segregação indireta de resíduos utilizando um robô colaborativo e uma câmera RGB. A função desta última é completada com a implementação de um sistema de inteligência artificial, que processa as imagens geradas pela câmera, tendo como saída a localização e o tipo de resíduo.

A função do robô colaborativo é a de manipulação do resíduo. A escolha deste tipo de manipulador possui a vantagem de resultar em uma solução que pode ser implementada aos poucos e sem grandes modificações na planta, uma vez que um robô colaborativo pode coabitar o mesmo ambiente que os separadores manuais substituindo os mesmos pouco a pouco.

A utilização de métodos computacionais baseados em inteligência artificial como redes de aprendizado profundo (*Deep Learning)*, têm a capacidade de tornar mais eficiente o processo de separação de resíduos. Além dos benefícios produtivos, a separação automática de lixo utilizando recursos como robôs manipuladores colaborativos tem a capacidade de proporcionar diversos benefícios econômicos e sociais (VO et al., 2019).

Algoritmos utilizando visão computacional clássica, que utilizam-se de binarização ou determinação de bordas, podem ser utilizados para solucionar o problema da separação de resíduos. Porém, com o avanço das redes neurais de aprendizado profundo e com a disseminação de ferramentas que simplificam a implementação, treinamento e utilização desses algoritmos, sua utilização tem maior relevância em problemas de classificação já que fornecem uma precisão significativamente superior. Algumas redes de detecção e segmentação de objetos adequadas para esta tarefa, como YoloV5 (JOCHER, 2020) e Mask-RCNN (HE et al.,

2020), estão disponíveis como bibliotecas de código aberto, geralmente implementadas e contidas em modelos pré-treinados. (VUOLA; AKRAM; KANNALA, 2019).

Os robôs hoje são um fator fundamental para garantir a competitividade na indústria. Segundo estimativas da Federação Internacional de Robótica, existem atualmente cerca de 3 milhões de robôs industriais em todo o mundo. Além disso, verificou-se um aumento exponencial da quantidade de robôs colaborativos principalmente em pequenas e médias empresas (WORLD ROBOTICS, 2021).

A expansão do uso dos robôs colaborativos, também chamados de cobots, se dá por permitirem a interação direta entre os humanos e robôs, superando assim a antiga métrica que existia no chão de fábrica onde robôs deveriam ser alocados em gaiolas de segurança e trabalhavam isolados dos humanos. A possibilidade da interação trabalhador/robô permite o aumento da produtividade e a diminuição dos riscos e desgaste aos quais os funcionários estão expostos (VILLANI et al., 2018).

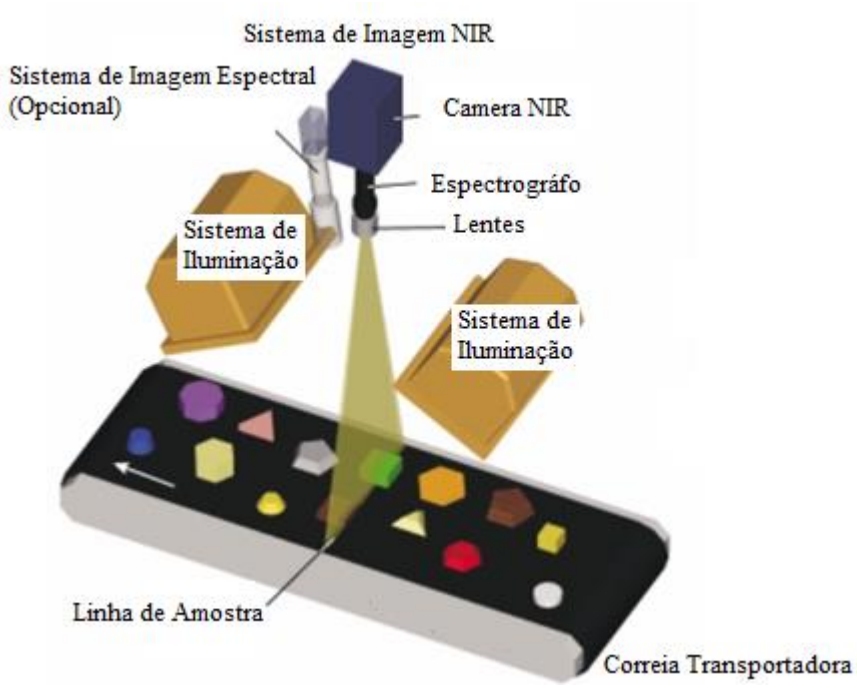
A utilização da robótica colaborativa aliada às técnicas de visão computacional e inteligência artificial demonstra um potencial enorme na resolução de diversos problemas da indústria incluindo problema da separação automática de resíduos.

1.1 TRABALHOS CORRELATOS

Existe uma diversidade de trabalhos relacionados ao tema dessa pesquisa, o que demostra a importância do assunto.

Em 2003, Kulcke et al., desenvolveram um sistema para realizar a separação de polímeros sintéticos. A Figura 2 ilustra a arquitetura básica do trabalho desenvolvido, onde se observa que o sistema é baseado na tecnologia *Near Infrared* (NIR), que consiste basicamente na utilização de uma câmera capaz de capturar as características espectrométricas dos materiais submetidos à ela. Formou-se um banco de imagens com as características espectrométricas de cada material e foram desenvolvidos dois algoritmos para treinar o modelo, o *Quadratic Discriminant Classifier* (QDR), baseado em métodos estatísticos clássicos, e o *Spectral Angle Mapper* (SAM), que é um algoritmo de inferência baseado na comparação do produto interno das imagens com uma imagem de referência da classe.

Figura 2 - Arquitetura básica do trabalho de Kulcke et al.



Fonte: Adaptado de Kulcle et al.(2003).

Em 2016, Paulraj, Hait e Thakur desenvolveram um sistema de separação de lixo sólido urbano utilizando robôs móveis. O objetivo do robô é detectar o tipo de resíduo e o depositar na localidade apropriada para cada tipo de lixo. O sistema de classificação é baseado na informação proveniente de sensores de proximidade e de uma câmera NIR embarcados em um robô manipulador instalado em uma base móvel. A extração de características foi realizada através do algoritmo chamado *Speeded Up Robust Features* (SURF). A clusterização do mapa de características foi realizada utilizando o método K-Means e a classificação por meio da técnica de *Support Vector Machine* (SVM). A partir da classificação e detecção do objeto, o robô recebe as coordenadas do centróide do objeto, identificado e o algoritmo de controle do robô promove sua movimentação no sentido de capturar o objeto e depositá-lo em local apropriado. No trabalho são classificados elementos compostos de alumínio e plástico, como latas de refrigerante e garrafas plásticas.

O trabalho de Zhihong et al. (2017) tem como premissa a implementação de um sistema a partir da utilização de um robô manipulador e algoritmos baseados em *deep learning* como a *Region Proposal Network* (RPN) e a VGG16. Os algoritmos de inteligência artificial implementados enviam as informações do centro geométrico do objeto classificado e o ângulo de inclinação do objeto referente ao seu maior lado para o sistema de controle do robô que faz a manipulação

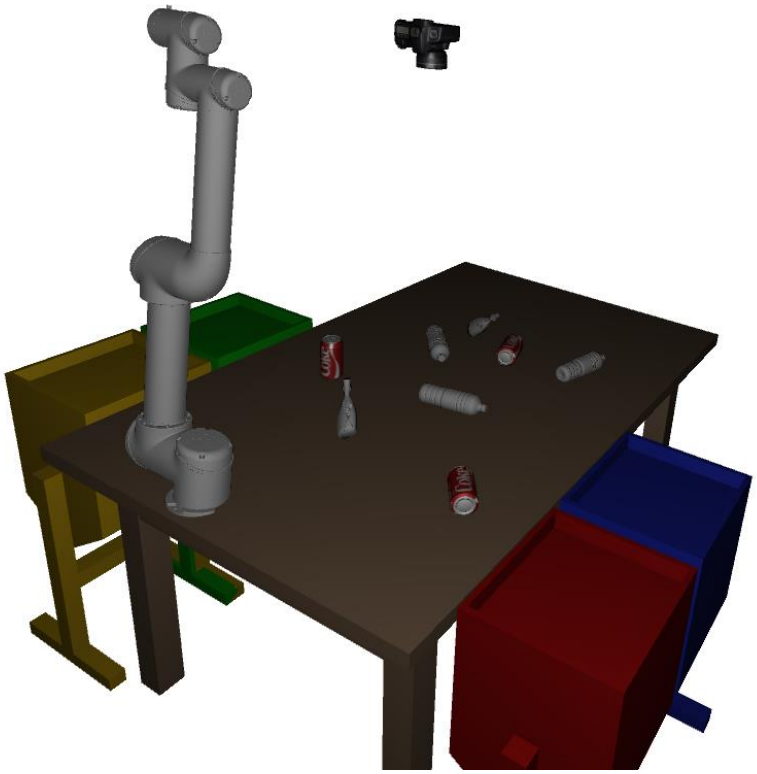
dos resíduos classificados. O trabalho citado tem como objetivo realizar apenas a classificação e a separação de garrafas plásticas.

Salimi, Bayu Dewantara e Wibowo (2019) desenvolveram um sistema de visão estéreo a fim de classificar e realizar o rastreamento de materiais em uma correia transportadora. O algoritmo desenvolvido pelos autores permite que após a extração do mapa de característica da imagem, os objetos presentes possam ser classificados como orgânicos, não orgânicos ou não resíduos. O sistema desenvolvido apresentou ao final da pesquisa 82,5% de acurácia na classificação.

A Universal Robots, empresa fabricante do robô utilizado nesse presente projeto, apresenta diversas aplicações do robô com sistemas de visão. Em um deles, o Pick IT, o robô colaborativo é utilizado para fazer a separação de produtos com condições de luminosidade adversas e com precisão excepcional. Em uma das aplicações, o robô é utilizado para alocar diversos tipos de materiais em diferentes compartimentos em uma correia transportadora (PICKIT, 2020).

Esse trabalho se distingue dos demais citados pois utiliza apenas uma câmera RGB como objeto de sensoriamento e toda a classificação e treinamento são baseados em redes de aprendizado profundo como a YoloV5 e Mask RCNN que têm como base uma rede ResNet 101 pré-treinada com o banco de dados *COCO Dataset.* Além disso, diferente dos trabalhos citados, não utiliza soluções comerciais, o robô colaborativo é controlado via comunicação *Socket IP* implementada em python, o que permite a integração do sistema inteligente também desenvolvido nessa linguagem. O controle é centralizado em um computador responsável por processar os dados, classificar as imagens, calcular a posição dos objetos e enviar a posição ao manipulador colaborativo. O sistema desenvolvido visa a implementação de um protótipo de separação de lixo, onde uma câmera RGB fixa sobre uma mesa tem a função de capturar imagens dos objetos que estão sob ela para que o robô também fixo possa realizar a separação do lixo em seu local de destino. A Figura 3 ilustra o sistema que será detalhado ao longo dessa monografia.

Figura 3 - Representação do protótipo desenvolvido



Fonte: Elaborado pelo autor (2022)

1.2 OBJETIVOS

1.2.1 Objetivo geral

Desenvolver um sistema autônomo que identifica diferentes tipos de resíduos sólidos (plástico, vidro, metal e papel) a partir de imagens de uma câmera RGB e com a utilização de um robô colaborativo separar esses diferentes tipos de resíduos.

1.2.2 Objetivos específicos

- Desenvolver os algoritmos de visão computacional capaz de identificar e classificar objetos como plástico, vidro, metal e papel;
- Avaliar a capacidade do modelo computacional desenvolvido;
- Comparar os resultados obtidos utilizando diferentes métodos;

- Implementar um algoritmo capaz de estabelecer a rotina de movimentação do robô;
- Integrar os sistemas de visão e controle do robô;
- Avaliar a eficiência da solução.

1.3 ESTRUTURA DA DISSERTAÇÃO

Os demais capítulos dessa dissertação encontram-se divididos da seguinte maneira: o Capítulo 2 descreve todo referencial teórico que foi tido como base para a elaboração desse trabalho; o Capítulo 3 descreve os materiais e métodos utilizados; no Capítulo 4 são apresentados os resultados obtidos e no Capítulo 5 são explicitadas as conclusões bem como algumas sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo traz uma abordagem teórica acerca das diferentes técnicas e estruturas utilizadas para a elaboração do trabalho. Serão abordados conceitos fundamentais de *Deep Learning* e de modelos e *frameworks* específicos como a Mask-RCNN. Além disso, é descrito como funciona o braço robótico utilizado nesse trabalho bem como são determinadas as posições e orientações dos objetos identificados pelo sistema de visão computacional.

2.1 ROBÔS COLABORATIVOS

O primeiro conceito que deve ser introduzido para o entendimento do trabalho é o conceito de robótica colaborativa. A ISSO/TS 15.066 apresenta as seguintes definições:

> Espaço de trabalho colaborativo: espaço de operação onde o sistema robótico (incluindo a peça trabalhada) e humanos podem desempenhar simultaneamente tarefas durante a produção (NORMUNG, 2016).
>
> Operação colaborativa: estado no qual um sistema robótico propositalmente projetado e um operador trabalham dentro de um espaço de trabalho colaborativo (NORMUNG, 2016).

Sendo assim, robôs colaborativos podem ser definidos como sistemas robóticos projetados para desempenhar operações colaborativas em espaços de trabalho também colaborativos.

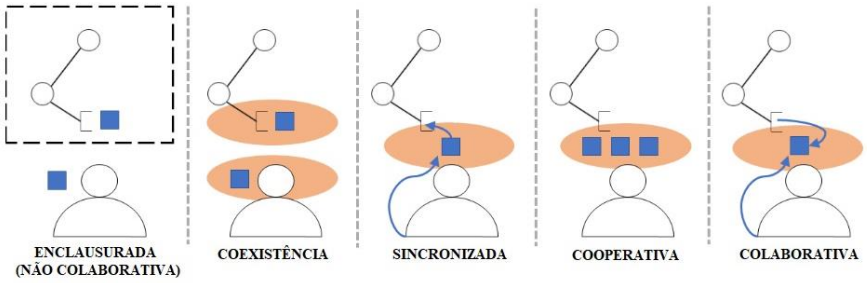
2.1.1 Utilização da robótica colaborativa na indústria

A utilização de robôs colaborativos no setor industrial pode se dar de diferentes formas. Müller, Vette e Geenen (2017) classificaram a utilização de robôs colaborativos na indústria em quatro modalidades, sendo elas:

- **Coexistência**, quando robô e humano se encontram em um mesmo ambiente, porém não interagem;
- **Sincronizada**, quando robô e humano interagem em um mesmo ambiente, porém em tempos distintos.
- **Cooperativa**, quando robô e humano coexistem em um mesmo ambiente e realizam tarefas diferentes ao mesmo tempo.
- **Colaborativa**, quando robô e humano executam a mesma tarefa ao mesmo tempo em um mesmo ambiente.

A Figura 4 ilustra as relações previamente descritas. Da esquerda para a direita pode-se observar uma representação das relações enclausurada, coexistência, sincronizada, cooperativa e colaborativa.

Figura 4 - Modos de interação entre homem e robô.



Fonte: Adaptado de Matheson et al.(2019).

Existem diversas maneiras e possibilidades onde humanos e robôs podem trabalhar de forma colaborativa. Para isso, os sistemas de automação devem ser bem planejados e desenvolvidos a fim de garantir a perfeita execução da atividade e a segurança do trabalhador. Em um processo de solda por exemplo, robô e humano coexistem no mesmo ambiente apenas enquanto o humano realiza a preparação da atividade. A partir do momento em que o robô der início ao processo de soldagem, o humano deve se posicionar em uma zona segura e sistemas de sensoriamento devem existir a fim de detectar se o trabalhador se encontra em local adequado (MÜLLER; VETTE; GEENEN, 2017).

Com o avanço da tecnologia, foi possível desenvolver robôs capazes de coexistir em um mesmo ambiente com os humanos realizando as mesmas tarefas ou tarefas complementares, que se classificam como colaborativas. A Nota Técnica nº 31/2018/CGNORDSST/SIT/MTb, expedida pelo Ministério do Trabalho do Brasil, esclarece o uso de “Robôs Colaborativos” e de robôs tradicionais em “Aplicações Colaborativas” no parque industrial brasileiro, frente aos requisitos de segurança necessários ao trabalho seguro e à luz da interpretação técnica da NR 12. Na nota técnica define-se que o uso de robôs colaborativos que trabalham juntos com os seres humanos em espaços devidamente projetados para esse fim, estão em acordância com as definições da NR12 (CNI, 2018). Com essa nova maneira de relação entre homem e robô algumas normas internacionais foram criadas a fim de reger o novo modo de trabalho. São elas:

- ISO TR15066:2016 – Especifica os requisitos para a utilização de robôs colaborativos em ambientes industriais, regulamenta as condições do espaço de trabalho e condições de operação;
- ISO 10218-1 – Requisitos de segurança para robôs industriais.

Destaca-se também que funcionalidades implementadas nesse tipo de robô promovem e possibilitam a interação com humanos respeitando limites de segurança, funções como a definição de zonas seguras, detecção de colisão, redução de velocidade e limites de força nas garras, aceleram a aceitação do trabalhador a realizar atividades em conjunto com robôs (BLOSS, 2016). El Zaatari, et al. (2019), listaram as principais aplicações no meio industrial onde robôs colaborativos são aplicados. A Tabela 1 descreve o papel de cada um dos elementos, humano e robô no contexto de algumas atividades.

Tabela 1 - Aplicações do cobot na indústria.

Cenário	Tarefa Humana	Tarefa do Cobot
Co-manipulação	Humano segura e move objetos. O humano guia o objeto em seu caminho.	O Robô suporta o peso do objeto.
Suporte	O humano realiza serviço de polimento em uma peça	O Robô segura a peça na posição ergonomicamente mais adequada para o trabalho do humano.
Transferência	O humano recebe do robô objetos e os posiciona no lugar desejado.	O robô transfere e entrega objetos ao humano.
Montagem	As ações de montagem são distribuídas entre o ser humano e o cobot de acordo com a carga de trabalho esperada e a energia consumida	Captura objetos e os posiciona no lugar desejado. *Pick and Place.*
Iluminação	O humano realiza tarefas manuais sobre bancada.	Cobot utilizando uma luminária como ferramenta direciona e iluminação para a execução da atividade humana.
Inspeção	O humano aperta parafusos em orifícios	Cobot inspeciona se todos os parafusos foram apertados e

		alerta em caso de orifícios faltantes
Perfuração	O humano especifica o local do furo	Cobot executa furo conforme especificações.
Lixamento	O humano prepara a superfície para ser lixada	Cobot executa tarefa de lixar, executando movimento paralelos a superfície de trabalho.
Parafusar	O humano insere parafusos nos orifícios	O Robô aperta os parafusos conforme especificações de torque.

Fonte: El Zaatari, et al. (2019).

Observando a Tabela 1, como pode-se depreender, a utilização de robôs colaborativos é muitas vezes condicionada à utilização conjunta de elementos sensoriais que possibilitam a melhor interação entre homem e máquina e maior amplitude de possibilidade de realização de tarefas. Pode-se citar dentre esses elementos a visão. A utilização conjunta de robôs e câmeras permite que uma grande gama de tarefas possa ser realizada e muitas vezes esses sistemas de visão computacional são baseados em redes neurais convolucionais de aprendizado profundo que permitem a classificação e segmentação de imagens com enorme precisão e assertividade.

2.2 ROBÔ UR3E

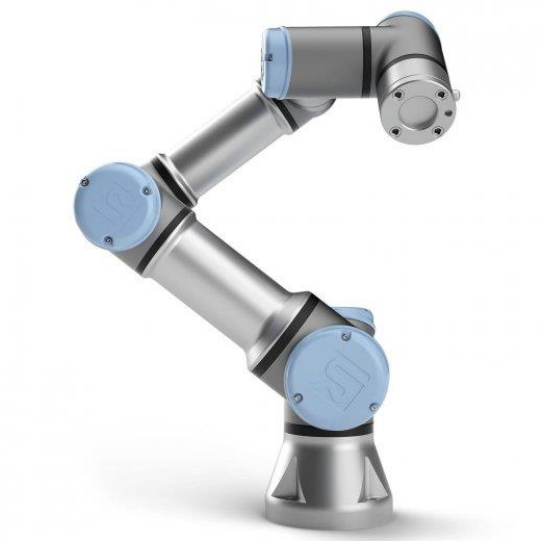
O Robô UR3e, da empresa Universal Robot, é um robô industrial colaborativo, ideal para aplicações onde a interação com humanos é necessária. Este equipamento pode ser instalado até mesmo sobre uma mesa, devido ao seu *design* ultraleve e compacto. O cobot pesa 11kg e possui uma carga útil de 3kg. O robô manipulador possui seis graus de liberdade, que possuem limitações de giro em 360º e uma última junta que não possui limitação de giro (ROBOTS, 2021).

O equipamento em questão apresenta uma enorme facilidade de programação, sendo possível criar rotinas de movimentação via terminal de programação de forma intuitiva e visual. A central de controle do cobot possui uma saída de rede que possibilita a comunicação do robô via ethernet com outros dispositivos como computadores.

O UR3 foi escolhido baseado em sua disponibilidade para uso na instituição onde esse trabalho foi desenvolvido. Além disso a escolha teve como base sua fácil integração com ambiente de

programação em *Python*, mesma linguagem utilizada para o desenvolvimento do sistema de visão computacional. Além disso a escolha do robô. A Figura 5 ilustra o modelo de robô utilizado neste trabalho.

Figura 5 - Robô UR3.

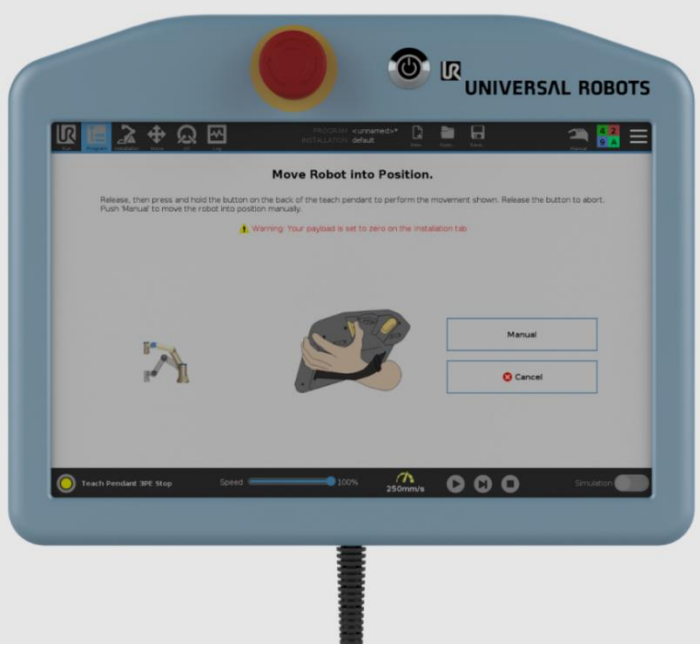


Fonte: Universal Robots (2021)

2.2.1 Operação e Possibilidades de integração do robô UR3e.

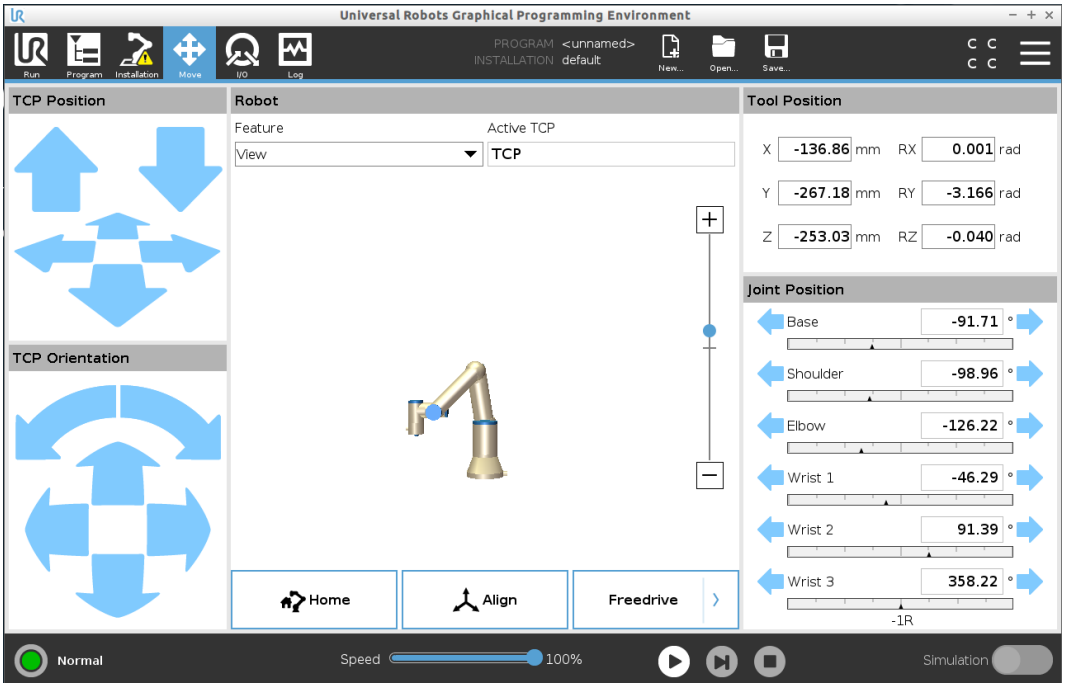
O cobot utilizado neste trabalho é reconhecido por sua facilidade de operação e integração com outros tipos de sistemas computacionais. A operação do robô se dá de forma extremamente didática e intuitiva por meio de um terminal de programação (TP) que funciona como um dispositivo de interface homem máquina (IHM). Este terminal é fornecido junto ao robô e é conectado diretamente à sua unidade de processamento. Nesse dispositivo é possível monitorar os valores de posição e orientação atuais do robô bem como programar rotinas de trabalho. A Figura 6 ilustra a IHM do UR3e e na Figura 7 pode-se observar a tela de informações de posição e orientação do robô.

Figura 6 - Interface de operação e programação do cobot UR3e.



Fonte: Universal Robots (2021).

Figura 7 - Interface gráfica de operação e programação do robô.

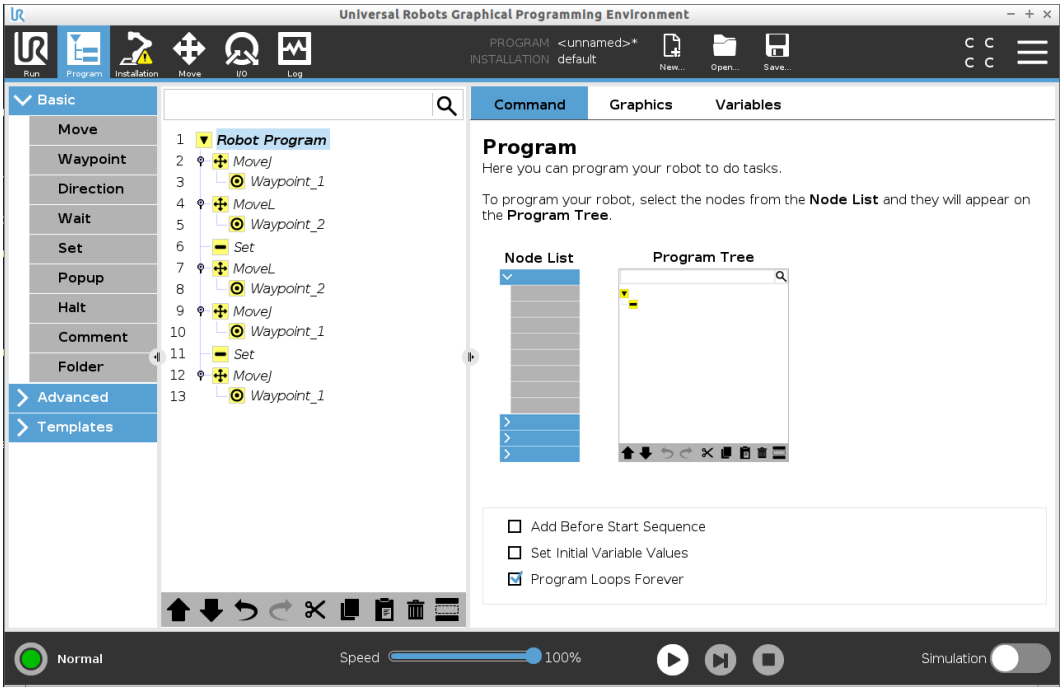


Fonte: Elaborado pelo Autor (2022).

A programação do robô UR3 é baseado no envio de instruções sequenciais. Pode-se programar o robô de maneiras distintas, utilizando-se das funções pré-definidas pelo sistema ou através da

criação de scripts. A Figura 8 ilustra um exemplo de estrutura sequencial de programação onde o robô realiza uma atividade de *pick and place*.

Figura 8 - Exemplo de programação sequencial de movimentação do robô.

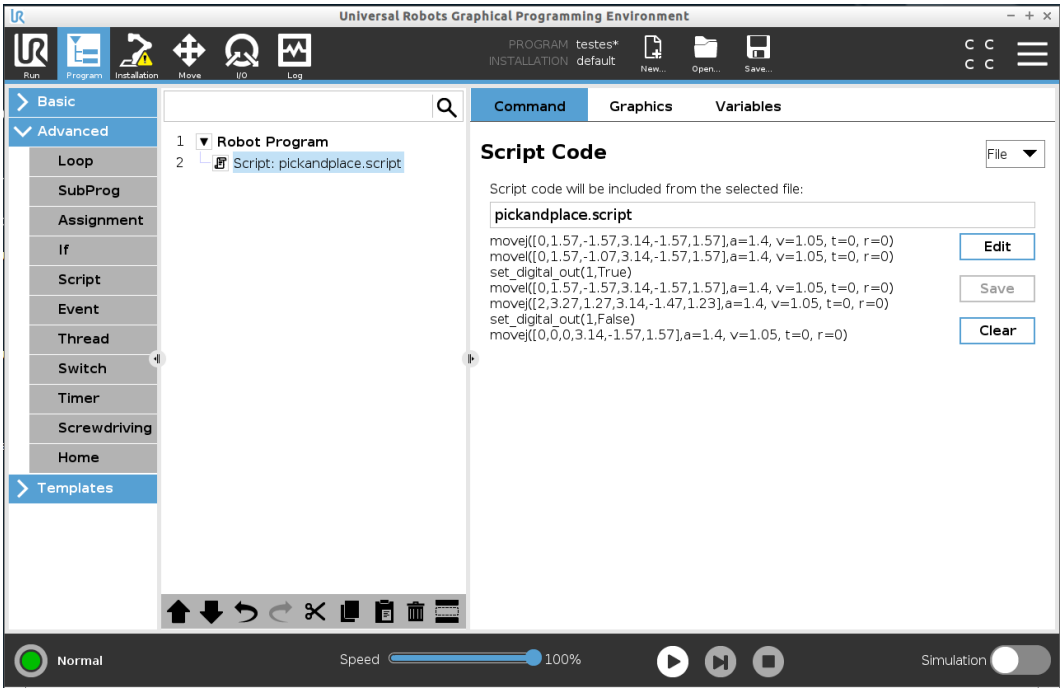


Fonte: Elaborado pelo Autor (2022).

Pode-se observar na imagem que basicamente são fornecidas orientações de movimentação para o robô através das funções *Movej* e *Movel*. A principal diferença entre as funções é que na Movej o robô percorre a trajetória otimizada evitando singularidades para chegar até a posição final (*waypoint*) definida pelo usuário, já na *Movel* o robô irá deslocar a ferramenta de forma linear até o ponto destino definido. A *waypoint* é caracterizada como um vetor de 6 posições sendo as três primeiras as coordenadas x,y e z da posição desejada no espaço e as 3 últimas os valores de orientação rx, ry e rz. A função set é normalmente utilizada para alterar os valores lógicos das saídas digitais do controlador do robô. Normalmente ferramentas, como garras, são ligadas as saídas digitais do robô e, portanto, utiliza-se a função *set* para abrir ou fechar a garra.

Além da possibilidade de programação do robô de forma visual e intuitiva conforme exibido na Figura 8, pode-se programar o UR3e através do uso de scripts. A Figura 9 ilustra um programa implementado utilizando-se script que executa exatamente a mesma função do ilustrado na Figura 8.

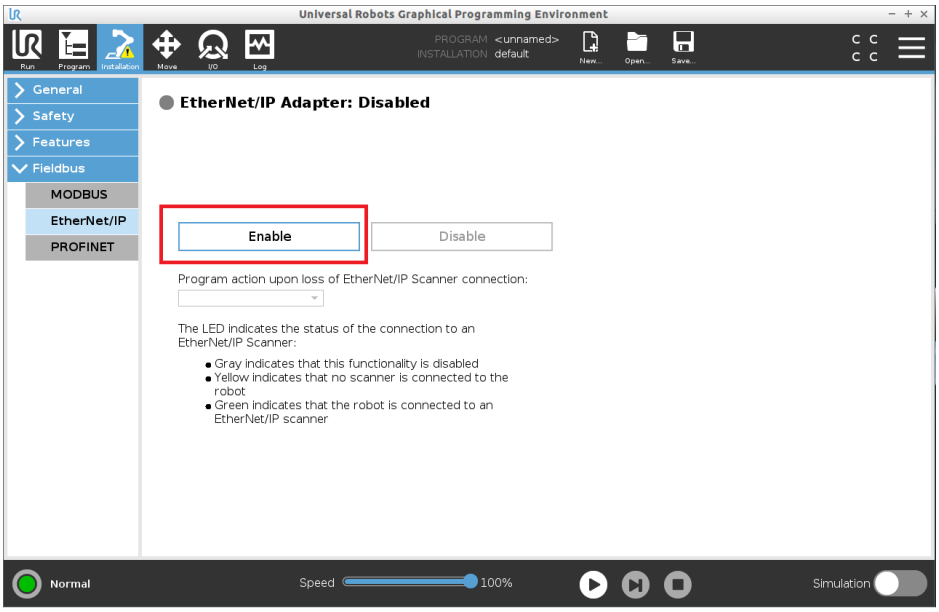
Figura 9 - Estrutura do programa de *pick and place* utilizando-se script.



Fonte: Elaborado pelo autor (2022).

O cobot UR3e permite também que instruções de script sejam enviadas sequencialmente através de estrutura de comunicação *socket IP*. Para que a comunicação Ethernet IP seja estabelecidao driver de comunicação deve ser habilitado. A Figura 10 ilustra a tela de configuração do driver.

Figura 10 - Tela de configurações do UR3e e botão responsável por habilitar a comunicação em Ethernet/IP



Fonte: Elaborado pelo autor (2022).

As informações de movimentação e monitoração de parâmetros são enviadas ao robô no mesmo formato das que são escritas e lidas na própria IHM do robô, o processador do cobot lê, interpreta e executa as ações.

Neste trabalho, utiliza-se o recurso descrito para ler e enviar comandos ao robô, utilizando uma biblioteca própria desenvolvida em *python*. Esta é capaz de estabelecer uma comunicação *socket TCP/IP* entre o computador e o robô e enviar comandos de movimentação e leitura de informações do sensoriamento do manipulador. As informações detalhadas sobre o método de controle remoto do robô serão descritas na Seção 3.5.

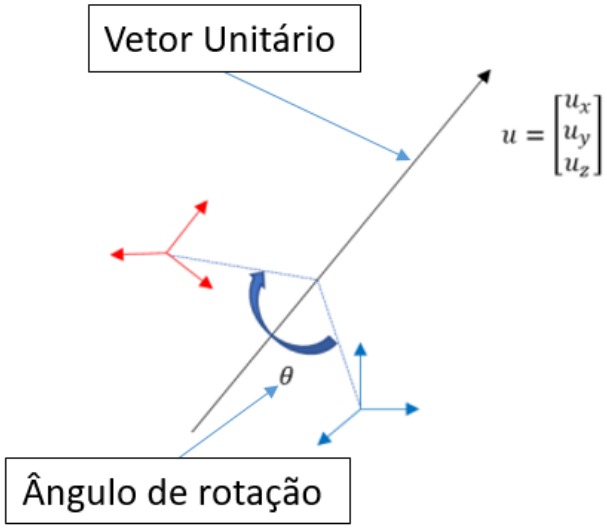
2.2.2 Sistema de Coordenadas e Orientação do robô UR3e

Como todo robô manipulador, o robô UR3e utiliza de referenciais para definir poses (posição e orientação) no espaço de trabalho. Neste robô, definindo-se um sistema de coordenadas cartesianas na base do robô (referencial de base) e uma sistemas de coordenadas cartesianas no centro da ferramenta (referencial da ferramenta), tem-se que a posição da ferramenta será dada, em relação ao referencial da base, pelas coordenadas X, Y e Z da origem do referencial da ferramenta. Já a orientação da ferramenta, no caso do robô UR3e, será dada utilizando-se a representação de orientação vetor-ângulo.

O teorema da rotação de corpos rígidos no espaço de Euler define o conceito de representação de orientação vetor-ângulo, ou vetor de rotação. A Figura 11 ilustra a representação vetor-ângulo de Euler. Segundo Euler, a orientação de qualquer corpo rígido no espaço pode ser representada pela rotação de um vetor unitário, que tem como origem um eixo fixo, em torno de si próprio (PESCE, 2004).

Esse tipo de representação de orientação é utilizado em sistemas robóticos para a indicação de orientação de TCP (*Tool Center Point*) para contornar problemas numéricos encontrados, por exemplo, quando utiliza-se matriz de rotação para a determinação de orientação onde em alguns pontos tem-se configurações singulares.

Figura 11 - Representação do sistema de orientação vetor-angulo. O eixo em azul representa a orientação de um corpo antes da rotação desse mesmo eixo em θ graus em torno do vetor u. O eixo vermelho ilustra a orientação final.



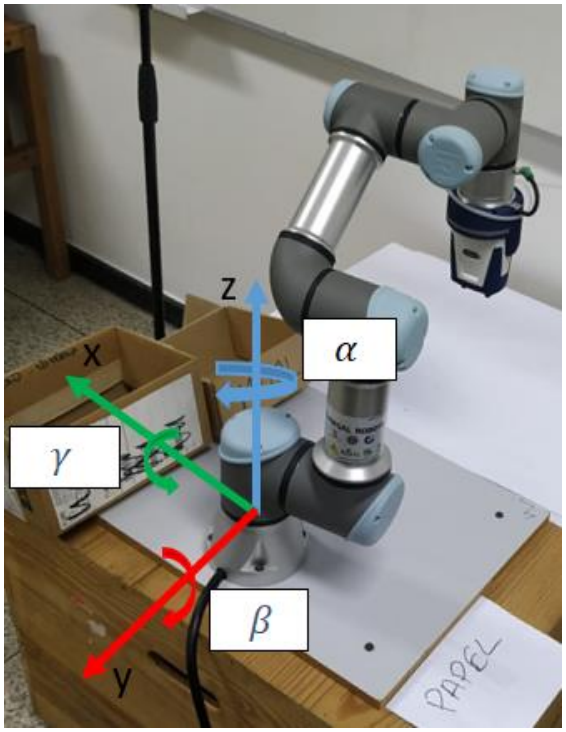
Fonte: Adaptado de (LAVALLE, 2006).

No caso do UR3e, definindo-se como eixo de referência a base do robô, a orientação da ferramenta é definida com apenas três parâmetros, rx, ry e rz, ou seja, o vetor de rotação é calculado a partir das componentes espaciais x, y e z do vetor unitário u e de sua rotação θ:

$$r = \begin{bmatrix} rx \\ ry \\ rz \end{bmatrix} = \begin{bmatrix} \theta u_x \\ \theta u_y \\ \theta u_z \end{bmatrix} \quad (1)$$

Apesar de contornar problemas numéricos, o método do vetor de rotação é de certa forma mais abstrato, já que é mais complexo correlacionar a disposição física de elementos no espaço com os valores numéricos dos vetores de rotação. Sendo assim, a fim de tornar o cálculo do ângulo de rotação do robô menos abstrato, utiliza-se um sistema de orientação baseado em matriz de rotação. Esse sistema de orientação é baseado na definição de ângulos de rotação em torno dos eixos cartesianos do robô z, y e x, respectivamente α, β, e γ, tendo como origem a base do cobot. A Figura 12 ilustra a definição dos ângulos α, β, γ em um manipulador robótico.

Figura 12 - Representação das rotações em torno dos eixos no UR3e.



Fonte: Elaborado pelo autor (2022).

Pode-se observar que uma rotação de um ângulo α com relação ao eixo z da base do robô corresponde a uma rotação dos eixos X e Y do TCP. Essa rotação pode ser representada pela seguinte matriz:

$$R_z(\alpha) = \begin{bmatrix} cos\alpha & -sen\alpha & 0 \\ sen\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Analogamente para os eixos x e y tem-se as matrizes:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sen\gamma \\ 0 & sen\gamma & cos\gamma \end{bmatrix} \quad (3)$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sen\beta \\ 0 & 1 & 0 \\ -sen\beta & 0 & cos\beta \end{bmatrix} \quad (4)$$

A matriz de rotação é dada pela multiplicação das matrizes 3D de cada eixo. Sendo assim:

$$R\ (\alpha, \beta, \gamma) = \ R_z(\alpha) * R_y(\beta) * R_x(\gamma) \tag{5}$$

Chegando-se à:

$$R\ (\alpha, \beta, \gamma) = \begin{bmatrix} cos\alpha.cos\beta & cos\alpha.sen\beta.sen\gamma - sen\alpha.cos\gamma & cos\alpha.sen\beta.cos\gamma + sen\alpha.sen\gamma \\ sen\alpha.cos\beta & sen\alpha.sen\beta.sen + cos\alpha.cos\gamma & sen\alpha.sen\beta.cos\gamma - cos\alpha.sen\gamma \\ -sen\beta & cos\beta.sen\gamma & cos\beta.cos\gamma \end{bmatrix} \tag{6}$$

As três componentes de orientação rx, ry e rz da ferramenta do robô UR3e podem ser definidas a partir da sua matriz de rotação $R\ (\alpha, \beta, \gamma)$ da seguinte forma:

$$\theta = \ \cos^{-1}\left(\frac{R_{11} + R_{22} + R_{33} - 1}{2}\right) \tag{7}$$

$$w = \frac{1}{2sen\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - \ R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{8}$$

$$rx = w_1 = \ \frac{1}{2sen\theta}(R_{32} - R_{23}) * \theta \tag{9}$$

$$ry = w_2 = \ \frac{1}{2sen\theta}(R_{13} - \ R_{31}) * \theta \tag{10}$$

$$rz = w_3 = \ \frac{1}{2sen\theta}(R_{21} - R_{12}) * \theta \tag{11}$$

Dessa forma tem-se a relação entre os inputs que determinam a orientação do TCP do robô rx, ry e rz, com os valores dos ângulos de rotação nos eixos de cada eixo do UR3 (LAVALLE, 2006).

2.3 DEEP LEARNING NA CLASSIFICAÇÃO DE IMAGENS

A classificação de imagens pode ser definida como a tarefa de categorizar imagens em categorias predefinidas ou classes, um problema recorrente em processamento de imagens. O processo de classificação pode ser considerado a base de outras diversas técnicas em visão computacional como detecção, localização e segmentação (CIRES et al., 2003).

Para os seres humanos, classificar ou identificar objetos é uma tarefa intrínseca de sua formação como ser pensante, onde desde bebê, o cérebro aprende por meio das imagens capturadas pela visão e pelos outros sentidos que a imagem de uma cadeira, por exemplo, representa uma cadeira. Para máquinas, por sua vez, a tarefa de classificar e identificar objetos pode ser um processo extremamente complexo. Algumas das complicações encontradas por sistemas de visão computacional no que diz respeito à caracterização de objetos é a diversidade de ângulos em que um objeto pode ser fotografado ou a variabilidade de uma classe, onde objetos variados podem representar a mesma categoria. Uma solução para os problemas descritos foi a implementação de meios computacionais, conhecidos como descritores de recursos (*features descriptors)* capazes de realizar a extração do mapa de características de uma imagem por meio de diversas operações envolvendo a convolução da imagem de entrada em redes neurais convolucionais de aprendizado profundo (RAWAT; WANG, 2017).

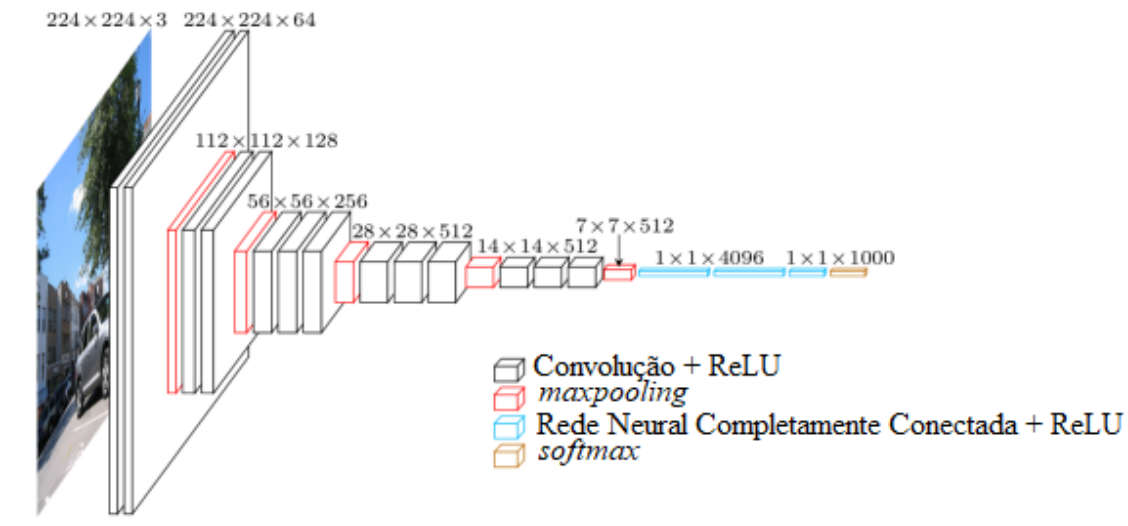
2.3.1 Redes Neurais Convolucionais

As redes neurais convolucionais surgiram a partir da necessidade de se processar dados que possuem sua estrutura formada por múltiplos vetores ou matrizes, como é o caso de uma imagem no espectro RGB. Uma imagem 2D em RGB é composta por 3 matrizes, uma para cada canal de cores, contendo a intensidade de cada pixel posicional. Outros exemplos de sinais compostos por mais de um canal seriam espectrogramas de áudio e vídeos (LECUN; BENGIO; HINTON, 2015).

Segundo Lecun, Bengio e Hinton (2015), redes neurais convolucionais possuem uma estrutura típica e são formadas por camadas. As duas primeiras camadas da rede, são geralmente identificadas como camadas de convolução e de *pooling*. A convolução de um sinal gera o que se denomina como mapa de características (*feature map)*: cada mapa gerado é submetido a uma rede neural com propagação inversa e ao final o resultado da soma de cada pixel de cada mapa de características pela matriz de pesos é submetido a uma camada de ativação geralmente

composta por uma função não linear como a ReLU que gera a resposta da rede neural convolucional. A Figura 13 ilustra uma arquitetura padrão de rede neural convolucional.

Figura 13 - Estrutura padrão de uma rede neural convolucional.



Fonte:Adaptado de Smeda (2019)

A utilização de camadas convolucionais e geração de mapas de características proporciona que características padrões de um determinado tipo de objeto sejam sempre extraídas o que permite que uma rede neural convolucional classifique diferentes tipos de objetos dentro de um universo mesmo que haja grande variabilidade entre eles. Além disso essa técnica possibilita que o ângulo o qual a foto foi tirada ou a posição do objeto na imagem não tenha influência sob a classificação (LECUN; BENGIO; HINTON, 2015).

Com o tempo e o desenvolvimento desse método computacional, diversas arquiteturas surgiram dentre elas a ResNet, rede utilizada na composição do *framework MASK RCNN* utilizado no desenvolvimento desse trabalho e explicada a seguir.

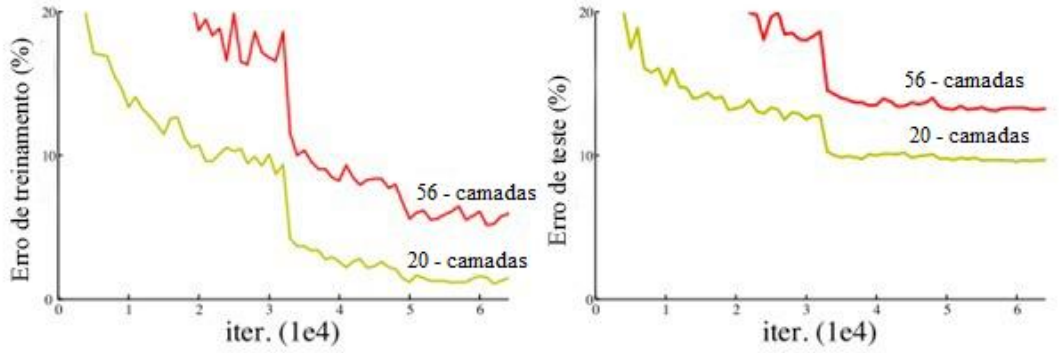
2.3.1.1 ResNet

O teorema da aproximação universal afirma que uma rede neural que se utiliza de aprendizado reverso (*Backpropagation*) em seu treinamento e que possui apenas uma camada interna de neurônios é capaz de representar qualquer função matemática (WINKLER; LE, 2017). Com a evolução da teoria computacional, desde a invenção da rede *AlexNet*, até então estado da arte das redes neurais convolucionais, o número de camadas de redes famosas, como a *AlexNet e GoogleNet22*, segue um padrão de crescimento. Enquanto em 2015 a rede *AlexNet* tinha apenas

cinco camadas, a rede VGG criada posteriormente possuía dezenove camadas e, a *GoogleNet*, 22.

No entanto, o número de camadas não representa necessariamente a eficiência de uma rede neural, ou seja, não basta apenas acumular camadas para obter-se uma rede mais eficiente. Com o acúmulo de camadas, o treinamento utilizando *backpropagation* que promove diversas multiplicações com camadas mais superficiais da rede faz com que o gradiente tenda a valores infinitesimais que acabam saturando a rede em seu processo de treinamento (HE et al., 2016). A Figura 14, do trabalho de Feng (2017) ilustra um exemplo de treinamento entre duas redes com o número de camadas convolucionais distintos. Nela pode-se observar que o aumento da quantidade de camadas não proporciona uma menor taxa de erro no treinamento da rede.

Figura 14 - Comparação dos índices de erro de treinamento e teste em redes de 20 e 56 camadas.



Fonte: Adaptado de Feng (2017).

A rede neural convolucional, conhecida atualmente como ResNet, tem sua etimologia baseada na expressão Redes Residuais (*Residual Networks*). Essa estrutura computacional teve seu reconhecimento mundial em 2015 após resultados expressivos aplicados à visão computacional como detecção de objetos e reconhecimento facial (FENG, 2017).

As redes neurais convolucionais residuais (*ResNets*) foram criadas no intuito de solucionar o problema da saturação do erro que faz com que redes neurais planas (redes neurais não residuais) mais profundas apresentem piores resultados quando comparados a redes mais rasas, como menos camadas ocultas.

Segundo He et al. (2016), as redes neurais residuais têm como objetivo adicionar a camadas mais profundas resultados encontrados em camadas mais rasas. Considerando que redes neurais convolucionais planas seguem a arquitetura representada na Figura 15.

Onde:

a → Entrada/saída de cada camada da rede;

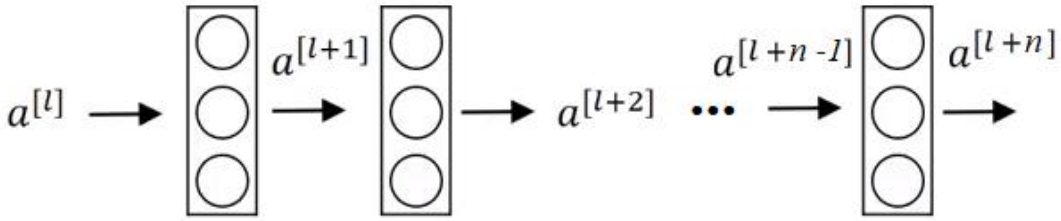
W → Matriz de pesos das ligações sinápticas da rede;

b → Bias;

l → Número da camada da rede;

$g()$ → Função não linear de ativação;

Figura 15 - Representação básica de uma rede neural plana.



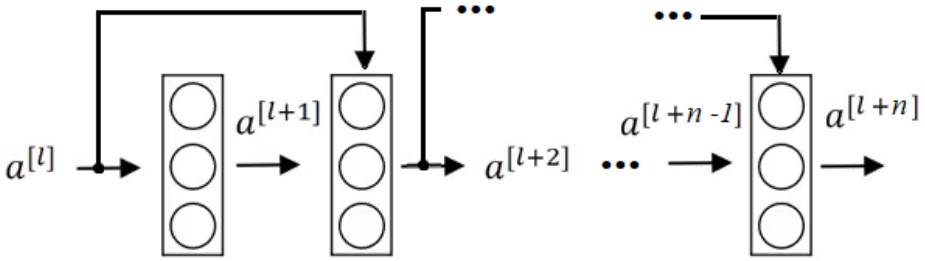
Fonte: Elaborado pelo autor (2021).

Pode-se descrever matematicamente a saída da rede com n camadas como mostrado em (12):

$$a^{[l+1]} = g(a^{l}.W^{[l+1]} + b^{[l+1]})$$
$$a^{[l+2]} = g(a^{[l+1]}.W^{[l+2]} + b^{[l+2]})$$
$$\vdots$$
$$a^{[l+n]} = g(a^{[l+n-1]}.W^{[l+n]} + b^{[l+n]}) \quad (12)$$

As ResNets têm como diferenciação das redes planas a adição de atalhos que fazem com que resultados de camadas anteriores sejam adicionados a camadas mais profundas, amenizando dessa forma o problema da saturação do gradiente, permitindo assim que mais camadas convolucionais possam ser adicionadas sem que a acurácia da rede seja prejudicada. A Figura 16 ilustra a arquitetura básica de uma ResNet. Nela cada retângulo contendo círculos representa um conjunto de neurônios de uma camada da rede, as setas representam o fluxo de treinamento e adequação dos pesos e "a" representa matematicamente a saída de cada uma das camadas.

Figura 16 - Representação básica de uma ResNet.



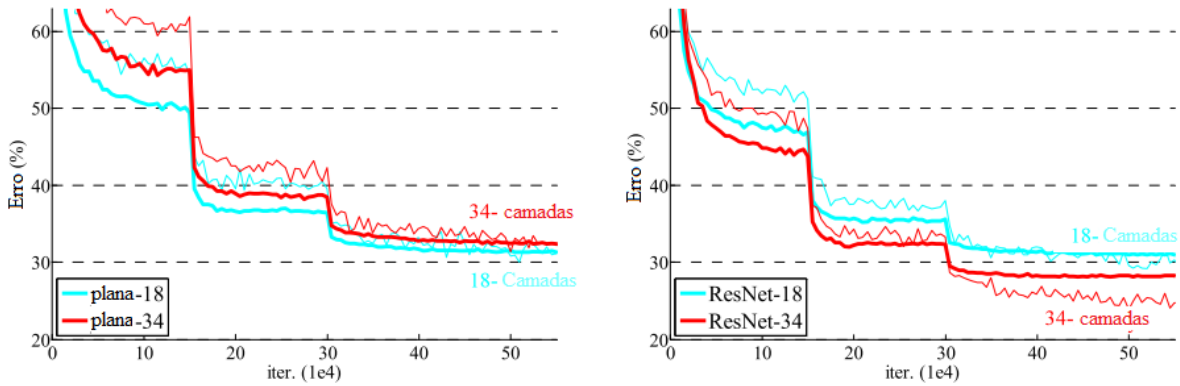
Fonte: Elaborado pelo autor (2021).

Pode-se descrever matematicamente a saída da n-ésima camada da rede como mostrado em (13):

$$a^{[l+1]} = g(a^{l}.W^{[l+1]} + b^{[l+1]})$$
$$a^{[l+2]} = g(a^{[l+1]}.W^{[l+2]} + b^{[l+2]} + a^{l})$$
$$\vdots$$
$$a^{[l+n]} = g(a^{[l+n-1]}.W^{[l+n]} + b^{[l+n]} + a^{[l+n-2]}) \quad (13)$$

Conforme os estudos realizados por He et al. (2016), a adição do termo residual conforme pode ser observado, proporciona que redes neurais convolucionais residuais atinjam camadas até oito vezes mais profundas do que redes planas sem que seja perdida acurácia. A Figura 17 ilustra os resultados encontrados comparando os dois tipos de estrutura quando aplicadas na classificação do dataset ImageNet. Nela, as curvas finas denotam erro de treinamento e curvas em negrito denotam erro de validação. Esquerda: redes planas de 18 e 34 camadas. Direita: ResNets de 18 e 34 camadas. Neste gráfico, as redes residuais possuem os mesmos parâmetros das redes planas.

Figura 17 - Treinamento utilizando o dataset ImageNet.



Fonte: Adaptado de He et al., (2016).

2.3.2 Mask-RCNN

Houve uma grande evolução na teoria computacional relacionada à segmentação semântica, grandemente impulsionada pelo desenvolvimento de *frameworks* como a Faster RCNN e as redes completamente conectadas (*Fully Conected Networks* – FCN). No entanto, a segmentação semântica é incapaz de classificar elementos em uma imagem pixel a pixel, e com isso surgiu a necessidade de que fosse desenvolvido um *framework* tão eficiente quanto os existentes para segmentação semântica, assim foi criada a Mask-RCNN (HE et al., 2017).

Segundo He et al.(2017), as redes de segmentação semântica, como a Faster RCNN, possuem como resultado do processo de classificação de imagem a identificação do objeto, ou objetos, representados e a geração de uma caixa, mais conhecida como *bouding box*, que contorna a região onde o objeto classificado se encontra. Para o completo entendimento do conceito e de como funciona a Mask RCNN, deve-se primeiro entender conceitos básicos da composição do *framework* Faster RCNN.

A Mask RCNN pode ser considerada como uma extensão da Faster RCNN, onde além de gerar rótulos das classes presentes na imagem e as *bouding boxes* nas regiões de interesse (RoI) previstas nas classificações, ela gera a classificação pixel a pixel da imagem apresentada na RoI, criando assim uma máscara sobreposta ao objeto classificado. A Figura 18 ilustra os resultados obtidos pela Mask RCNN aplicada a classificação do dataset *Comon Objects in Context* (COCO) (LIN et al., 2014). Pode-se observar nos exemplos que compõem a figura a robustez e a alta capacidade de classificação do modelo Mask RCNN aplicado à base de dados COCO na identificação dos mais variados objetos.

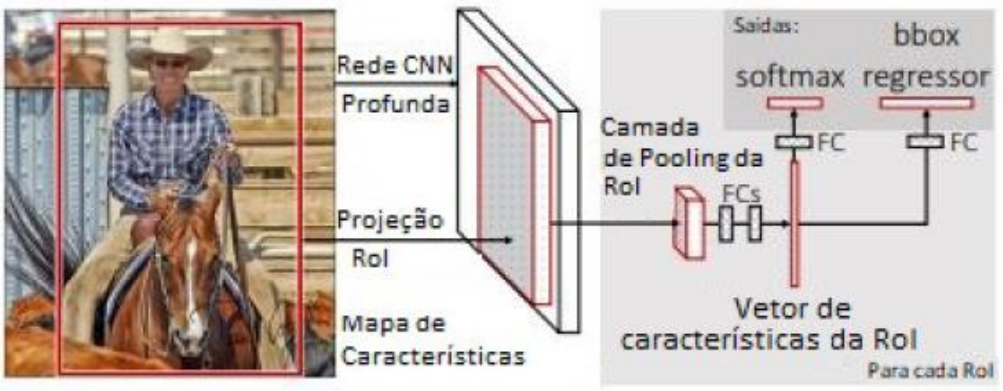
A Faster RCNN é composta por dois estágios. O primeiro estágio é conhecido como *Region Proposal Network* (RPN), que tem como função elencar *bounding boxes* de elementos candidatos a pertencerem a uma determinada classe. O segundo estágio, é responsável por realizar a extração do mapa de características e o *RoI Pooling* de cada *bounding box* determinada pelo RPN, ou seja, redimensionar cada imagem de cada caixa delimitadora para as mesmas dimensões. Esse passo é necessário tendo em vista que o próximo passo da rede, conforme pode ser observado na Figura 19, é a submissão de cada imagem segmentada nas *bounding boxes* a uma rede completamente conectada que possui uma única dimensão em cada camada. Ao final desse estágio, cada RoI é classificada a partir de seu vetor de características utilizando a função de ativação *softmax* e as dimensões das caixas delimitadoras de cada objeto,

uma vez redimensionadas no processo de pooling, são restituídas por meio de um processo de regressão (GIRSHICK, 2015).

Figura 18 - Resultados da aplicação da Mask RCNN na classificação das imagens do COCO dataset. O modelo utiliza a ResNet-101 em sua implementação.

Fonte: He et al., (2017).

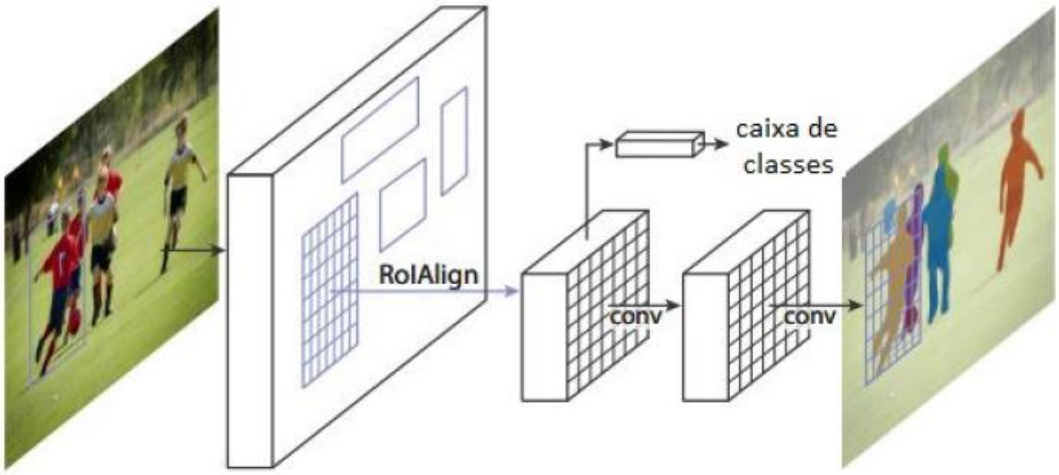
Figura 19 - Arquitetura básica da rede Faster RCNN.



Fonte: Adaptado de Girshick (2015).

A Mask RCNN adota em sua arquitetura esses dois mesmos estágios da Faster RCNN. O primeiro estágio, caracterizado pelo RPN, funciona da mesma maneira para ambos os modelos. No segundo estágio, além de serem geradas as definições de classes e *bounding boxes*, ainda é gerada uma binarização dos pixels do elemento identificado em cada caixa. Ou seja, internamente à *bounding box*, cada pixel passa por um processo de classificação. Caso o pixel seja pertencente à classe definida, ele assume valor com intensidade máxima; caso contrário, o pixel assume valor de intensidade nula. À essa imagem gerada a partir da classificação e binarização de cada pixel de todas as saídas da RPN é dada o nome de máscara. A arquitetura básica da Mask RCNN pode ser observada na Figura 20.

Figura 20 - Arquitetura básica do modelo Mask RCNN.



Fonte: Adaptado de He et al., (2017).

Segundo Khandelwal (2019), pode-se descrever o funcionamento da Mask-RCNN da seguinte forma:

a) A imagem de entrada é submetida a diversas camadas convolucionais que filtram atributos da imagem e geram o mapa de características;
b) A rede de região proposta- *Region Proposal Network* (RPN) realiza a identificação de regiões de interesse múltiplas utilizando classificador binário. A RPN utiliza nove caixas com âncoras que se deslocam pela imagem e o classificador é utilizado para determinar se naquela região encontra-se um objeto e qual é a probabilidade de que aquela região corresponda de fato a um objeto. Feito isso, utiliza-se a supressão não máxima a âncoras com alta pontuação para determinar qual âncora representa da melhor forma a RoI daquele objeto. A supressão não máxima é baseada na análise da

Intercessão sobre a União – *Interception Over Union (IoU)*. A IoU realiza uma análise da porcentagem da área da *bounding box* prevista pela rede que corresponde à sobreposição com as demais caixas de seleção previstas pela rede. Quanto maior a sobreposição, maior a chance da região selecionada ser a que representa da melhor forma a RoI de determinado objeto. Caso o valor da IoU seja inferior a 0.5, a *bounding box* é descartada. A Figura 21 ilustra o processo de seleção de *bounding* box realizada nessa etapa. Nela é possível observar o processo de refinamento da RoI utilizando a supressão não máxima. À imagem da esquerda exibe as diversas *bouding boxes* identificadas pela rede. A imagem da direita exibe o resultado do processo de supressão não máxima utilizando IoU.

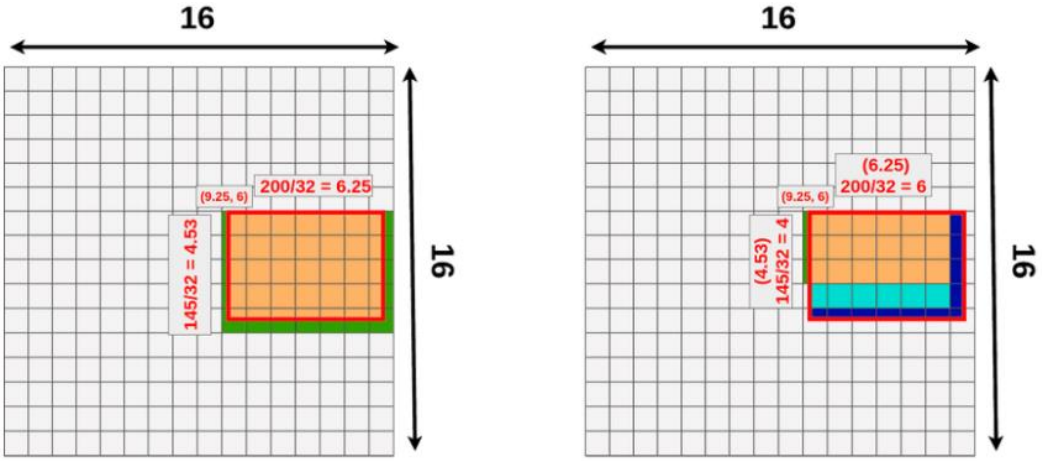
Figura 21 – Refinamento da RoI utilização o método da supressão máxima.

Fonte: Matterport (2019).

c) A próxima etapa do modelo pode ser caracterizada pela etapa de alinhamento de RoI, utilizando a *RoI Aligment Network*. Cada *bounding box* necessita ser redimensionada para que seja submetida à FCN que irá classificá-la. No caso da Faster RCNN, esse redimensionamento se dá utilizando a *RoI Pooling Network* que simplesmente divide as dimensões de cada *bounding box* pela dimensão da camada da FCN e realiza o processo de *pooling* para determinar o conteúdo de cada componente da RoI redimensionada. Nesse processo, no entanto, uma parcela dos dados reais da imagem são perdidos por conta da variabilidade do tamanho da RoI de entrada e do arredondamento das dimensões da imagem. Para sanar esse problema, visando a preservação da posição de cada pixel da imagem na Mask, é utilizada a *RoI Aligment*. Essa técnica realiza uma interpolação bilinear entre os pixels adjacentes da RoI redimensionada a fim de preservar a maior quantidade de dados da real com a melhor perspectiva. A Figura 22 ilustra a diferença entre a *RoI Pooling* e *Aligment* (ERDEM, 2020). Na figura pode-se

observar a comparação entre *RoI Aligment* (direita) e *RoI Pooling* (esquerda). A área em laranja representa a região de dados reais da RoI, regiões verdes representam dados agregados e regiões em azul, dados perdidos no processo de redimensionamento

Figura 22 – Diferença entre RoI Pooling e RoI Aligment.



Fonte: Erdem (2020).

d) Tendo realizado o processo *de RoI Aligment*, as regiões de interesse já redimensionadas são submetidas à classificação por FCN's utilizando como funções de ativação a softmax e a regressão de *bounding box* conforme acontece na Faster RCNN.
e) Os mapas de características contidos em cada RoI já ajustada no processo de *RoI Aligment* também são submetidos à classificação por duas CNN's responsáveis por classificar pixel a pixel a região do mapa de características selecionada, o que proporciona a binarização da imagem e a criação da máscara. Esse processo permite que a rede gere máscaras para cada classe sem que haja competição ou sobreposição entre classes.

Tendo determinado o modelo, necessita-se definir as métricas para avaliar a qualidade da segmentação inferida. Uma métrica comumente utilizada para a avaliação da Mask RCNN é a *Minimum Average Precision* (mAP).

Segundo Everingham et al. (2010), para que haja o entendimento do conceito de mAP, deve-se primariamente definir-se o que seria a precisão média, ou *average precision* (AP). Para que o conceito de AP seja definido, deve-se compreender o que os termos precisão e *recall* significam. A precisão pode ser definida como a quantificação de falsos positivos em uma amostra de

detecção de objetos e o recall a quantidade de falsos negativos. Ou seja, quando o valor de precisão é próximo de um tem-se uma alta taxa de acerto tendo em vista que existem pouquíssimos falsos positivos. Já quando o valor de recall se aproxima de um, tem-se que praticamente todas as imagens dentro do conjunto de dados estão sendo consideradas como positivos e, sendo assim, existe uma alta taxa de falsos negativos.

Tendo definido esses termos, AP pode ser definido como a sumarização do formato da curva de precisão/recall e é definido como a precisão média de um conjunto de onze igualmente espaçados níveis de recall. A equação em (14) demonstra o cálculo desse índice (EVERINGHAM et al., 2010).

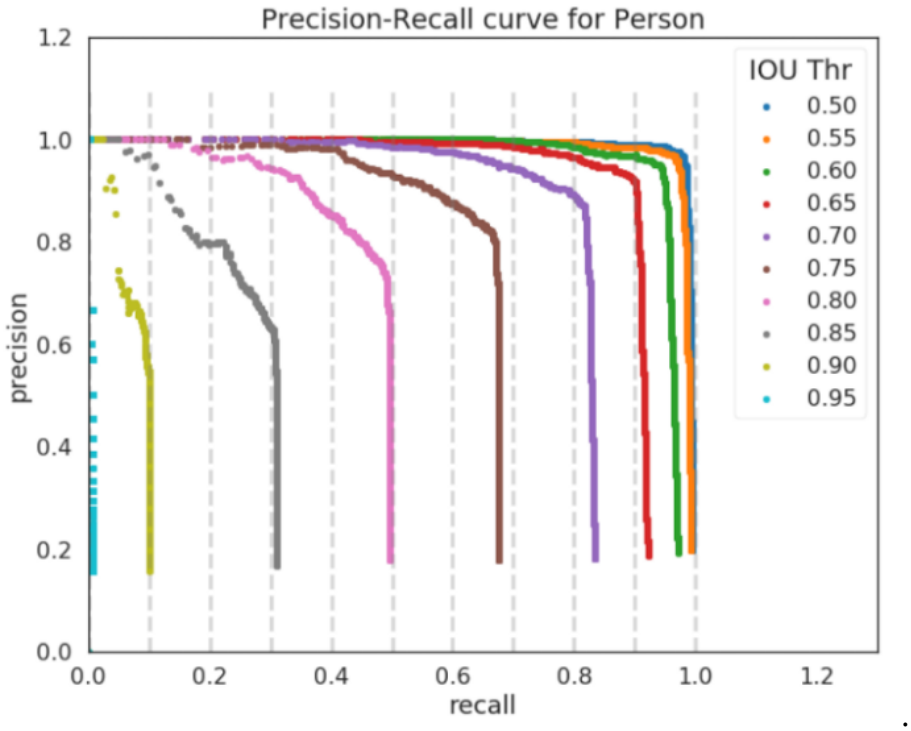
$$AP = \frac{1}{11} \sum_{Recall_i} Precisão(Recall_i) \tag{14}$$

Outro fator determinante na composição da métrica mAP é a posição detectada do objeto na imagem original, ou seja, quão assertiva é a posição da *bounding box* com relação a delimitação real do objeto. Esse índice, conforme já descrito nessa seção, é descrito pela IoU que determina a porcentagem de sobreposição da caixa delimitadora inferida pelo modelo com a *bounding box* desejada.

Sendo assim, a mAP pode ser definida como o valor médio da AP para diferentes índices de IoU. A Figura 23 exibe a representação das curvas de precisão/recall para a predição da classe pessoa utilizando a base de dados COCO para distintos valores de IoU. A média da sumarização do formato dessas curvas define a métrica utilizada nesse trabalho. Em suma, a mAP define o quão precisa é a detecção da classe e o quão precisa é o posicionamento da *bounding box* (ARLEN, 2018).

Tendo definido os métodos de classificação e posicionamento dos objetos em uma determinada imagem, faz-se necessário determinar a orientação desses objetos para que sejam capturados por manipuladores robóticos. O próximo tópico descreve outra técnica utilizada nesse trabalho para classificar e localizar os objetos na imagem.

Figura 23 - Plotagem das curvas de precisão/recall para cada indice de IoU. A média da sumarização de formato dessas curvas em cada um dos limiares de IoU compõem a mAP.



Fonte: Adaptado de Arlen (2018).

2.3.3 YOLOv5

Outra técnica utilizada nesse trabalho para a realização da classificação dos objetos nas cenas é a YOLOv5. O termo YOLO é derivado da expressão em inglês *You Only Look Once*, em português, você olha apenas uma vez. Em suma, essa técnica divide a imagem em várias imagens menores, cada "mini" imagem é analisada e os objetos identificados são classificados. Diferente da Mask RCNN, que gera uma máscara sob o alvo classificado, a YOLO apenas encontra a *bounding box* a qual o objeto pertence. Essa técnica computacional é largamente utilizada em sistemas de visão que exigem classificação de imagens em tempo real e exige um baixo custo computacional quando comparada a redes de mais antigas (GUTTA, 2021). A Figura 24 ilustra um exemplo de classificação utilizando a YOLOv5.

A primeira rede YOLO foi criada em 2015 e desde então vem evoluindo.

Figura 24 - Exemplo de detecção utilizando a rede YOLOv5 pré treinada com o COCO dataset

Fonte: Elaborado pelo autor (2021)

De acordo com Gutta (2021), a YOLOv5, utiliza-se basicamente da mesma arquitetura da YOLOv4, no entanto difere-se de sua antecessora nos seguintes quesitos:

- A YOLOv4 foi desenvolvida com base no framework *Daknet* desenvolvido em C. Já a YOLOv5 é baseada em um framework completamente desenvolvido em Python, o PyTorch.
- O arquivo que carrega as configurações da YOLOv4 tem extensão. cfg, já na YOLOv5 esse arquivo possui extensão .yaml. Esse tipo de arquivo possui maior nível de compactação o que torna a YOLOv5 uma rede mais leve e que necessita de menor espaço de armazenamento para ser executada.

A YOLOv5 possui sua estrutura baseada em três componentes principais: O tronco (*backbone)*, a cabeça (*Head)* e o pescoço (*Neck)*. Conforme os próprios nomes sugerem, essas partes são conectadas, e funcionam de fato como um organismo vivo onde o tronco sustenta o pescoço que sustenta a cabeça (RAJPUT, 2020).

O tronco ou *backbone* da YOLOv5 é basicamente a parte do modelo responsável pela extração das principais características da imagem de entrada. Esse estágio utiliza-se basicamente de redes do tipo estágio parcial cruzado (*Cross Stage Partial Networks -CSP).* Esse tipo de rede tem como objetivo particionar o mapa de características da camada base da imagem e em seguida os fundir. Essa técnica tem como propósito fazer com que o fluxo gradiente se propague

por diferentes caminhos da rede. A utilização desse tipo de estrutura como *backbone* promove uma diminuição do custo computacional durante a extração do mapa de características e com isso possibilita uma maior velocidade de inferência (WANG et al., 2019).

A camada intermediária da YOLOv5, o pescoço, é basicamente responsável por gerar as pirâmides de características. Essas estruturas são responsáveis pela detecção dos objetos de forma que sua escala não influencie no resultado. Em outros modelos como na *Faster RCNN* e na *Mask RCNN* essa estrutura também é adotada e é conhecida como *Feature Pyramid Network* (FPN) (RAJPUT, 2020).

Na cabeça da YOLOv5 é onde acontece a parte final da detecção dos objetos nas imagens. Nela são formadas as *bounding boxes* e as probabilidades de acerto de detecção. A YOLOv5 gera três tamanhos diferentes (18 × 18, 36 × 36, 72 × 72) de mapas de características para obter previsão em várias escalas, permitindo que o modelo lide com objetos pequenos, médios e grandes (XU et al., 2021).

3 MATERIAIS E MÉTODOS

Essa seção visa apresentar as principais técnicas utilizadas e os principais passos dados para que a solução do problema proposto nesse trabalho fosse alcançada.

Todo o treinamento e transferência de aprendizagem do modelo de visão computacional foi realizado utilizando a plataforma *online* gratuita *Google Colaboratory,* tendo em vista que essa ferramenta possibilita a utilização de *hardware* poderoso no processamento das imagens e treinamento do modelo. A classificação das imagens foi realizada utilizando um computador pessoal com as seguintes configurações:

- Processador Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz
- 8GB de memória RAM
- Placa de Vídeo NVídia GeForce 1050TI 6GB

Além disso, o robô manipulador utilizado no desenvolvimento desse trabalho é um equipamento da empresa *Universal Robots*: o *UR3 e-series*.

3.1 DATASET: TRASHNET

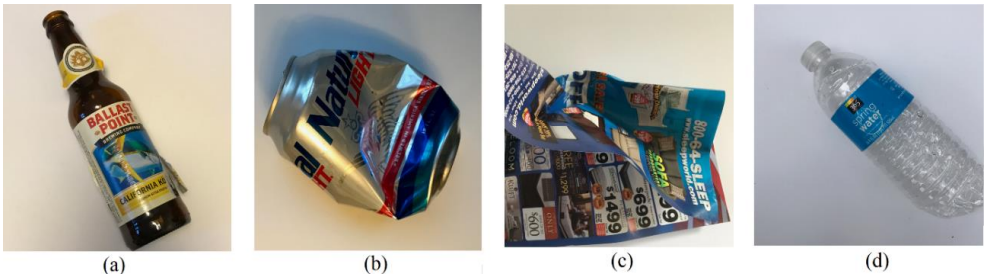
Conforme já mencionado, este trabalho tem como objetivo principal a implementação de um sistema capaz de separar diferentes tipos de lixo reciclável, utilizando um sistema de visão computacional e um robô colaborativo.

O sistema de visão computacional proposto é baseado na utilização de *frameworks* construídos utilizando redes neurais convolucionais profundas cujo método de aprendizagem é a aprendizagem profunda (*Deep Learning).* Sendo assim, para que o objetivo proposto fosse atingido, fez-se necessária a definição de uma base de dados para o treinamento da rede escolhida. Definiu-se como base de dados um *dataset* público composto por diversas imagens de diferentes tipos de lixo doméstico, o TrashNet (YANG; THUNG, 2016).

Este conjunto de dados é composto por 6 classes: vidro, metal, papel, plástico, papelão e lixo genérico. A maioria dos conjuntos de dados públicos disponíveis apresenta esses materiais no estado original e sem danos. No entanto, na verdade, esses materiais, dispostos no lixo, estão totalmente sujos, danificados e amassados. A Figura 25 ilustra um exemplo de cada classe do conjunto de dados utilizado para treinar o modelo de visão computacional. Em a) pode-se

observar uma garrafa de vidro, em b) uma lata de metal, em c) papel amassado e em d) garrafa plástica.

Figura 25 - Exemplos de imagens da base de dados TrashNet.



Fonte: Meneguelli, Cavalieri e Resende (2020).

Cada classe do conjunto de dados utilizado possui de 400 a 600 imagens em diferentes ângulos e níveis de luminosidade o que é um ponto positivo para o dataset já que esse fato promove uma variabilidade de condições para a aprendizagem da rede neural. Um ponto negativo dessa base de dados é o fato de que todas as imagens possuem fundo liso e branco, o que pode ser prejudicial ao aprendizado do modelo, tendo em vista que além das classes principais o plano de fundo (*background)* também deverá ser identificado pela rede, e a baixa variação de condições desse elemento tende a causar problemas de identificação quando a rede for validada com planos de fundo de outras colorações ou texturas.

3.1.1 Preparação da base de dados

A fim de promover o treinamento dos *frameworks* Mask RCNN e YOLOv5, o conjunto de dados TrashNet foi divido em 3 diferentes conjuntos de imagens: treino, validação e teste. Sendo assim, foi definido o volume de dados que cada conjunto deveria ter de acordo com o que geralmente se adota para treinamento de modelos de visão computacional e o que pode ser consultado no trabalho de Guyon (1997). Dessa forma, foram destinadas para o conjunto de dados de treino 350 imagens da classe vidro, 287 de metal, 415 de papel e 338 de plástico. Para o conjunto de dados de validação foram destinadas 126 imagens de vidro, 102 de metal, 148 de papel e 120 de plástico. Por último, para o conjunto de teste, foram separadas 25 imagens correspondentes à classe vidro, 20 à metal, 30 correspondentes à papel e 24 à plástico. Dessa forma, pode-se dizer que o dataset foi dividido de forma arbritária em 70% das imagens para treino, 25% para validação e 5% para testes.

3.1.2 Arranjo da base de dados para o treinamento da Mask R-CNN

Como já mencionado nesse trabalho, a Mask RCNN, *framework* utilizado no desenvolvimento, possui como saída de sua classificação três elementos: a classe propriamente dita, a caixa de delimitação, conhecida como *bouding box*, e a máscara, ou seja, a delimitação exata de onde o objeto classificado encontra-se na imagem. Dessa forma, para que esses resultados sejam gerados, o conjunto de treinamento e validação deve possuir já definido em cada imagem a representação de sua máscara e sua caixa delimitadora, além da a classe correspondente.

Sendo assim, utilizou-se o *software* gratuito e *online VGG Image Annotator* para demarcar em cada uma das imagens dos conjuntos de treinamento e validação, uma a uma, a região correspondente ao objeto mostrado na imagem, além de indicar a qual classe o objeto pertence. O resultado dessa segmentação manual é gerado em formato de arquivo de texto na extensão .json (*JavaScript Object Notation*). Esse arquivo por sua vez é utilizado no treinamento e validação do modelo da Mask RCNN para a determinação da posição dos elementos gráficos (máscara e *bounding box*) e classe correspondente. A Figura 26 ilustra como foi realizado o processo de anotação manual das imagens do dataset. Na figura, a linha azul representa o contorno dos objetos realizado no *software VGG Image Annotator*.

Figura 26 - Processo de demarcação manual da posição dos objetos na imagem utilizando o *software VGG Image Annotator*.

Fonte: Meneguelli, Cavalieri e Resende (2020).

A fim de aumentar ainda mais a variabilidade do dataset, foram aplicados alguns recursos de aumento de dados (*data augmentation*). Esse recurso tem como objetivo modificar ou inserir na imagem elementos e a partir daí gerar novas imagens com características similares à imagem original, porém com algumas modificações. Sendo assim, a aumentação de dados, recurso utilizado nesse trabalho, tem como principal objetivo expandir e variabilizar a base de dados existente a partir de manipulações nas formas, contraste, brilho e posição das imagens. Além disso, em alguns casos é possível aplicar efeitos de ruído e blur (borramento) a imagens. A

adoção dessas últimas duas técnicas, no entanto, deve ser realizada com cautela tendo em vista que pode-se modificar demais as características das imagens originais.

3.1.3 Arranjo da base de dados para o treinamento da YOLOv5

Conforme já mencionado neste trabalho, a YOLOv5, diferente da Mask R-CNN, produz como resultado da classificação apenas dois elementos: as *bounding boxes* e a classe à qual cada objeto pertence. Sendo, assim não é gerada máscara por essa rede o que torna o processo de preparação da base de dados para treinamento menos custoso quando comparado ao da Mask R-CNN, onde cada objeto necessita ser contornado manualmente por completo.

Dessa forma, para a YOLOv5, se faz necessária apenas a delimitação através de retângulos na imagem na posição onde o objeto alvo se encontra e o apontamento da classe à qual ele pertence. Para tal preparação, foi utilizado o *software* gratuito *LabelIMG*. No software, para cada uma das imagens separadas para o conjunto de dados de treinamento e validação foram traçados os retângulos delimitando a posição de cada objeto e foi indicada a classe à qual cada objeto pertence. A Figura 27 ilustra o processo de segmentação do conjunto de dados para o treinamento da YOLOv5.

Figura 27 - Processo de demarcação manual utilizando o *software LabelIMG* para o treinamento do modelo YOLOv5

Fonte: Elaborado pelo autor (2021).

Tendo realizado todos os processos com relação à base de dados, a próxima etapa do trabalho se caracteriza pelos treinamentos dos modelos.

3.2 TRANSFERÊNCIA DE APRENDIZAGEM

A transferência de aprendizagem no âmbito do aprendizado de máquina se caracteriza como a capacidade de se aproveitar um “conhecimento” já obtido por um modelo pré treinado para a aplicação na resolução de um problema distinto. Realizando analogias com situações reais, é como se o conhecimento necessário para que que se identifique uma maçã fosse utilizado para a identificação de uma pêra. Outra situação seria como se o conhecimento adquirido para que se toque teclado eletrônico fosse utilizado para aprender a tocar piano. Fica claro que, partindo de conhecimentos prévios em assuntos similares, o aprendizado para a solução de novos problemas se torna mais simples e esse é o intuito da transferência de aprendizagem (PANIGRAHI; NANDA; SWARNKAR, 2021).

Existem diversas redes de domínio público já conhecidas que possuem elevada taxa de acurácia na identificação de diversas classes. Essas redes muitas vezes são utilizadas como ponto de partida para a realização de transferência de aprendizagem para a resolução de problemas mais específicos. Os modelos utilizados nesse trabalho, a *Mask RCNN* e YOLOv5, possuem versão pública pré treinada com pesos do *Microsoft COCO Dataset (Common Objects in Context*) (LIN et al., 2014)

Essa base de dados é formada por duzentas mil imagens pré-classificadas em oitenta diferentes categorias. Sendo assim, os modelos treinados com esse *dataset* possuem uma grande gama de conhecimento absorvido o que o torna um ótimo ponto de partida para que a transferência de aprendizagem seja aplicada, já que é muito mais simples treinar a rede para classificar uma imagem de garrafa de vidro, sendo que a rede existente já identifica o que é uma garrafa, por exemplo.

Para que o treinamento fosse otimizado e o resultado do trabalho atingido, fez-se necessária a adaptação de alguns parâmetros fundamentais de treinamento durante o processo de transferência de aprendizagem. O que é discutido na seção 3.3.

3.3 APRIMORAMENTO DOS MODELOS

A fim de se atingir o objetivo proposto pelo sistema de visão computacional nesse trabalho, que é reconhecer e classificar os diferentes tipos de materiais que compõem o lixo doméstico utilizando os modelos descritos, a Mask RCNN e a YOLOv5, foram testados diferentes

conjuntos de parâmetros e configurações de treinamento até que um melhor *setup* fosse alcançado para ambas as redes.

O aprimoramento dos modelos teve como base duas métricas como referência (1) a aquisição do valor de mAP quando aplicada à detecção ao conjunto de imagens de teste e (2) a plotagem da matriz de confusão que elucida quais são as principais deficiências e acertos do modelo treinado.

Os modelos foram aperfeiçoados de forma empírica, ou seja, a partir da observação dos valores das métricas explicadas anteriormente. Foi tomado como metodologia a ação de treinar o modelo alterando um conjunto de parâmetros ou um atributo específico, observando o resultado em termos de métrica, buscando manualmente um possível ponto de máximo local encontrado. Na Tabela 2 pode-se observar os parâmetros modificados no aprimoramento dos modelos.

Tabela 2 - Parâmetros dos *frameworks (*MASK RCNN e YOLOv5) alterados a fim de obter-se o melhor resultado.

Parâmetro	Descrição básica
Dimensão das imagens de entrada	O dataset é composto por imagens com dimensões variadas e de valor muito elevado. Para que as imagens sejam utilizadas na rede neural, deve-se realizar um redimensionamento de todas as imagens de treinamento e validação a fim de equalizar a quantidade de informação de todas as imagens da base de dados. Sendo assim foram testados diferentes valores de resolução.
Taxa de aprendizado (*Learning Rate)*	A taxa de aprendizado, ou *Learning Rate*, é um hiperparâmetro do modelo que ele quantifica o quanto os pesos das camadas conectadas serão modificados em cada época, ou seja, esse parâmetro que normalmente assume valores positivos entre zero e um descreve o quão rápido a rede irá se aproximar da solução. Todavia, caso o valor adotado seja muito alto, o modelo assume valores não convergentes e acaba não atendo aos objetivos.
Número de épocas	Esse hiperparâmetro define a quantidade de vezes que o conjunto de dados de treinamento será aplicado à rede. Ou seja, esse parâmetro define a quantidade de ciclos que irão ser realizados durante o treinamento do modelo.
Tamanho de Lote (*Batch Size)*	Este hiperparâmetro define o tamanho do conjunto de amostras que será utilizado pela rede neural antes

	de que sejam atualizados os pesos das camadas conectadas.
Camadas de Treinamento	Define quais camadas da rede passarão pelo processo de treinamento. Esse parâmetro indica se apenas as camadas mais rasas (Cabeça) ou toda a rede passará pelo processo de transferência de aprendizagem.

Fonte: Próprio autor (2021).

Além da alteração dos parâmetros descritos, foram também alteradas algumas configurações de *data augmentation*, conforme descrito na Tabela 3.

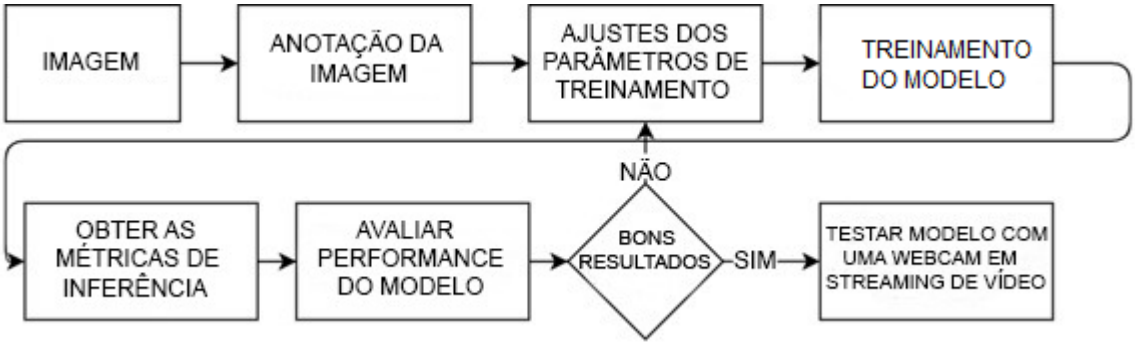
Tabela 3 - Tipos diferentes de aumentação de dados.

Configuração	Descrição básica
Giro (*Flip)*	Essa configuração gera o espelhamento das imagens da base de dados.
Cortar (*Crop)*	Esse parâmetro de aumentação de dados corta em regiões aleatórias alguns componentes do dataset de treinamento.
Normalização de Contraste (*Contrast Normalization)*	A normalização de contraste pode ser definida como a normalização do histograma de cada imagem.
Escala (*Scale)*	Realiza a modificação da escala das imagens a fim de gerar novas características nas imagens existentes na base de dados
Rotacionar (*Rotate*)	Rotaciona as imagens da base de dados em ângulos e sentidos aleatórios dentro de um limite estabelecido pelo usuário.
Borrar (*Blur*)	Esse filtro adiciona um efeito de borrado à imagem. Trata-se de um filtro passa baixas que tem como objetivo modificar as características da imagem a fim de diminuir a definição das bordas
Ruído Gaussiano (*Gaussian Noise*)	Essa configuração adiciona à imagem ruído gaussiano a fim de modificar suas características gerando mais informação à base de dados.

Fonte: Próprio Autor (2021).

A Figura 28 ilustra o diagrama de blocos que representa o processo de aprimoramento dos modelos. Existem alguns tipos de aprimoramento de modelos, como a busca em grade, a busca aleatória, a otimização Bayesiana, dentre outras. (MALATO, 2021). O aprimoramento realizado neste trabalho é caracterizado como a busca em grade, ou varredura de parâmetros, que se baseia na busca exaustiva a partir da análise de métricas pré-definidas como a mAP e matriz de confusão. O ajuste dos parâmetros segue uma sequência lógica, onde inicialmente varia-se o número de épocas e pode ser observado que o aumento do número de épocas gera um aumento da eficiência do modelo. Em seguida, são diminuídas as dimensões da imagem e observa-se novamente um aumento na eficiência do treinamento. O próximo parâmetro alterado foi a remoção de ruídos e blur nas configurações da aumentação de dados, o que gerou novamente melhores valores para as métricas definidas. Em seguida, foram modificadas as camadas de treinamento da rede e verificou-se que o treinamento com apenas as camadas de cabeça da rede se demonstrava mais efetivo. Por último, foram testados valores maiores e menores de taxa de aprendizado, e pôde-se verificar que valores próximos a 0,001 forneciam melhores valores para as métricas de treinamento.

Figura 28 - Diagrama de blocos que representa o processo de aprimoramento do modelo.



Fonte: Adaptado de Meneguelli, Cavalieri e Resende (2020).

Dessa forma, a partir da modificação dos parâmetros listados nas Tabelas 2 e 3, pôde-se obter um modelo para MASK RCNN e outro para a YOLOv5 capazes de classificar as imagens da base de dados com precisão suficiente para solucionar o problema desse trabalho.

3.4 DETERMINAÇÃO DA POSIÇÃO E ORIENTAÇÃO DOS OBJETOS

Tendo definida a classe à qual pertence a imagem submetida aos modelos, é necessário calcular a orientação do objeto identificado para que o manipulador seja capaz de capturar o objeto identificado pelo sistema de visão. Os métodos desenvolvidos para a determinação da

orientação diferem de acordo com o modelo utilizado. Nessa seção ambos os métodos serão descritos, o utilizado em conjunto com a Mask RCNN, onde a orientação é encontrada com base na máscara gerada, e com a YOLOv5, onde será gerada uma máscara do objeto para o cálculo de sua orientação.

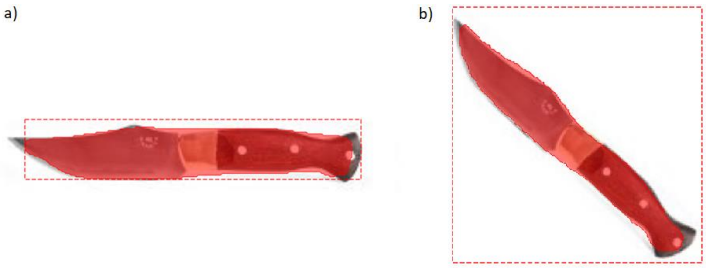
3.4.1 Mask RCNN

Conforme já mencionado, o modelo Mask RCNN fornece como informação na saída da classificação a categoria a qual o objeto pertence, a caixa delimitadora em torno do objeto classificado e a máscara binária pixel a pixel de onde se encontra o objeto na imagem.

A posição dos objetos pode ser determinada a partir das coordenadas do encontro das diagonais das *bounding boxes* geradas na classificação das imagens. Os valores obtidos dessas posições incialmente são definidos em pixels. Sendo assim faz-se necessário a adoção de um processo de conversão para transformar os valores dos pontos encontrados para milímetros para que a informação da posição dos objetos possa ser transmitida ao robô manipulador. O processo de calibração consiste basicamente em medir a posição real entre dois pontos da imagem em milímetros e realizar a equivalência dessa medida com a distância em pixels entre esses pontos. A metodologia utilizada na calibração será melhor detalhada na Seção 3.6 desta dissertação. Dessa forma, encontra-se um fator de correlação que corresponde a quantos milímetros equivalem uma quantidade específica de pixels.

A *bouding box* é sempre gerada sem orientação, ou seja, mesmo que o objeto classificado esteja rotacionado em relação ao eixo longitudinal, a caixa delimitadora sempre terá a mesma orientação conforme ilustram as Figura 29 a) e b). Em a), a faca está orientada paralela ao eixo longitudinal da imagem. Em b), a faca está orientada com ângulo com relação ao eixo longitudinal. Pode-se observar que em ambas a caixa não varia sua orientação.

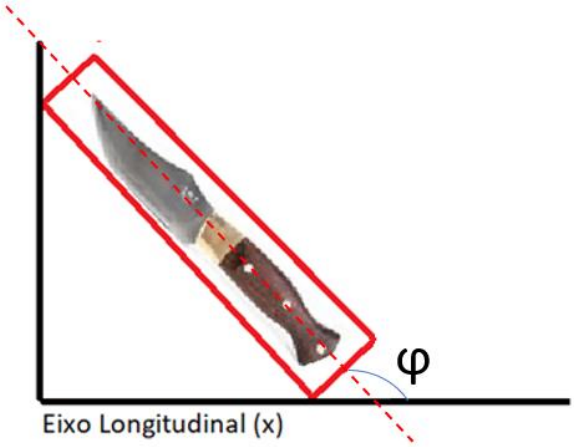
Figura 29 - Ilustração da caixa delimitadora gerada pela MaskRCNN.



Fonte: Elaborado pelo autor (2021).

Sendo assim, fez-se necessário implementar um algoritmo com o objetivo de reconstruir a *bounding box* gerada pela Mask RCNN de forma que a caixa assuma uma forma análoga a da Figura 30.

Figura 30 - Caixa delimitadora a ser gerada pelo algoritmo.

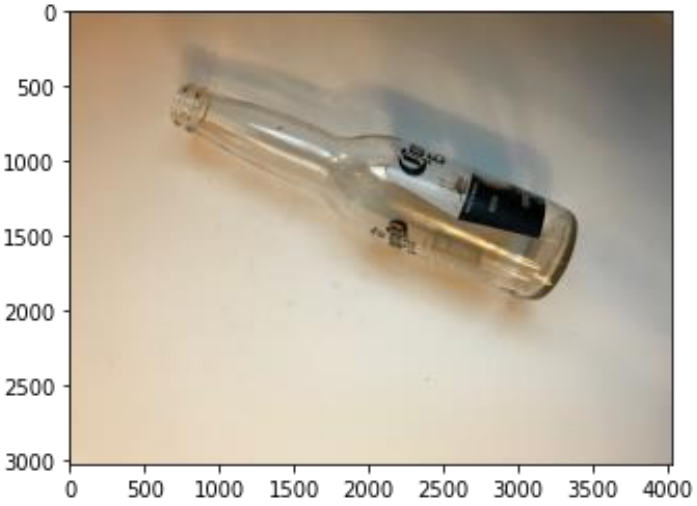


Fonte: Elaborado pelo autor (2021).

A determinação do ângulo φ, conforme exibido na Figura 30, é necessária para determinar a correta orientação da garra do manipulador que permite a captura do objeto. Para a determinação desse parâmetro adotou-se a metodologia elucidada a seguir.

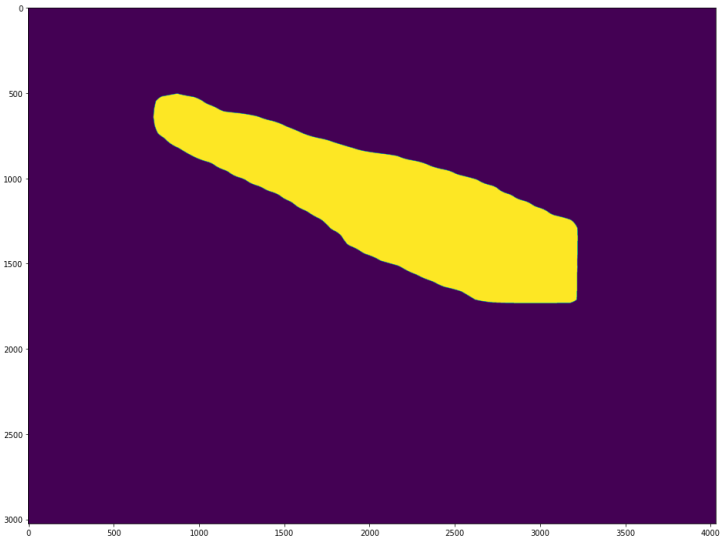
Incialmente foi obtida da classificação utilizando a Mask RCNN a máscara da imagem, ilustrada na Figura 31, conforme pode-se observar na Figura 32.

Figura 31- Imagem de garrafa retirada do *dataset* TrashNet.



Fonte: Elaborado pelo Autor (2021)

Figura 32 - Máscara gerada pela classificação utilizando Mask-RCNN.

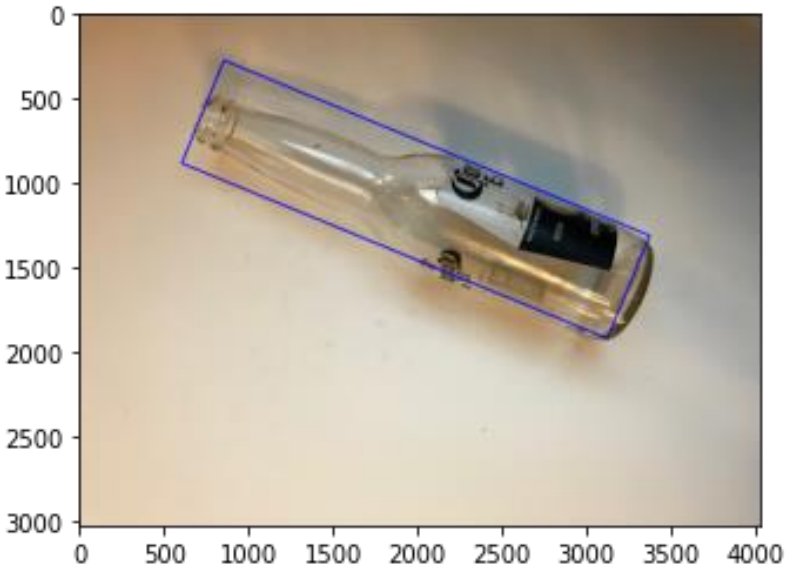


Fonte: Elaborado pelo autor (2021).

A partir da obtenção da máscara são obtidos os contornos da imagem através do método *findContours* da biblioteca OpenCV. Esse método encontra contornos em imagens binarizadas como é o caso da máscara gerada, e armazena os pontos relativos às fronteiras em um vetor.

Tendo encontrado as informações de contorno da imagem do objeto binarizado, é utilizado o método *minAreaRect*, também da biblioteca OpenCV. Esse método encontra o retângulo com menor área que compreende todos os pontos de contorno armazenados no vetor pela função *findContours*. Por fim, a função implementada utilizando os recursos já citados, calcula as coordenadas x e y do vértice superior esquerdo do retângulo, sua largura e altura e o ângulo de rotação do retângulo com relação ao eixo horizontal da imagem. Sendo assim, com a aplicação desse algorítmo, pôde-se obter o seguinte resultado, ilustrado na Figura 33. No exemplo da Figura 33 foi calculado como menor ângulo de inclinação com relação ao eixo longitudinal 168.7º.

Figura 33 - Utilização dos métodos *findContours e minAreaRect* da biblioteca OpenCV. Com a aplicação dos métodos, é possível resolver o problema da detecção da orientação dos objetos classificados.



Fonte: Elaborado pelo autor (2021).

3.4.2 YOLOv5

Para a YOLOv5, os métodos desenvolvidos para a determinação da posição dos objetos classificados são exatamente os mesmos utilizados pela Mask RCNN. No entanto o cálculo da orientação dos objetos é realizado de outra forma.

Conforme mencionado anteriormente, a YOLOv5 não gera como resultado do processo de classificação a máscara, ou seja, a segmentação pixel a pixel do objeto alvo. Sendo assim, fez-se necessário criar artifícios para que algo como uma máscara pudesse ser gerado. Para isso, o seguinte método foi implementado:

- Cada imagem gerada como resultado da YOLOv5 é tratada e, para cada objeto detectado, a *bounding box* correspondente é copiada formando assim uma nova imagem, de menor dimensão que a original, contendo o alvo da detecção da YOLOv5.
- Toda a imagem menor gerada pelo passo anterior tem seu espectro de cor transformado para escala de cinza, e em seguida passam por um processo de binarização.
- O processo de binarização adotado, utiliza como base o método de Otsu. Esse método é um dos mais populares algoritmos de limiarização de imagens tendo em vista que define de forma dinâmica o valor ideal de limiar para realizar a binarização de imagens.

Esse algoritmo é ideal para aplicações onde o plano de fundo da imagem e o objeto são bem definidos, fato que é caracterizado neste trabalho.

- Com a binarização, pode-se adotar os mesmos métodos, utilizando o recurso *findCountours* da biblioteca *opencv*, para determinar a angulação de cada componente classificado da imagem.

Na Figura 34 pode-se observar os resultados encontrados na aplicação do método descrito, onde o ângulo e a orientação de cada objeto pode ser determinados.

Figura 34 - Utilização do método de binarização de Otsu para geração de máscara e determinação de orientação de objeto utilizando a YOLOv5

Fonte: Elaborado pelo Autor (2021)

3.5 INTERFACE COM O ROBÔ

A partir da determinação da posição e da orientação dos objetos detectados na cena pelo sistema de visão computacional proposto, fez-se necessário elaborar um software também na linguagem de programação *python* capaz de se comunicar com o sistema de controle do robô a fim de enviar os dados de posição para o manipulador.

O sistema de controle do robô é executado por sua CPU em um sistema operacional Linux. O robô utilizado da Universal Robots possui uma interface Ethernet que foi utilizada para comunicação com o computador responsável por realizar a classificação do vídeo. Para que a comunicação entre esse computador e o robô fosse estabelecida, foi construído um algoritmo capaz de trocar dados com o sistema de controle do robô via Socket IP.

O algoritmo construído foi modularizado em funções para que sua utilização se dê de forma simples. A primeira função implementada é responsável por estabelecer a comunicação Socket IP com o sistema de controle do robô. Essa função utiliza a biblioteca socket e necessita apenas que o IP do robô e porta de comunicação entre computador e robô sejam inseridos como argumentos da função *socket.connect*.

Alguns importantes parâmetros do robô necessitam ser acessados para que o sistema possa ser implementado. Para isso, foi desenvolvido um script que acessa os dados enviados a todo momento pelo robô ao computador e direciona a leitura de parâmetros específicos por meio da definição dos bits de início e fim correspondentes a cada parâmetro. Na Tabela 4 pode-se observar os endereços dos registros que foram utilizados nesse trabalho, onde cada informação do robô é disponibilizada no mapeamento de rede. A primeira coluna descreve o parâmetro que pode ser acessado, na segunda o tipo de dado, na terceira o número de posições do vetor relacionado ao parâmetro desejado, a quarta coluna o tamanho em bytes da informação, a quinta coluna apresenta em qual endereço do mapa o parâmetro está disponibilizado e por último a sexta coluna apresenta uma breve descrição sobre o parâmetro a ser adquirido.

Tabela 4 - Mapa de comunicação do UR3, disponibilizado na porta 80 da rede. Parâmetros utilizados no controle do robô.

Parâmetro	Tipo	Número de valores	Tamanho em bytes	Posição da informação no mapa	Observações
Vetor real da pos da ferramenta	Double	6	48	56 – 61	Coodernada Cartesiana real da ferramenta: (x,y,z,rx,ry,rz), onde rx, ry e rz é uma representação da rotação do vetor da ferramenta de orientação
Força do TCP	Double	6	48	68 – 73	Forças generalizadas no TCP

Vetor da posição alvo do robô.	Double	6	48	74 – 79	Ccoordenadas cartesiana do alvo da ferramenta: (x,y,z,rx,ry,rz), onde rx, ry e rz é uma representação da rotação do vetor da ferramenta de orientação

Fonte: Adaptado de ROBOTS (2015)

Um exemplo para adquirir os valores atuais da posição do robô o baseou-se na implementação de um script que busca a informação contida entre os bits 56 e 61. Além desse parâmetro, foram implementados métodos para ler a posição alvo do robô, as forças no centro de massa da ferramenta do manipulador e a posição atual das juntas do robô. A definição dos bits de início e fim de cada informação puderam ser consultadas no manual de programação do UR3.

Além disso, foi implementado no código a funcionalidade para realizar a movimentação do robô, tal script recebe do usuário um vetor de seis posições que define a posição em x, y e z da ferramenta e a matriz de rotação do robô representada por rx, ry e rz. Essa informação é escrita via socket para o robô como argumento da função interna do equipamento *movej*. A função *movej* implementada internamente ao robô da Universal Robots é responsável por calcular a cinemática inversa para uma posição determinada de forma que as juntas realizem a menor movimentação possível e realiza a movimentação do robô de acordo com os valores de ângulos de juntas obtidos. Além disso, é passado como parâmetro para essa função a velocidade com a qual o robô realizará a movimentação.

A partir da implementação do código, foi possível controlar o robô remotamente utilizando a linguagem de programação *Python*, mesma linguagem utilizada para a implementação do sistema de visão o que proporciona a integração entre os sistemas de visão e controle. Conforme mencionado na seção 2.2.1, as informações são enviadas via socket IP por meio da programação em *scripts* no robô. A Figura 35 ilustra um exemplo de como a informação é enviada no código em Python.

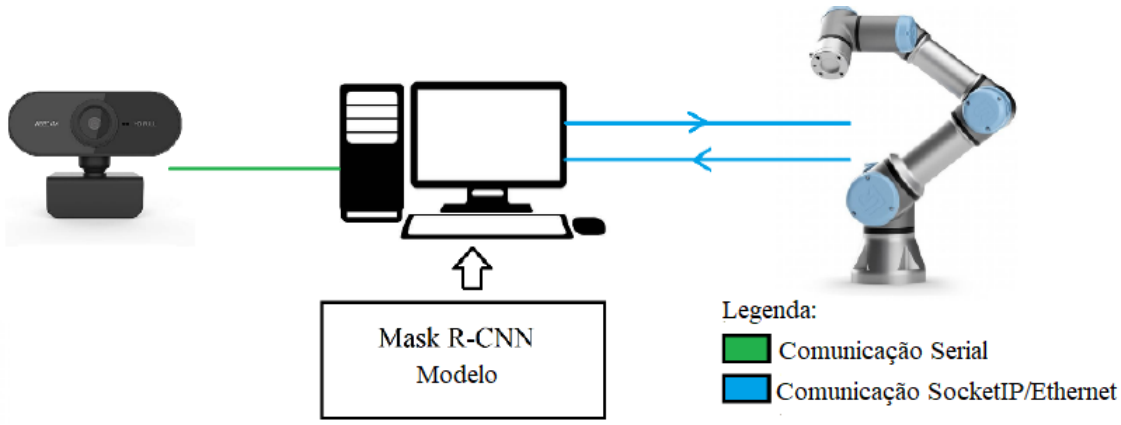
Figura 35 - Exemplo de envio de comandos e recebimento de informações do robô via Python

```
##### EXEMPLO DE COMANDO PARA O UR3 ######
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Configura a comunicação via Socket
s.connect((HOST, PORT)) #Estabelece a conexão socket IP
stringPos = "movej(p[0,0,2.45,3.1412,0,2.45], t=2, r=0)" # Instrução de comando via script conforme manual UR
s.send ((stringPos + "\n").encode('utf-8')) # Envia a string utilizando a comunicação socket em formato de texto utf-8

##### EXEMPLO DE LEITURA DE PARÂMETRO DO UR3 ######
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Configura a comunicação via Socket
s.connect((HOST, PORT)) #Estabelece a conexão socket IP
dataB=s.recv(1108) # Recebe pacote de dados
  for i in range(4,1108,8):
    temp=struct.unpack("!d",dataB[i:i+8])
    data.append(temp[0]) # Organiza o pacote de dados em forma de vetor

parameterRead=data[startBit:finalBit] # Lê parâmetro baseados nos Bits da tabela de comunicação da Tabela 4
```

Fonte: Elaborado pelo Autor (2022)

A Figura 36 ilustra a arquitetura básica do sistema implementado nesse trabalho. Nela pode-se notar que o robô se comunica com um computador via *Ethernet/SocketIP* e o sistema de visão utiliza uma *webcam* conectada a unidade de processamento via USB para a aquisição das imagens e seu posterior processamento.

Figura 36 - Arquitetura básica do sistema proposto.



Fonte: Adaptado de Meneguelli, Cavalieri e Resende (2020).

3.6 INTEGRAÇÃO ENTRE OS SISTEMAS

A etapa de integração entre os sistemas consiste basicamente na relação entre o sistema de visão e o sistema de movimentação do robô desenvolvidos. Nesta seção serão descritos os métodos utilizados para a determinação da posição real dos objetos no plano de trabalho e como esse valor é utilizado pelo sistema de movimentação do robô.

Entende-se como plano de trabalho a área onde o robô pode capturar os objetos que é a mesma área que representa a imagem capturada pela câmera. Conforme elucidado na Seção 3.4, a posição de cada objeto no plano de trabalho é encontrada a partir da intercessão das diagonais da *bouding box* do elemento classificado pela rede neural. Esse método retorna as coordenadas (xi,yi) em pixels do objeto nos eixos ordenados da imagem, faz-se necessário, portanto, encontrar as coordenadas do elemento classificado com relação ao eixo do robô, incialmente em pixels, e em seguida, realizar a conversão dos valores de posição para milímetros. Na Figura 37 pode-se observar a representação das coordenadas de um elemento classificado com relação a ambos os eixos, o eixo da imagem (xi,yi) e o do robô (xr,yr). A distância Dx em pixels pode ser encontrada a partir de medições realizadas na própria imagem. Para determinação das coordenadas do objeto no eixo do robô, as seguintes conversões são realizadas:

$$xr = xi - Dx \tag{15}$$

$$yr = yi \tag{16}$$

Tendo encontrado as coordenadas dos elementos classificados na imagem com relação ao eixo da base do robô, faz-se necessária agora a conversão dos valores encontrados em pixels de xr e yr para valores em milímetros.

Para realizar tal conversão, foram marcados no plano de trabalho dois pontos arbritários, conforme ilustrado na Figura 38, em seguida, foram realizadas as medidas entre os pontos na imagem em pixels e no próprio plano em milímetros, com o auxílio de uma régua, e com isso pôde-se determinar a relação pixel/milímetro da seguinte maneira:

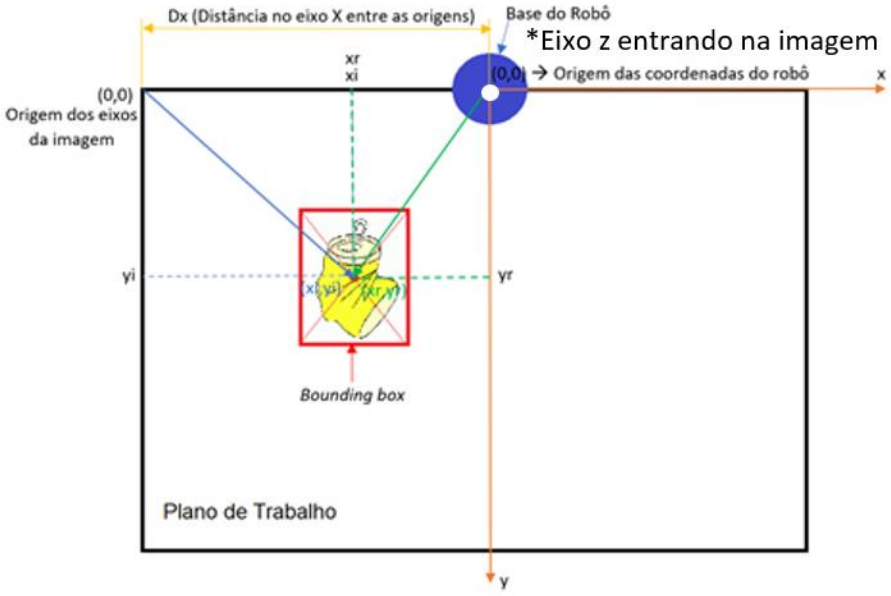
$$K = \frac{distância\ medida\ em\ milímetros}{distancia\ medida\ em\ pixels} \tag{17}$$

$$x_{mm} = K.xr \tag{18}$$

$$y_{mm} = K.yr \tag{19}$$

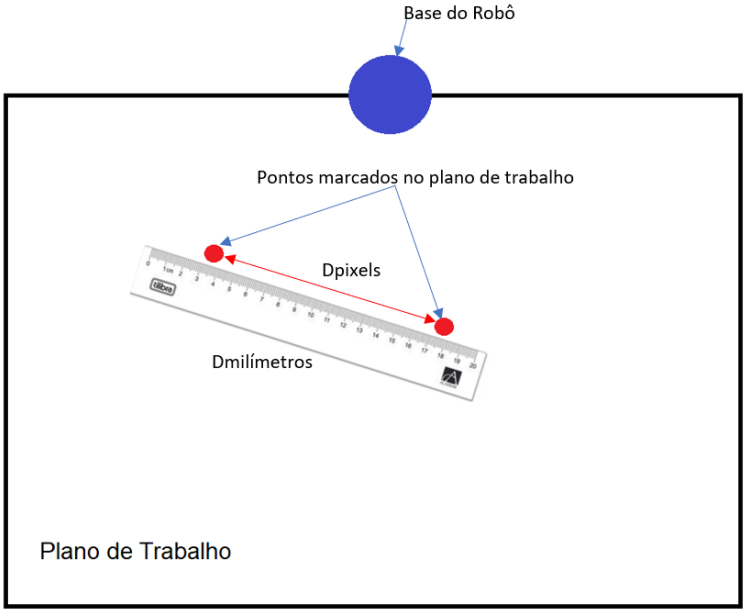
Os valores encontrados das coordenadas dos objetos classificados em milímetros (x_{mm}, y_{mm}) são utilizados como *inputs* para o sistema de movimentação do robô. A partir da definição desses valores, o robô pode se movimentar no plano de trabalho até as coordenadas apontadas a fim de capturar os objetos identificados.

Figura 37 - Representação do método utilizado para a determinação das coordenadas do objeto na imagem



Fonte: Elaborado pelo Autor (2022)

Figura 38 - Representação do método utilizado para determinação da relação pixel/milímetros.



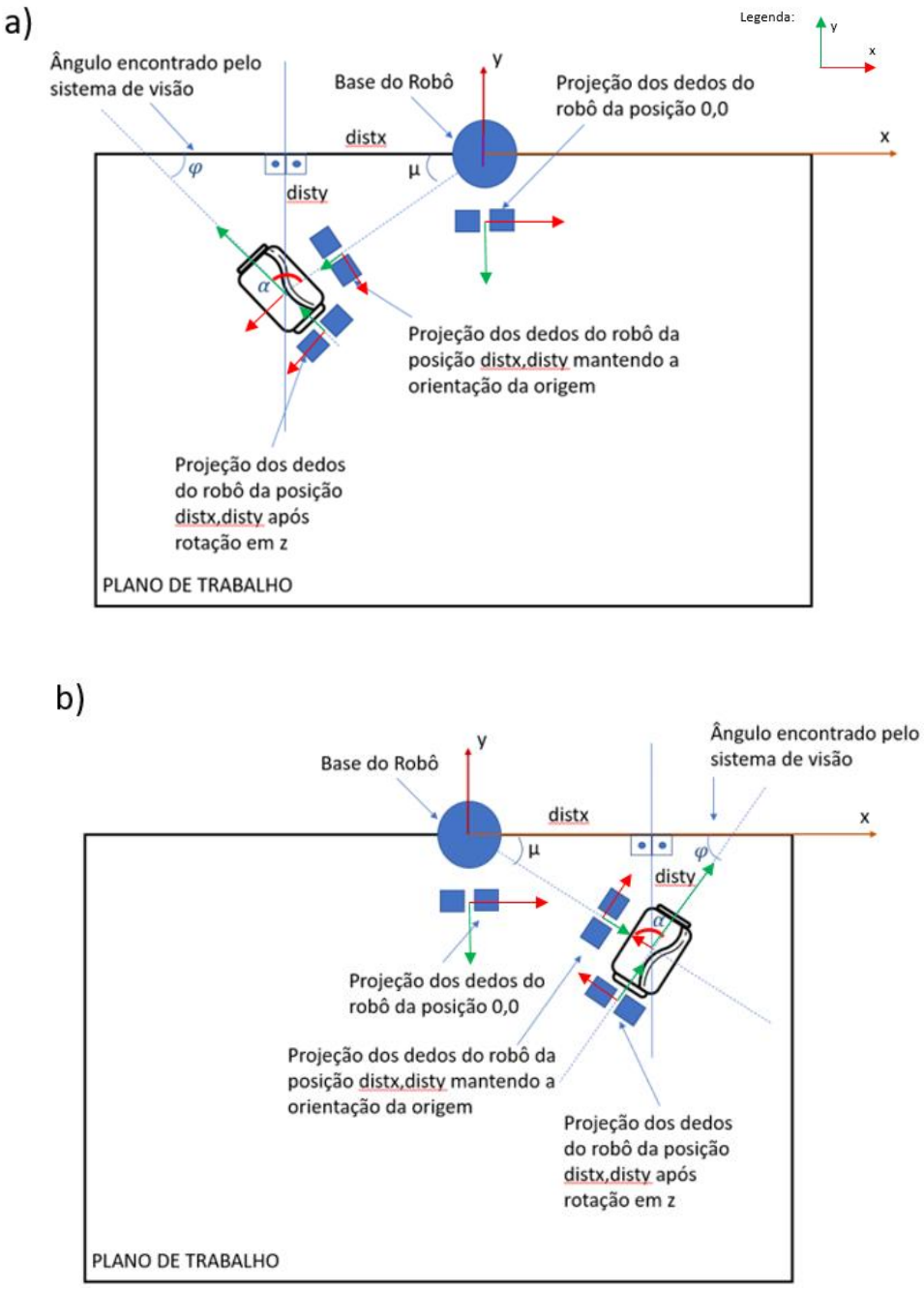
Fonte: Elaborado pelo autor (2022).

3.6.1 Utilização dos ângulos calculados no sistema de orientação do robô

Tanto para a Mask RCNN quanto para a YoloV5 calcula-se o ângulo dos objetos identificados com relação ao eixo horizontal da imagem (φ) a fim de se definir a angulação do último eixo

do robô colaborativo. A Figura 39 ilustra a relação entre o ângulo encontrado pelo sistema de visão e a posição dos dedos da garra do robô em diferentes situações.

Figura 39 - Relação entre o ângulo encontrado pelo sistema de visão computacional e o ângulo da garra do robô. Observa-se o objeto em diferentes disposições em a) e b).



Fonte: Elaborado pelo Autor (2022).

Conforme definido na seção 2.2.2 deste trabalho, o UR3 utiliza-se do vetor de rotação composto por rx, ry e rz para compor a orientação do *tool center point* (TCP). Observa-se que como os objetos encontram-se sempre sobre o plano, a garra do robô sempre trabalhará normal à

superfície, assim somente haverá rotação em torno do eixo z do sistema de orientação do robô para o ajuste da orientação da garra em relação ao objeto a ser transportado e as componentes de rotação em torno do eixo x (γ) e do eixo y *(β)* serão fixas. Observa-se também na Figura 39 que na posição em que o robô captura os objetos no plano de trabalho, existe uma rotação de 180° no eixo x, ou seja, a garra do robô assume uma posição normal ao plano, com os dedos orientados para baixo. Com isso, tem-se que a orientação da garra robô para a captura dos objetos pode ser dada por:

$$\gamma = 180° = 3.1415\, rad \tag{20}$$

$$\beta = 0° = 0\, rad \tag{21}$$

Observa-se na Figura 39 que os ângulos formados pela projeção da posição do robô e pelo ângulo encontrado pelo sistema de visão formam um triângulo e, portanto, a soma dos ângulos internos é igual à 180º. Sendo assim:

$$\alpha = (180 - \mu° - \varphi°)\frac{\pi}{180} rad \tag{22}$$

Substituindo μ pela relação trigonométrica equivalente, tem-se que:

$$\alpha = \left[180 - \tan^{-1}\left(\frac{disty}{distx}\right) - \varphi\right]\frac{\pi}{180} rad \tag{23}$$

Com isso, de posse dos ângulos de rotação em relação à cada eixo, $\alpha, \beta\ e\ \gamma$, sendo o primeiro determinado pelo sistema de visão e os dois últimos assumindo valores fixos, determina-se os valores de rx, ry e rz utilizando as equações (9), (10) e (11) apresentadas na seção 2.2.2 deste trabalho.

Os valores de rx,ry e rz encontrados são enviados ao robô de forma que esse *input* proporcione uma orientação da TCP de acordo com o ângulo do objeto a ser capturado.

4 RESULTADOS E DISCUSSÕES

Nessa seção são descritos e discutidos os resultados encontrados na aplicação das técnicas escolhidas nesse trabalho. Os próximos tópicos irão descrever os resultados obtidos pelos sistemas de visão computacional desenvolvidos (Mask RCNN e YOLOv5), pela biblioteca implementada para o comando remoto do robô e pela integração dos sistemas.

4.1 RESULTADOS OBTIDOS UTILIZANDO A MASK-RCNN

Conforme descrito anteriormente, foram utilizadas diversas configurações e combinações de parâmetros de treinamento a fim de que se atingisse os maiores percentuais de acurácia do sistema. E para cada uma dessas combinações, foram adquiridos os valores de métricas importantes como o mAP e a matriz de confusão. Além disso, foram também construídos os gráficos correspondentes de *loss e val_loss* que representam, respectivamente, o erro do modelo no processo de treinamento em valores percentuais e o erro do modelo quando aplicado ao conjunto de validação.

A Tabela 5 descreve as configurações de treinamento adotadas para seis diferentes conjuntos de parâmetros. O dado em negrito representa o melhor conjunto de parâmetros encontrados.

Tabela 5 - Diferentes conjuntos de parâmetros de treinamento testados durante o aprimoramento do modelo da Mask RCNN.

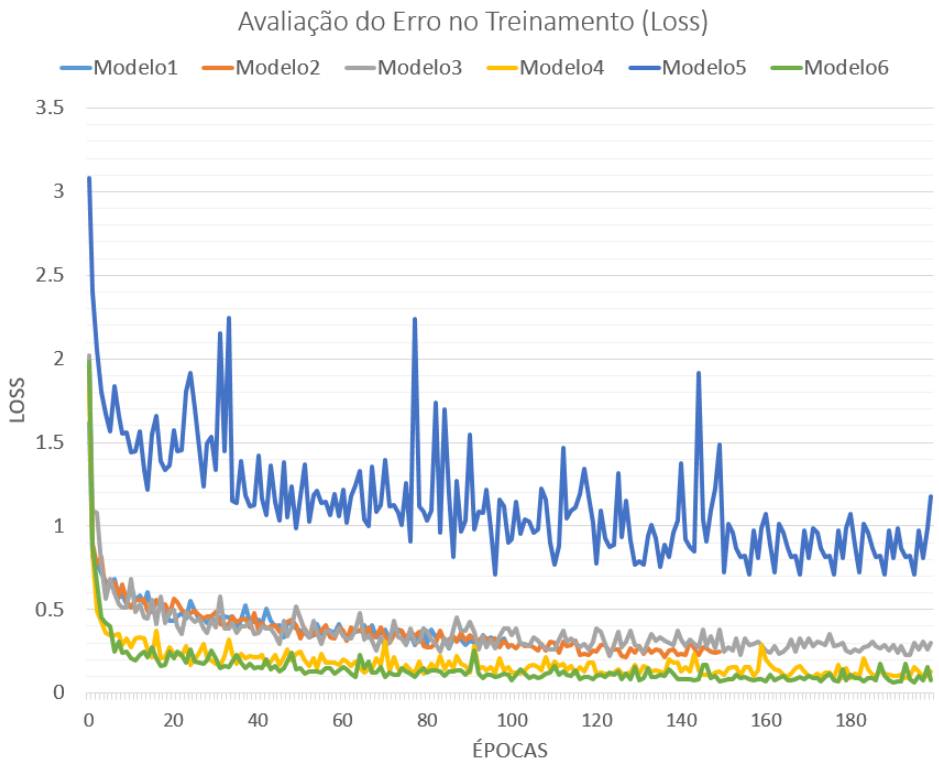
Modelo	Taxa de Aprendizado	Aumentação de dados	Épocas	Passos por época	Tamanho de entrada	Camadas da rede
1	0,001	-	100	100	1024x1024	Heads
2	0,001	*Rotation, scaling, crop, noise and blur*	150	100	1024x1024	Heads
3	**0,001**	***Rotation, scaling and crop***	**200**	**70**	**512x512**	**Heads**

4	0,001	*Rotation, scaling and crop*	200	70	512x512	All
5	0,01	*Rotation, scaling and crop*	200	70	512x512	Heads
6	0,0005	*Rotation, scaling and crop*	200	70	512x512	Heads

Fonte: Adaptado de Meneguelli, Cavalieri e Resende (2020).

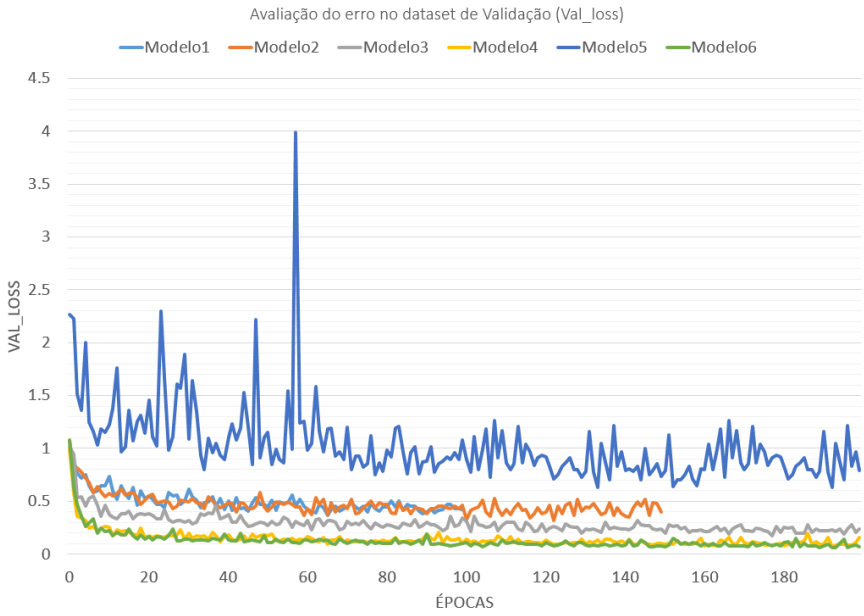
As Figura 40 e Figura 41 ilustram os valores de erro durante o processo de treinamento e no processo de validação referentes a cada um dos seis modelos testados e referenciados na Tabela 5. Além dos valores de erro foram adquiridos os valores de mAP gerados no treinamento utilizando cada conjunto de parâmetros listados na Tabela 5. Na Figura 42 pode-se observar os valores encontrados para essa métrica para cada um dos modelos testados. Conforme esperado, o modelo 3 possui o maior valor de mAP e com isso, a maior precisão dentre os modelos gerados.

Figura 40 - Erro no treinamento dos diferentes modelos



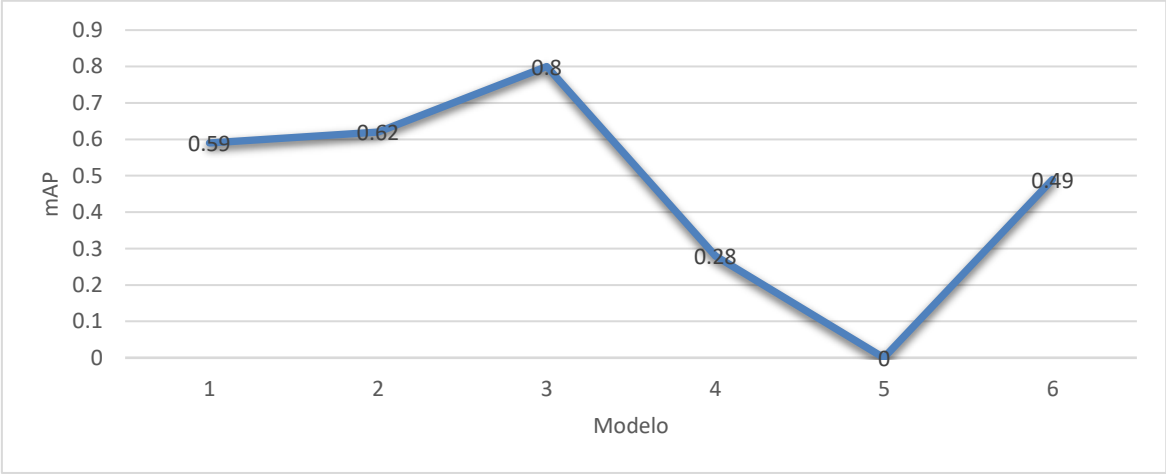
Fonte: Elaborado pelo autor (2021)

Figura 41 - Erro da classificação no processo de validação



Fonte: Elaborado pelo autor (2021)

Figura 42- Valores de mAP para cada modelo testado.

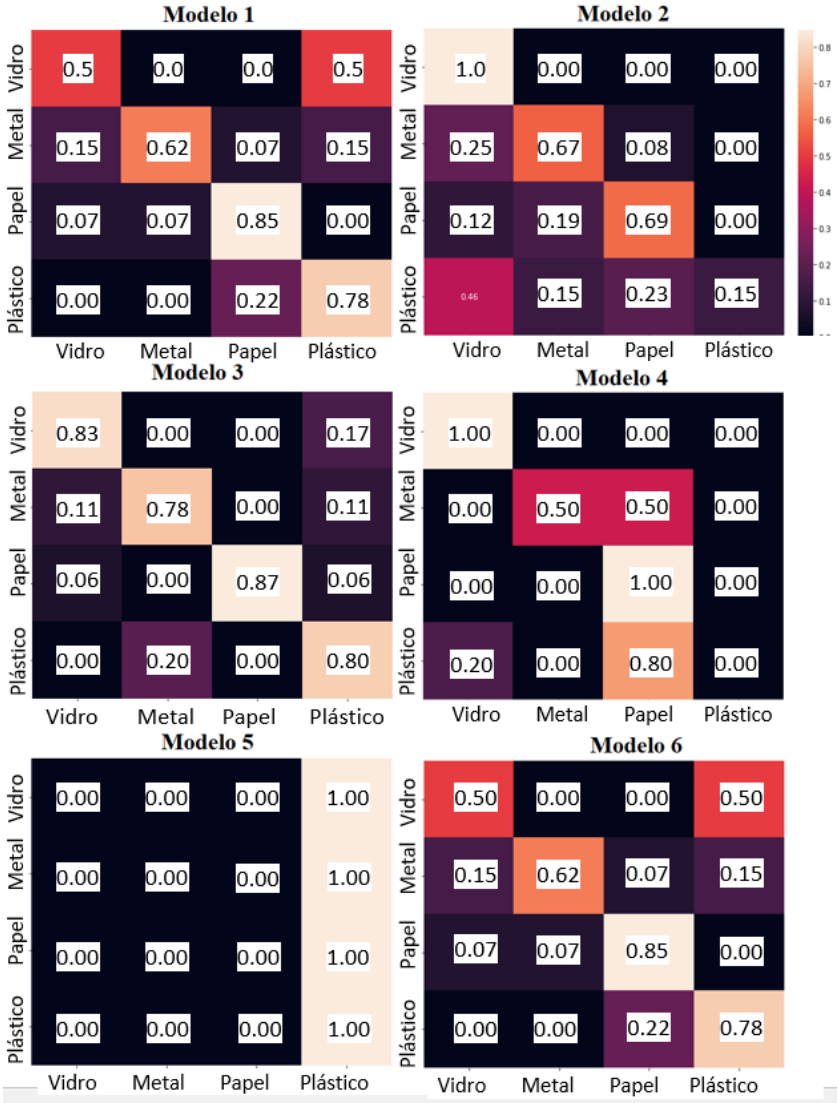


Fonte: Elaborado pelo Autor (2021).

Outra métrica utilizada para a aferição da precisão dos modelos utilizados foi a matriz de confusão. A análise dessa ferramenta é interessante pois a partir dela pode-se determinar os principais pontos de falha de classificação dos modelos e tem-se uma precisão geral de acerto para cada uma das classes testadas. A Figura 43 ilustra a matriz de confusão gerada por cada um dos modelos treinados.

Quanto maior os valores encontrados na diagonal principal das matrizes de confusão, mais preciso pode ser considerado o modelo. Sendo assim, pode-se observar que o modelo que fornece a maior precisão na identificação de todas as classes, vidro, metal, papel e plástico é novamente o modelo 3 o que reforça a afirmação de que esse foi o mais preciso modelo gerado para a resolução do problema proposto. Pode-se também observar que as classes mais confundidas são vidro e plástico devido à grande semelhança entre os materiais que as compõem. Na maioria das vezes tanto vidro como plástico são materiais translúcidos e que tem difícil distinção visual, o que corrobora com o resultado observado pelas matrizes de confusão.

Figura 43 - Matrizes de confusão geradas pelos modelos propostos com dados normalizados.

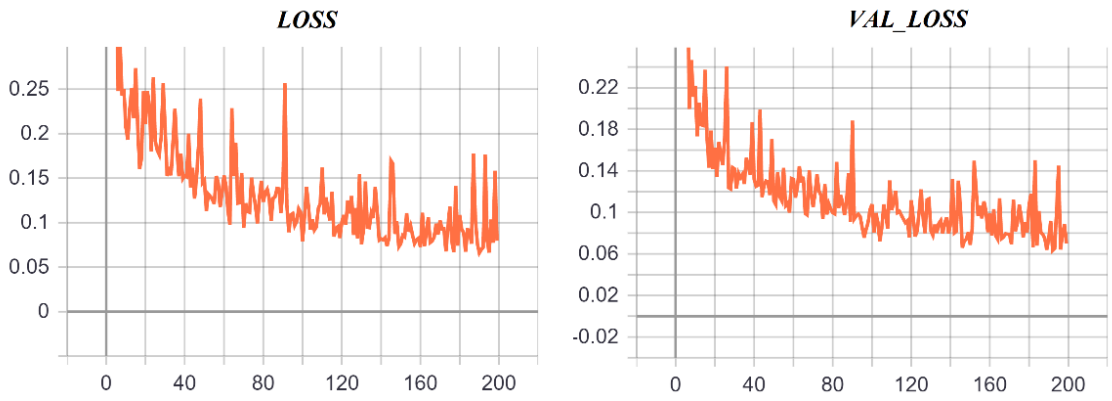


Fonte: Elaborado pelo autor (2021).

A partir dessa constatação, definiu-se que a classe vidro seria eliminada do dataset e que o modelo 3 seria treinado novamente sem essa classe. A Figura 44 ilustra os valores de erro

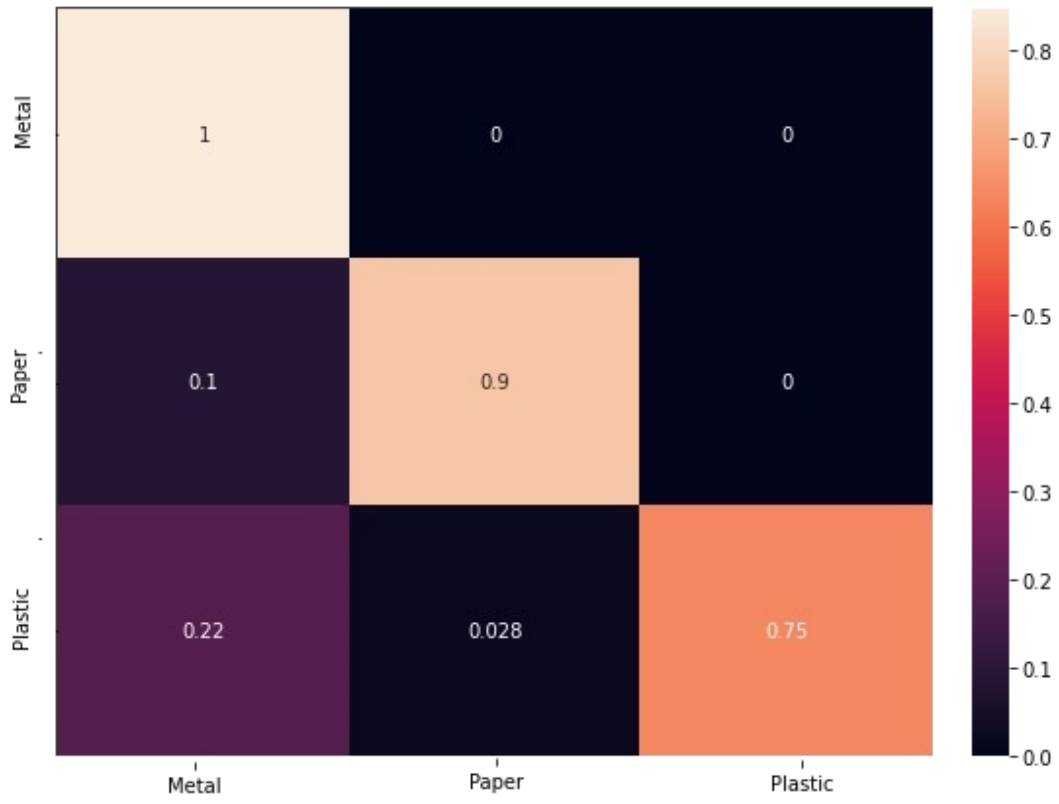
obtidos nesse novo treinamento e a Figura 45 ilustra a matriz de confusão gerada sem a classe vidro.

Figura 44 - Erros de treinamento e validação obtido sem a classe vidro pelo modelo 3.



Fonte: Elaborado pelo autor (2021)

Figura 45 - Matriz de confusão normalizada gerada pelo modelo 3 sem a classe vidro.



Fonte: Elaborado pelo autor (2021).

Conforme pode-se observar na Figura 44, os valores de erro obtidos foram inferiores aos valores obtidos pelo modelo 3 quando a classe vidro ainda estava presente o que indica o aumento na

acurácia geral do modelo. Além disso, conforme observa-se na Figura 45, a precisão na identificação de cada classe foi melhorada. Foi calculado também o valor de mAP para o novo modelo e obteve-se 0,95, valor superior aos 0,8 encontrado pelo mesmo modelo com vidro incluso entre as classes.

Sendo assim, conforme discutido e analisado, o modelo número 3 foi o mais eficiente na classificação das imagens do dataset utilizado. A Figura 46 ilustra a classificação de imagens utilizando o modelo definido.

Figura 46 - Classificação de imagens utilizando o modelo 3.

Fonte: Adaptado de Meneguelli, Cavalieri e Resende (2020).

A próxima etapa do trabalho consistiu em checar a velocidade com a qual o método proposto era capaz de identificar cada imagem e aplicar a classificação a imagens capturadas por uma *webcam*. Sendo assim, foi calculado o tempo necessário para que a rede pudesse realizar uma inferência em uma imagem submetida utilizando o hardware descrito na seção 3 desse trabalho e capturando imagens utilizando uma *webcam* com resolução 720x468 pixels. Foram obtidos, em média, 1,45 segundos de processamento de cada imagem, o que significa que o sistema é capaz de gerar a classificação das imagens a uma taxa de 0,68 quadros por segundo. A Figura 47 ilustra a imagem dos objetos classificados a partir da captura de imagens da *webcam* e utilizando o modelo 3 sem a classe vidro.

Figura 47 - Imagens classificadas utilizando imagens de *webcam* como entrada do modelo 3.

Fonte: Elaborado pelo Autor (2021).

Dessa forma, a partir dos resultados obtidos, depreende-se que o modelo obtido atende às demandas necessárias para solucionar o problema proposto por este trabalho: desenvolver um sistema protótipo capaz de separar diferentes tipos de materiais recicláveis. Conforme já mencionado, o sistema funciona de forma mais precisa sem a presença da classe vidro já que suas características visuais são muito similares ao plástico e que, por isso, o sistema desenvolvido é capaz de realizar a separação de metal, papel e plástico. Entende-se que o tempo de inferência encontrado é de certa forma alto para a aplicação proposta, porém com a utilização de um *hardware* com maior capacidade computacional esse problema é facilmente resolvido.

4.2 RESULTADOS OBTIDOS UTILIZANDO A YOLOV5

Assim, como realizado para a Mask-RCNN, foram testados diversos conjuntos de parâmetros diferentes no treinamento da YOLOv5. Da mesma forma, foram observados os valores de mAP e matrizes de confusão de cada *setup* utilizado para ajustar os parâmetros até que se encontrasse o melhor modelo.

A Tabela 6 descreve as configurações de treinamento adotadas para seis diferentes conjuntos de parâmetros. O dado em negrito representa o melhor conjunto de parâmetros encontrados. Vale ressaltar que a YoloV5 se utiliza de taxa de aprendizagem dinâmica em seu processo de treinamento e com isso esse valor não foi alterado conforme realizou-se para Mask RCNN.

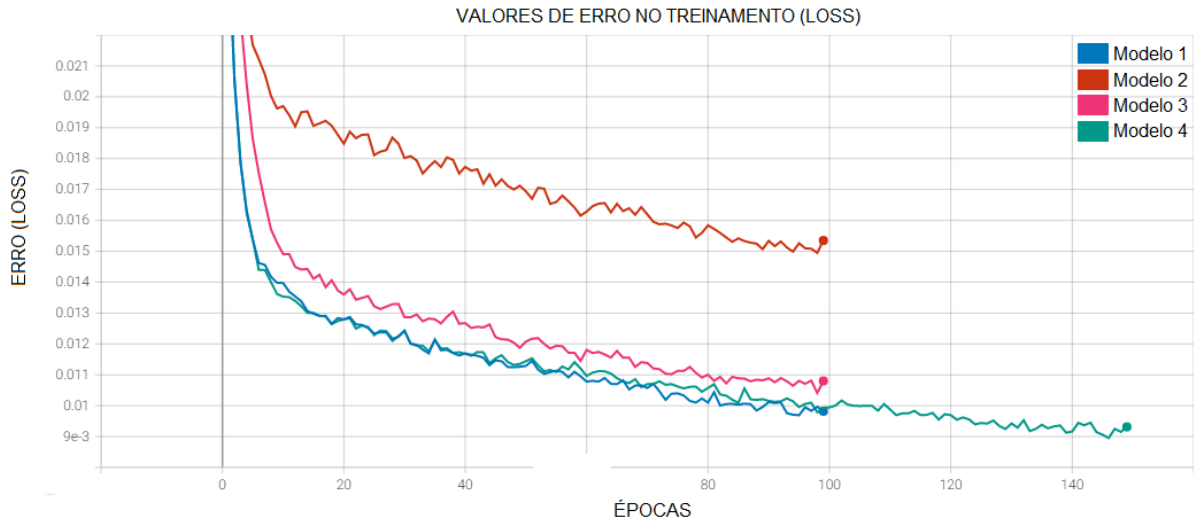
As Figura 48 e Figura 49 ilustram os valores de *loss* e *val_loss* (erros no treinamento e na validação) encontrados no treinamento dos modelos referenciados na Tabela 6. Pode-se observar que o modelo 4 é o que apresenta a menor taxa de erro tanto para o treinamento quanto para a validação. A Figura 50 ilustra os valores de mAP encontrados para cada um dos *setups* utilizados.

Tabela 6 - Diferentes conjuntos de parâmetros de treinamento testados durante o aprimoramento do modelo da YoloV5.

Modelo	Aumentação de dados	Épocas	Passos por época	Tamanho de entrada
1	Rotation (-15° a 15°), crop (0 a 5% de zoom) e flip (horizontal)	100	16	416x416
2	Rotation (-25° a 25°), crop (0 a 30% de zoom) e flip (horizontal e vertical)	100	16	416x416
3	Rotation (-15° a 15°), crop (0 a 5% de zoom) e flip (horizontal)	100	16	736x736
4	**Rotation (-15° a 15°), crop (0 a 5% de zoom) e flip (horizontal)**	**150**	**16**	**416x416**

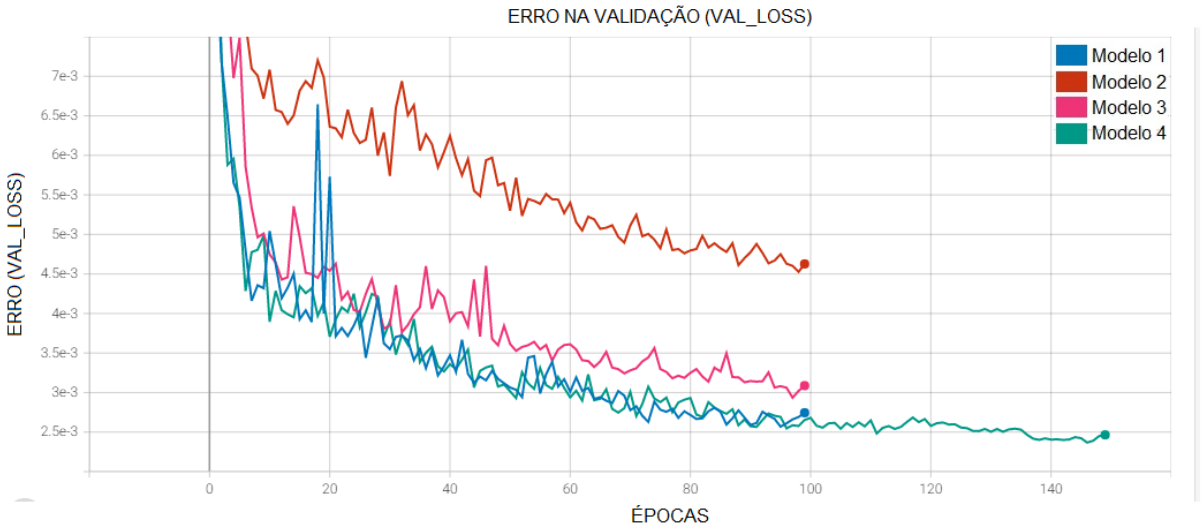
Fonte: Elaborado pelo autor (2022).

Figura 48 - Valores de erro obtidos no processo de treinamento da YoloV5



Fonte: Elaborado pelo autor (2022).

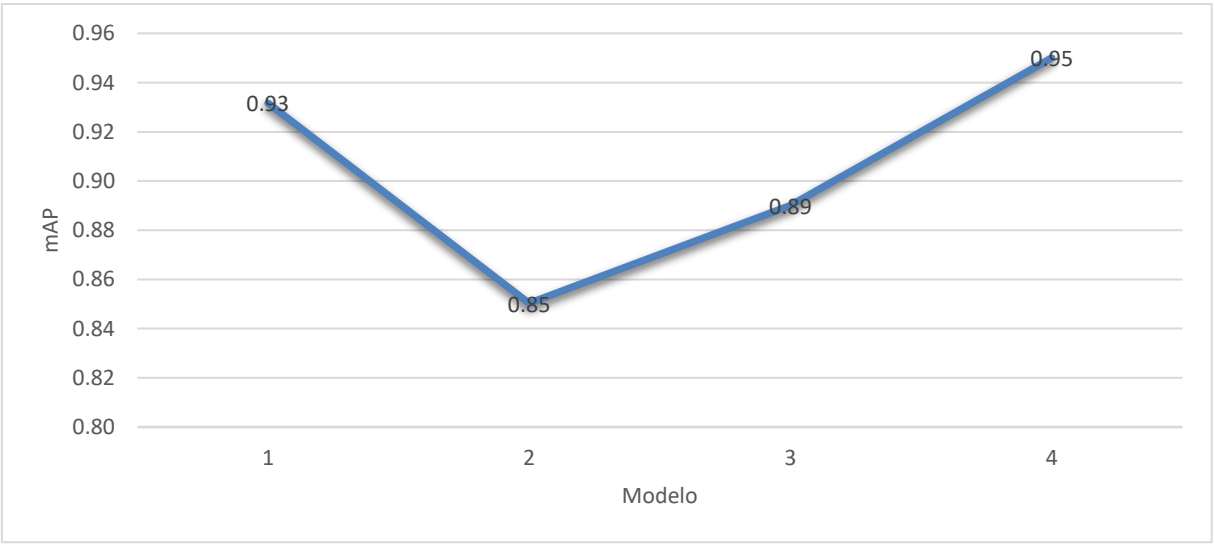
Figura 49 - Erro obtido no processo de validação do treinamento da YoloV5



Fonte: Elaborado pelo autor (2022).

Pode-se observar que novamente o modelo 4 é o que apresenta o maior valor de mAP, ou seja, é o modelo que classifica os objetos de forma mais assertiva, quanto a posição e classe.

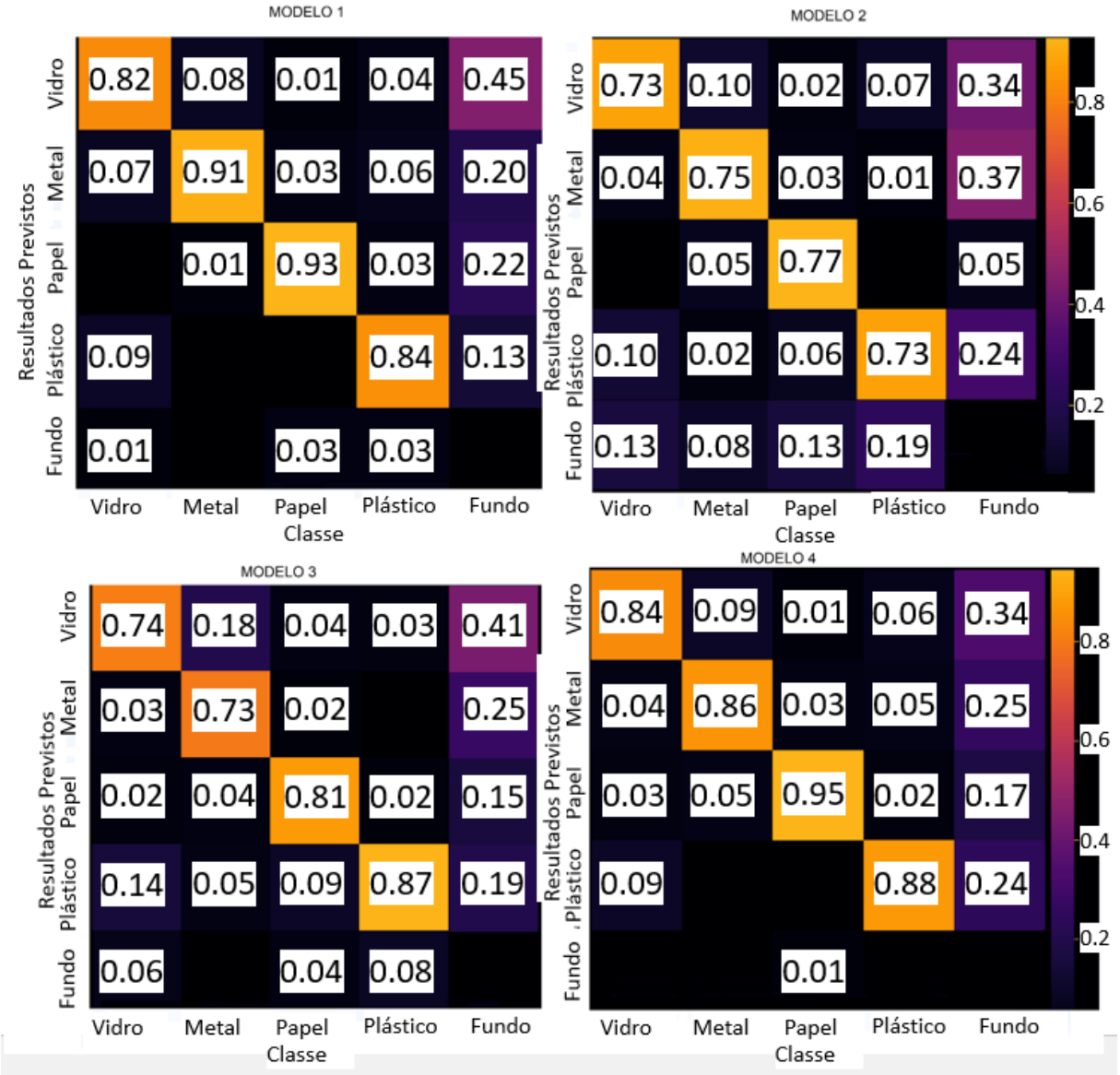
Figura 50 - Valores de mAP obtidos pelos modelos treinados



Fonte: Elaborado pelo Autor (2022).

Na Figura 51 pode-se observar as matrizes de confusão normalizadas de cada um dos modelos treinados. Novamente pode-se observar que vidro e plástico, são as classes que mais se confundem. Observa-se também que o modelo 4 é o que apresenta os valores mais elevados de assertividade para todas as classes, já que é o que apresenta valores mais próximos de um em sua diagonal principal.

Figura 51 - Matrizes de confusão geradas pelos modelos treinados

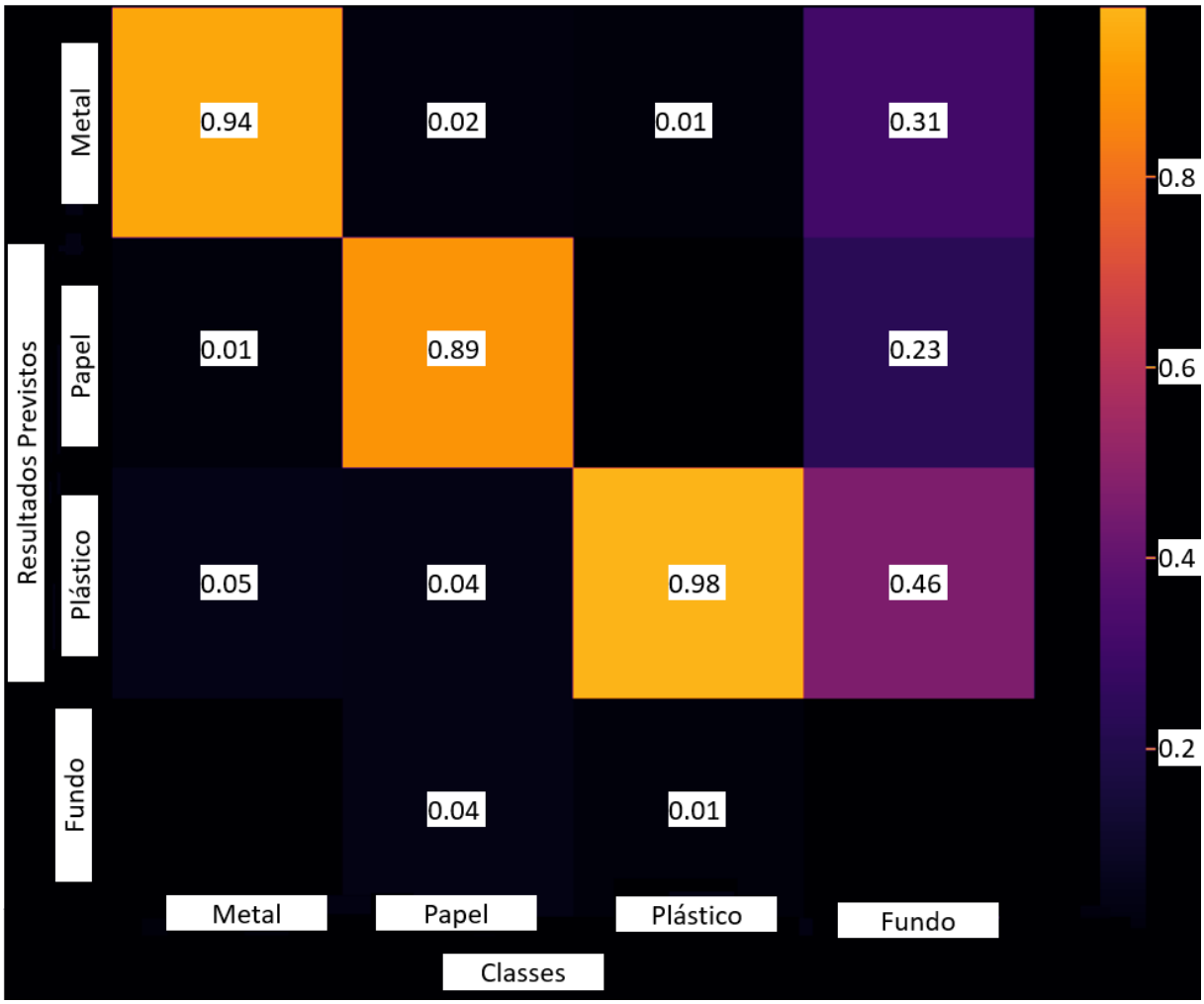


Fonte: Elaborado pelo autor (2022).

Conforme realizado com a Mask RCNN, foi realizado o treinamento de um novo modelo, com os mesmos parâmetros adotados no *setup* 4, sem a classe vidro, a fim de maximizar a taxa de acerto para as outras 3 classes. Nesse novo treinamento foi obtido um valor de mAP de 0,98. A Figura 52 ilustra a matriz de confusão obtida nesse novo treinamento.

Pode-se observar que o modelo se demonstra mais assertivo sem a classe vidro, e assim define-se como o modelo ideal para a resolução do problema proposto neste trabalho.

Figura 52 - Matriz de confusão gerada no treinamento do modelo 4 sem a classe vidro



Fonte: Elaborado pelo autor (2022)

Tendo encontrado o modelo otimizado, testou-se o *setup* encontrado com imagens adquiridas pela webcam, conforme feito com a Mask-RCNN, a fim de avaliar a velocidade de inferência da rede. Foi observado que o modelo proposto leva em torno de 0,0625s segundo para classificar uma imagem, com isso apresenta uma velocidade de inferência de 16 FPS. A Figura 53 exibe os resultados obtidos na detecção de alguns objetos utilizando a *webcam*.

Figura 53 - Exemplos de resultados de classificação gerados pelo modelo 4.

Fonte: Elaborado pelo autor (2022).

4.3 COMPARAÇÃO ENTRE OS MODELOS DE VISÃO DESENVOLVIDOS

Mediante análise dos resultados expostos nas seções anteriores, tendo em vista as métricas analisadas, pode-se afirmar que tanto a Mask-RCNN quanto a YoloV5 apresentaram precisão de classificação satisfatórias, com mAP superior a 0.9, e sendo assim, de acordo com esse parâmetro, ambas possuem capacidade de solucionar o problema proposto.

No entanto, conforme já descrito, a Mask-RCNN possui um tempo de inferência elevado quando comparada à YoloV5. Na Mask RCNN cada imagem leva cerca de 1,45s para ser classificada, enquanto a YoloV5 utilizando o mesmo *hardware* processa a mesma imagem em 0,0625s, ou seja, a YoloV5 se demonstra 23,2 vezes mais rápida do que o outro método de visão computacional utilizado neste trabalho.

Dessa forma, tendo em vista a eficiência, precisão e velocidade na classificação das imagens capturadas pela *webcam* elege-se a YoloV5 como a rede ideal para a resolução do problema proposto já que fornece a mesma precisão da Mask RCNN com velocidade de inferência significantemente maior. Deve-se ressaltar que quanto maior a velocidade de classificação das imagens maior a eficiência do sistema desenvolvido considerando que o robô deve adquirir as informações para seu posicionamento de maneira ágil e precisa.

4.4 MOVIMENTAÇÃO DO ROBÔ UTILIZANDO PYTHON

Os testes envolvendo a movimentação do robô utilizando a biblioteca implementada em Python capaz de se comunicar com o robô via socket IP foram inicialmente realizados utilizando a máquina virtual do robô da *Universal Robots* que simula uma operação real com o robô.

Inicialmente foi estabelecida a comunicação entre o robô simulado e a máquina a qual executa o código em Python. Foi configurado um endereço de IP fixo do robô.

Tendo sido realizada a comunicação com o simulador, foi testada a aquisição de parâmetros do robô como sua posição atual.

Pôde-se observar que os valores obtidos pelo código implementado conferem com os valores exibidos pela tela de operação do robô. Esses valores indicam a posição em x,y e z da ferramenta do robô bem como os valores de rotação em torno de x,y e z (rx, ry e rz), tendo como referencial a base do cobot.

O próximo teste realizado consiste basicamente em mover o robô até um ponto específico utilizando apenas e biblioteca implementada. Foram definidos arbitrariamente valores para um vetor com seis posições, que indicam as coordenadas e orientação final da ferramenta no espaço de trabalho com relação à base do robô. Foram também definidos parâmetros como o tempo o qual o robô deverá demorar para atingir o ponto alvo.

Tendo obtido sucesso com os testes realizados no simulador, os mesmos testes foram executados com o robô real. Para que os comandos via *script* fossem transmitidos ao robô, fez-se necessário habilitar a função controle remoto no terminal de operação do cobot. Os testes foram realizados e os resultados foram idênticos aos resultados já simulados.

4.5 RESULTADOS OBTIDOS NA INTEGRAÇÃO DOS SISTEMAS

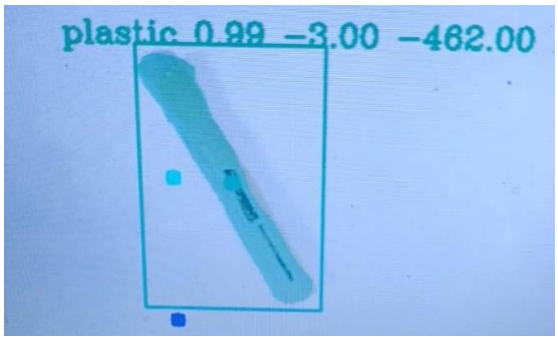
A partir da validação dos resultados dos sistemas de visão desenvolvidos e do sistema de movimentação do robô, os sistemas foram integrados conforme metodologia descrita na Seção 3.6. A Figura 54 ilustra como foi realizada a integração entre os sistemas. Na imagem pode-se observar sobre as *bounding boxes* dos objetos os valores reais das coordenadas de cada elemento em milímetros (-3,00;-462,00). Na Figura 55 pode-se observar o robô se posicionando sobre o objeto detectado a partir das coordenadas calculadas a partir da execução da função *movePos* na rotina de movimentação do robô.

Observa-se que a imagem fornece apenas as coordenadas da imagem no plano e com isso a informação de altura dos objetos identificados não pode ser definida apenas com a informação visual. Sendo assim, ajusta-se uma altura padrão, elevada com relação ao plano de trabalho, para a movimentação do robô e assim que a posição no plano xy da garra atinge as identificadas pelo sistema de visão, o cobot recebe a instrução para que movimente a garra apenas no eixo vertical no sentido para baixo. Nesse momento as forças aplicadas ao TCP são lidas e assim que é detectada uma força com sentido para cima, oposta ao sentido do movimento, é inferido que o robô atingiu sua posição no eixo vertical e a instrução de fechar garra é enviada.

A partir do posicionamento do robô e da aquisição do objeto, a próxima etapa seria depositar o lixo na lixeira correta. Para isso foram definidas coordenadas fixas para cada lixeira correspondente a cada material (plástico, metal e papel). E assim, de acordo com a saída da classificação da rede, o robô se movimenta para a coordenada correspondente ao tipo de material identificado pelo sistema de visão. Assim que o manipulador robótico atinge a posição

determinada, o comando de abrir garra é dado e o objeto é depositado em sua destinação. Até que não haja mais objetos detectados sob o plano de trabalho, o processo descrito é repetido. A Figura 56 ilustra o deposito do material coletado em sua destinação adequada.

Figura 54 - Detecção de objeto plástico no plano de trabalho. Sobre a *bounding box* estão representados respectivamente a classe à qual o elemento pertence, a precisão da classificação e as coordenadas em x e y em milímetros.

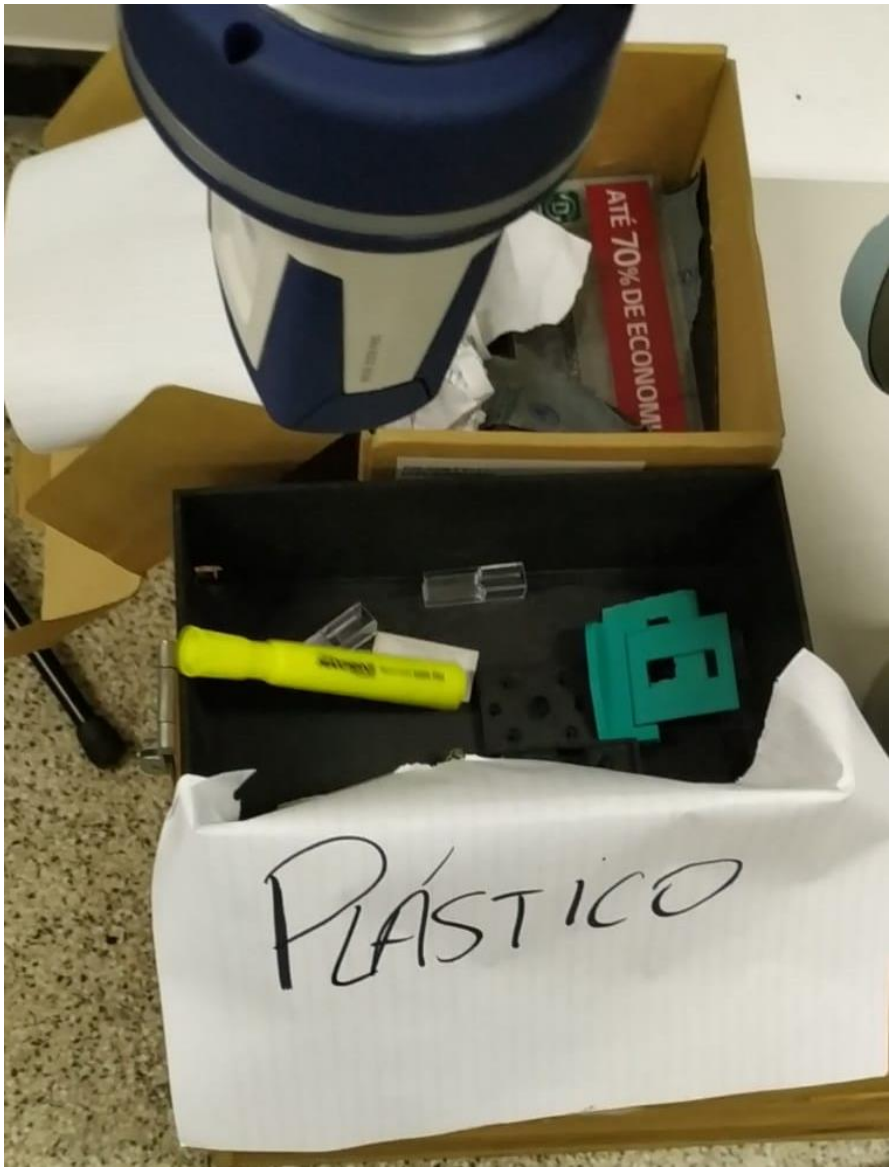


Fonte: Elaborado pelo autor (2021).

Figura 55- Robô se posicionando sobre o objeto detectado

Fonte: Elaborado pelo autor (2021)

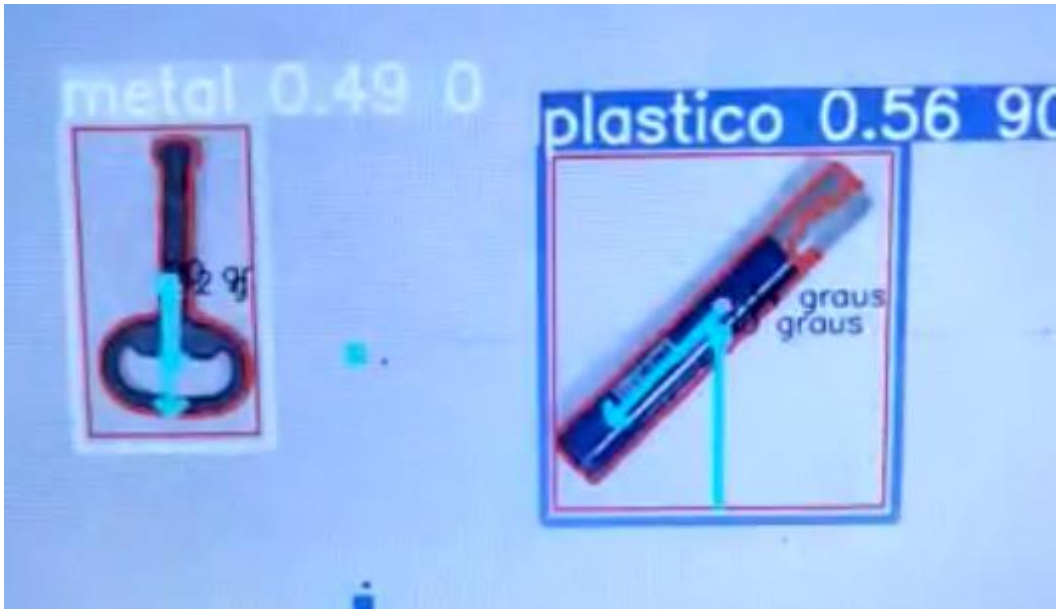
Figura 56 - Robô depositando material na lixeira adequada



Fonte: Elaborado pelo autor (2021)

Para situações em que mais de um objeto é detectado, como ilustrado na Figura 52, o sistema trata os objetos na seguinte ordem de prioridade: plástico, papel e metal. Ou seja, a partir da detecção dos objetos pelo sistema de visão, inicialmente todos os objetos plásticos são capturados, caso a classe papel tenha sido identificada o robô em seguida captura todos os objetos pertencentes a essa classe e por último os objetos metálicos. As Figuras 53, 54, 55 e 56 ilustram a sequência descrita.

Figura 57 - Detecção de dois objetos de classes diferentes no plano de trabalho



Fonte: Elaborado pelo Autor (2022).

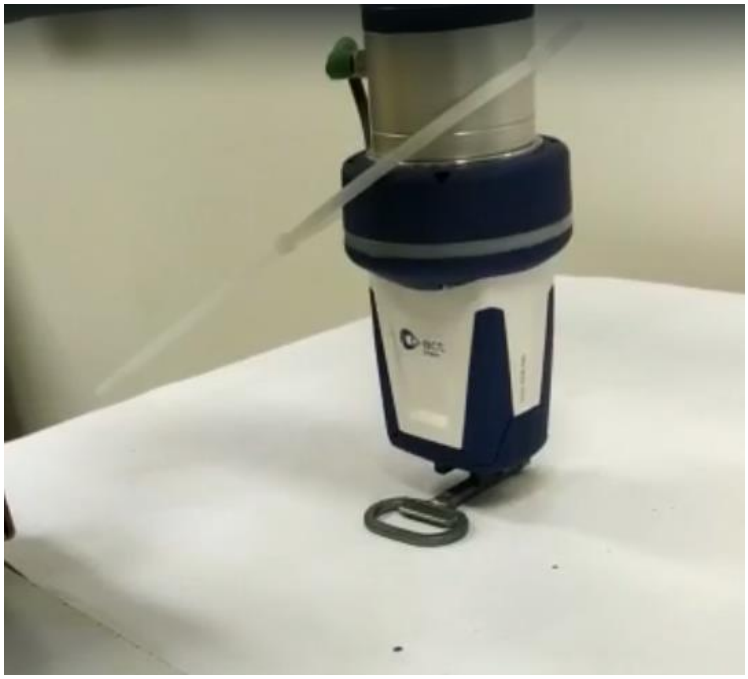
Figura 58 - Representação da priorização da classe plástico quando há a classificação de mais de um elemento na imagem.

Fonte: Elaborado pelo Autor (2022).

Figura 59 - Alocação do objeto plástico na lixeira determinada para essa classe de resíduo

Fonte: Elaborado pelo Autor (2022).

Figura 60 - Aquisição do objeto metálico na sequência do descarte do objeto plástico



Fonte: Elaborado pelo Autor (2022).

Figura 61 - Descarte do objeto metálico no local apropriado

Fonte: Elaborado pelo Autor (2022)

O funcionamento do sistema pode ser observado no seguinte link: https://youtu.be/w6xpPb7TXQQ.

A partir dos resultados obtidos com a implementação do sistema, algumas constatações podem ser realizadas imaginando a aplicação do sistema em uma planta de separação de lixo. No experimento realizado, foi utilizada uma garra com dois dedos como ferramenta do robô, constatou-se que esse elemento não se demonstra como o mais adequado para a aplicação tendo em vista que alguns elementos não conseguem ser capturados por essa ferramenta, como folhas de papel por exemplo. Uma ferramenta de ventosas a vácuo teria maior capacidade de capturar materiais de formatos mais genéricos que uma garra convencional e tornaria o sistema mais eficiente.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram abordadas duas técnicas atuais de *Machine Learning*, a *Mask R-CNN* e a YOLOv5. Em conjunto com a implantação de um código em *Python* capaz de se comunicar com um robô colaborativo industrial, o UR3 da *Universal Robots*, o sistema desenvolvido foi capaz de cumprir com os objetivos deste trabalho, realizar a separação de lixo reciclável utilizando um cobot e câmeras de vídeo.

O trabalho compara dois métodos de classificação e segmentação dos objetos, a Mask-RCNN e a YOLOv5, sendo o segundo o que demonstrou melhores características para resolver o problema proposto pelo trabalho. Como pôde ser observado, a Mask R-CNN mostrou-se robusta na classificação dos objetos analisados, apresentando facilidade em sua implementação e adaptabilidade. No entanto, esse modelo demonstrou-se custoso computacionalmente o que torna a detecção dos objetos lenta. Com isso foi desenvolvido outro sistema de visão computacional com o mesmo objetivo do primeiro, dessa vez utilizando a YOLOv5. Esse novo modelo demonstrou-se tão eficiente quanto a Mask-RCNN e apresentou velocidade de inferência 23 vezes superior, o que o caracteriza como o modelo ideal para a aplicação proposta.

A aplicação completamente desenvolvida em *Python* promove uma fácil integração entre os sistemas de visão e controle possibilitando o ajuste e a implementação de funções que conversam entre si e promovem a movimentação do robô baseada nas imagens classificadas e obtidas pelos sistemas de visão.

O sistema se mostrou robusto e poderia ser aplicado em diversas situações reais. O robô em conjunto com o sistema de visão poderia cumprir com o papel de um indivíduo no processo de separação de lixo, que poderia ser realocado para realização de outra atividade. Outra alternativa seria a utilização do robô apenas para separar um tipo de material específico, por exemplo metal, e outros indivíduos separarem outros tipos de resíduo.

5.1 TRABALHOS FUTUROS

Como sugestão de trabalhos futuros pode-se citar a aplicação de outras técnicas de segmentação de instâncias para a resolução do problema como a U-Net, a fim de comparar os resultados obtidos por essa rede com a Mask R-CNN e a YOLOv5. Além disso, a U-Net possuiu um menor custo computacional na inferência de imagens o que aumenta a eficiência do sistema e diminui

a necessidade de hardwares com maior valor para solucionar o problema de identificação dos elementos propostos.

Outra sugestão para futuros trabalhos é a implementação de um sistema com a câmera fixada ao pulso do robô. Dessa forma tem-se referências dinâmicas e maior possibilidades de visualização dos objetos em diferentes ângulos. Outra sugestão é a integração de vários robôs trabalhando em um mesmo espaço de trabalho cada um responsável por capturar determinado tipo de material. Além disso, para que o problema de similaridade entre vidro e plástico seja resolvido, pode-se aliar o sistema de visão computacional à um sensoriamento infravermelho ou indutivo capaz de diferenciar os dois tipos de materiais.

REFERÊNCIAS

ARLEN, T. C. **Understanding the mAP Evaluation Metric for Object Detection**. Disponível em: <https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3>. Acesso em: 16 abr. 2021.

BLOSS, R. Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications. **Industrial Robot**, v. 43, n. 5, p. 463–468, 2016.

CIRES, D. C. et al. IJCAI11-210.pdf. **Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence Flexible**, p. 1237–1242, 2003.

CNI. Nota Técnica do Ministério do Trabalho esclarece o uso de "Robôs Colaborativos" frente à NR 12. **RT Informa**, v. 4, n. 11, p. 1–2, 2018.

EL ZAATARI, S. et al. Cobot programming for collaborative industrial tasks: An overview. **Robotics and Autonomous Systems**, v. 116, p. 162–180, 2019.

ERDEM, K. **Understanding Region of Interest — (RoI Align and RoI Warp)**. Disponível em: <https://towardsdatascience.com/understanding-region-of-interest-part-2-roi-align-and-roi-warp-f795196fc193>. Acesso em: 16 abr. 2021.

EVERINGHAM, M. et al. The pascal visual object classes (VOC) challenge. **International Journal of Computer Vision**, v. 88, n. 2, p. 303–338, 2010.

FENG, V. **An Overview of ResNet and its Variants**. Disponível em: <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>. Acesso em: 8 abr. 2021.

GIRSHICK, R. Fast R-CNN. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2015 Inter, p. 1440–1448, 2015.

GUNDUPALLI, S. P.; HAIT, S.; THAKUR, A. A review on automated sorting of source-separated municipal solid waste for recycling. **Waste Management**, v. 60, p. 56–74, 2017.

GUTTA, S. **Object Detection Algorithm — YOLO v5 Architecture**. Disponível em: <https://medium.com/analytics-vidhya/object-detection-algorithm-yolo-v5-architecture-89e0a35472ef>. Acesso em: 14 dez. 2021.

GUYON, I. A scaling law for the validation-set training-set size ratio. **AT&T Bell Laboratories**, p. 1–11, 1997.

HE, K. et al. Deep residual learning for image recognition. **Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, v. 2016-Decem, p. 770–778, 2016.

HE, K. et al. Mask R-CNN. **Proceedings of the IEEE International Conference on Computer Vision**, v. 2017- Octob, p. 2980–2988, 2017.

HE, K. et al. Mask R-CNN. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 42, n. 2, p. 386–397, 2020.

HOLLOWAY, C. C. Method for separation, recovery, and recycling of plastics from municipal solid waste. **U.S. Patent**, n. 4,844,351, 1989.

JOCHER, G. **YoloV5**. Disponível em: <https://github.com/ultralytics/yolov5>. Acesso em: 3 jan. 2022.

KHANDELWAL, R. **Computer Vision: Instance Segmentation with Mask R-CNN**. Disponível em: <https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1>. Acesso em: 16 abr. 2021.

KULCKE, A. et al. On-line classification of synthetic polymers using near infrared spectral imaging. **Journal of Near Infrared Spectroscopy**, v. 11, n. 1, p. 71–81, 2003.

LAVALLE, S. M. **Planning Algorithms**. Disponível em: <http://planning.cs.uiuc.edu/node102.html>. Acesso em: 17 fev. 2022.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LIN, T.-Y. et al. Microsoft {COCO:} Common Objects in Context. **CoRR**, v. abs/1405.0, 2014.

MALATO, G. **Hyperparameter tuning. Grid search and random search**. Disponível em: <https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/>. Acesso em: 2 jan. 2022.

MATHESON, E. et al. Human-robot collaboration in manufacturing applications: A review. **Robotics**, v. 8, n. 4, p. 1–25, 2019.

MATTERPORT. **Mask R-CNN for Object Detection and Segmentation**. Disponível em: <https://github.com/matterport/Mask_RCNN>. Acesso em: 16 abr. 2021.

MENEGUELLI, R.; CAVALIERI, D. C.; RESENDE, C. Z. Recyclable Waste Classification Using a Deep Learning Vision System. 2020.

MÜLLER, R.; VETTE, M.; GEENEN, A. Skill-based Dynamic Task Allocation in Human-Robot-Cooperation with the Example of Welding Application. **Procedia Manufacturing**, v. 11, n. June, p. 13–21, 2017.

NORMUNG, E. K. FÜR. ISO/TS 15066 - Robots and robotic devices — Collaborative robots. v. 2016, 2016.

PANIGRAHI, S.; NANDA, A.; SWARNKAR, T. A Survey on Transfer Learning. **Smart Innovation, Systems and Technologies**, v. 194, n. 10, p. 781–789, 2021.

PAULRAJ, S. G.; HAIT, S.; THAKUR, A. Automated municipal solid waste sorting for recycling using a mobile manipulator. **Proceedings of the ASME Design Engineering Technical Conference**, v. 5A-2016, n. September, 2016.

PESCE, C. P. DINÂMICA DOS CORPOS RÍGIDOS. 2004.

PICKIT. **PickIt-3D**.

RAJPUT, M. **YOLO V5 — Explained and Demystified**. Disponível em: <https://towardsai.net/p/computer-vision/yolo-v5-explained-and-demystified>. Acesso em: 26 dez. 2021.

RAWAT, W.; WANG, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review Waseem. **Neural Computation**, 2017.

ROBOTS, U. Realtime Client Interface. 2015.

ROBOTS, U. **e-Series. Built to do more**. Disponível em: <https://www.universal-robots.com/pt/produtos/ur3-robot/>. Acesso em: 7 jun. 2021.

SALIMI, I.; BAYU DEWANTARA, B. S.; WIBOWO, I. K. Visual-based trash detection and classification system for smart trash bin robot. **International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2018 - Proceedings**, n. February 2019, p. 378–383, 2019.

SMEDA, K. **Understand the architecture of CNN**. Disponível em: <https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7>. Acesso em: 31 ago. 2022.

VILLANI, V. et al. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. **Mechatronics**, v. 55, n. February, p. 248–266, 2018.

VO, A. H. et al. A Novel Framework for Trash Classification Using Deep Transfer Learning. **IEEE Access**, v. 7, p. 178631–178639, 2019.

VUOLA, A. O.; AKRAM, S. U.; KANNALA, J. Mask-RCNN and u-net ensembled for nuclei segmentation. **Proceedings - International Symposium on Biomedical Imaging**, v. 2019-April, n. May, p. 208–212, 2019.

WANG, C. et al. CSPNET: A NEW BACKBONE THAT CAN ENHANCE LEARNING CAPABILITY OF CNN. 2019.

WINKLER, D. A.; LE, T. C. Performance of Deep and Shallow Neural Networks, the Universal Approximation Theorem, Activity Cliffs, and QSAR. **Molecular Informatics**, v. 36, n. 1–2, p. 1600118, 2017.

WORLD ROBOTICS. **IFR presents World Robotics 2021 reports**. Disponível em: <https://ifr.org/ifr-press-releases/news/robot-sales-rise-again>.

XU, R. et al. A forest fire detection system based on ensemble learning. **Forests**, v. 12, n. 2, p. 1–17, 2021.

YANG, M.; THUNG, G. Classification of Trash for Recyclability Status. **CS229Project Report**, p. 1–6, 2016.

ZHIHONG, C. et al. A vision-based robotic grasping system using deep learning for garbage sorting. **Chinese Control Conference, CCC**, p. 11223–11226, 2017.