Visão Computacional aplicada a um Manipulador Robótico: Um caso de estudo com YOLOv8 e Robo Staubli TS60

Guilherme Henrique Paiva Ferreira e Guilherme Miguel Roque .

***Resumo*— A inteligência artificial tem impulsionado avanços significativos nas aplicações industriais, especialmente na automação e controle de processos. Modelos de *deep learning*, como redes neurais convolucionais (CNNs), têm sido particularmente eficazes nessas aplicações. Este estudo apresenta uma solução de *Pick and Place*, utilizando um manipulador robótico SCARA e um sistema de visão computacional baseado no modelo YOLOv8. A comunicação entre o sistema de visão computacional e o robô SCARA é realizada via socket, permitindo a detecção de peças em tempo real e otimizando a precisão das operações robóticas. Os cinco modelos YOLOv8 treinados foram avaliados, com o modelo de maior acurácia atingindo 96.7% e tempo de processamento de 190,83 ms. Todos os modelos apresentaram tempos de processamento abaixo de 300 ms e desempenhos semelhantes, viabilizando a aplicação em tempo real.**

***Palavras-Chave*— Inteligência Artificial, SCARA, Socket, Visão Computacional, YOLOv8.**

***Abstract*— Artificial intelligence has driven significant advancements in industrial applications, particularly in automation and process control. Deep learning models, such as convolutional neural networks (CNNs), have been particularly effective in these applications. This study presents a *Pick and Place* solution utilizing a SCARA robotic manipulator and a computer vision system based on the YOLOv8 model. Communication between the computer vision system and the SCARA robot is achieved via socket, enabling real-time detection of parts and optimizing the precision of robotic operations. The five YOLOv8 models trained were evaluated, with the highest accuracy model achieving 96.7% and a processing time of 190.83 ms. All models demonstrated processing times below 300 ms and similar performances, enabling real-time application.**

***Keywords*— Artificial Intelligence, Computer Vision, SCARA, Socket, YOLOv8**

I. Introdução

A robótica representa uma nova era na manufatura e produção industrial, caracterizada pela integração de tecnologias digitais avançadas. O advento da inteligência artificial (IA) e do aprendizado profundo tem impulsionado uma revolução significativa nas aplicações industriais, promovendo melhorias notáveis na automação e no controle de processos. Entre as inovações mais destacadas, o uso da visão computacional (*VC*) vem ganhando destaque devido à sua versatilidade e ampla gama de aplicações. O processamento de imagens por meio de algoritmos de detecção de objetos em tempo real permite a localização e classificação simultânea e precisa de múltiplos objetos em ambientes tridimensionais complexos [1].

Os manipuladores robóticos têm desempenhado um papel cada vez mais proeminente na automação industrial. No entanto, esses sistemas frequentemente enfrentam desafios relacionados à flexibilidade e adaptabilidade a diferentes ambientes e tarefas de produção. A combinação de *VC* e robótica oferece uma solução promissora para esses desafios, permitindo que os robôs sejam mais adaptáveis e precisos em suas operações. Este estudo utiliza a rede neural You Only Look Once (YOLO), reconhecida por sua eficiência e desempenho em tarefas de detecção de objetos em tempo real. A versão YOLOv8, em particular, tem demonstrado resultados promissores em termos de acurácia, precisão e confiança, sendo uma escolha ideal para a identificação e localização de peças em ambientes dinâmicos [2].

Diante desse cenário, existe uma demanda crescente por soluções inovadoras que possam otimizar os processos de movimentação de peças em ambientes industriais. A tarefa de *Pick and Place*, fundamental em muitas linhas de produção, é significativamente beneficiada pela utilização de robôs *Selective Compliant Assembly Robot Arm* (SCARA), que se destacam em termos de velocidade e precisão. A partir disso, este estudo propõe uma abordagem de integração entre um sistema de *VC* e um manipulador robótico SCARA, com o objetivo de automatizar a detecção de peças, permitindo que as operações de manipulação e transporte sejam feitas com base na posição da peça em tempo real.

Neste trabalho, foram treinadas cinco versões do modelo YOLOv8 para avaliar as métricas de desempenho de cada um. A integração do sistema de *VC* foi realizada utilizando um notebook, mas a solução pode ser facilmente adaptada para outros dispositivos. Este sistema visa automatizar efetivamente as operações de manipulação de peças, contribuindo para a melhoria da eficiência, produtividade e capacidade de adequação a cenários de produção onde a peça possa estar em posições adversas.

Este trabalho está organizado da seguinte forma: na Seção II são apresentados conceitos importantes sobre os modelos de *VC* e tipos de manipuladores. A Seção III apresenta alguns trabalhos relacionados e discussões sobre os mesmos. A seção IV apresenta o material e métodos empregados para

Trabalho de Conclusão de Curso apresentado ao Instituto Nacional de Telecomunicações (INATEL) como parte dos requisitos para obtenção do Título de Bacharel em Engenharia de Controle e Automação. Aprovado em 18/06/2024 pela comissão julgadora: Prof. MSc. Eduardo Henrique Teixeira / INATEL - Orientador e Presidente da Comissão Julgadora, Prof. MSc. João Paulo Carvalho Henriques / INATEL - Coorientador, Prof. Dr. Yvo Marcelo Chiaradia Masselli / INATEL - membro da comissão avaliadora, MSc. Mateus Raimundo da Cruz / INATEL - membro da comissão avaliadora. Coordenador do Curso de Engenharia de Controle e Automação: Prof. Dr. Alexandre Baratella Lugli.

implementar a solução proposta. A seção VI apresenta os experimentos e resultados e, finalmente, na Seção VII, são apresentadas as conclusões dos autores.

II. Fundamentação Teórica

Esta seção tem como objetivo apontar os tipos mais comuns de manipuladores robóticos e detalhar os modelos de *VC* mais utilizados na literatura. Existem diferentes tipos de manipuladores robóticos, cada um com características e aplicações específicas. Os cinco tipos principais de manipuladores são robô articulado, robô esférico, robô cilíndrico, robô cartesiano e robô SCARA. Cada um desses manipuladores possui configurações de juntas que os tornam adequados para diferentes tarefas e ambientes industriais [3].

Além dos diferentes tipos de manipuladores, os modelos de detecção de objetos têm sido amplamente utilizados na literatura, destacando-se três modelos principais: *Region-based Convolutional Neural Networks* (RCNN), *Single Shot MultiBox Detector* (SSD) e *You Only Look Once* (YOLO). Esses modelos têm demonstrado grande eficácia em tarefas deVC, permitindo a detecção de objetos em tempo real com alta precisão dos modelos. [1].

A. Tipos de Manipuladores

Os manipuladores robóticos podem ser classificados em cinco tipos principais [3]:

1) **Robôs Articulados**: Amplamente utilizados nas indústrias, possuem um desenho similar ao de um braço humano, composto por eixos que se dobram em diversas direções, aumentando o alcance e as possibilidades de aplicação. Existem robôs articulados de 2 a 10 eixos, variando o número de pontos de conexão e dobra conforme a necessidade ou o tamanho da máquina. Os mais comuns são os robôs de 6 eixos, incluindo os robôs colaborativos. As principais vantagens dos robôs industriais articulados são a flexibilidade para aplicações em espaços confinados e sua facilidade para alinhamento de coordenadas.
2) **Robôs Esféricos**: Combinam 2 juntas rotacionais e uma prismática, permitindo movimentos em coordenadas polares ou esféricas. Sua versatilidade garante movimentação precisa em torno de si, sendo utilizados em missões de exploração interplanetária, avaliação de ambientes perigosos e vigilância.
3) **Robôs Cilíndricos**: Úteis em situações que envolvem objetos e fenômenos com simetria rotacional em torno de um eixo longitudinal. Eles são capazes de manusear materiais e realizar tarefas de *pick and place* facilmente, devido à sua menor complexidade e custo mais baixo. A maioria dos robôs cilíndricos realiza 2 movimentos de translação e um de rotação, utilizando 2 juntas prismáticas e uma rotacional.
4) **Robôs Cartesianos**: Escolhidos pela sua simplicidade e precisão, operam com coordenadas cartesianas e possuem 3 juntas prismáticas ortogonais, proporcionando rigidez e precisão em movimentos lineares. Esses robôs são amplamente utilizados em tarefas de transporte e manipulação de objetos, bem como em aplicações médicas.
5) **Robôs SCARA**: Conhecidos por sua alta precisão e velocidade, possuem 3 juntas rotativas e uma linear, sendo amplamente utilizados em operações de montagem e *pick and place* na indústria eletrônica e automobilística.

B. Modelos de Detecção baseados em Aprendizado Profundo

As redes neurais convolucionais (CNNs) são amplamente utilizadas em tarefas de *VC* devido à sua capacidade de extrair características relevantes das imagens. Entre os modelos de detecção de objetos mais utilizados estão:

1) **RCNN**: Introduziu a ideia de utilizar propostas de regiões para melhorar a precisão da detecção. O modelo RCNN divide a imagem em diversas regiões propostas e, em seguida, aplica uma CNN para classificar cada uma dessas regiões. Essa abordagem permite uma detecção mais precisa, pois cada região é analisada individualmente, resultando em uma maior precisão na localização dos objetos. No entanto, o processo de geração de propostas de regiões e a subsequente classificação podem ser computacionalmente intensivos [4].
2) **SSD**: Oferece uma abordagem de detecção em uma única etapa, combinando a detecção e a classificação de objetos em um único processo. O SSD utiliza múltiplas camadas convolucionais para detectar objetos de diferentes tamanhos e proporções, realizando a detecção e a classificação em um único passo. Isso resulta em um modelo mais rápido e eficiente em comparação com abordagens que utilizam múltiplas etapas, como o RCNN. O SSD é particularmente eficaz em cenários onde a velocidade é crucial, mantendo uma boa precisão na detecção de objetos [5].
3) **YOLO**: Conhecido por sua velocidade e eficiência, realiza detecção e classificação de objetos em tempo real. O YOLO trata a detecção de objetos como um problema de regressão único, dividindo a imagem em uma grade e atribuindo uma caixa delimitadora e uma classe a cada célula da grade. Essa abordagem permite que o modelo processe uma imagem inteira em uma única passagem, resultando em uma detecção extremamente rápida. O YOLO é amplamente utilizado em aplicações que requerem processamento em tempo real devido à sua capacidade de fornecer resultados rápidos sem comprometer significativamente a precisão [6].

III. Trabalhos Relacionados

A. Robotic Arm Guided by Deep Neural Networks and New Knowledge-Based Edge Detector for Pick and Place Applications

Este trabalho fez o uso de um braço robótico industrial com cinco graus de liberdade e uma garra impressa em 3D para realizar operações de *pick and place*. O objetivo foi desenvolver um método para a detecção de bordas de objetos, permitindo a extração precisa do centro de massa dos objetos [7].

O sistema utilizou uma câmera para capturar imagens, processadas por redes neurais profundas (*DNN*) como AlexNet, GoogLeNet e ResNet50. As imagens foram divididas em conjuntos de treinamento, validação e teste, e a detecção de bordas analisou a continuidade entre *pixels* candidatos e de vizinhança [7].

Os resultados mostraram que as redes *DNNs* aprenderam eficientemente, com *datasets* de imagens de bordas sendo mais eficazes na determinação do centro de massa para *pick and place* comparado a imagens brutas [7].

B. Six-Axis Robotic Arm Integration with Computer Vision for Autonomous Object Detection using TensorFlow

Este estudo desenvolveu um braço robótico com seis graus de liberdade, projetado no *SolidWorks* e composto por cinco componentes principais: base, braço inferior, braço superior, braço intermediário e garra. O objetivo foi criar um sistema de controle baseado em *Arduino* para ajustar os movimentos em tempo real utilizando feedback de sensores, visando aplicações industriais.

Para a detecção de objetos, o sistema utilizou uma câmera para capturar imagens, analisadas por algoritmos de detecção do *TensorFlow*. O processo incluiu a rotulação dos objetos, geração de arquivos *TFRecord*, configuração e treinamento do modelo, captura de quadros pela webcam, execução da detecção e transmissão serial das coordenadas dos objetos ao microcontrolador.

O sistema alcançou mais de 90% de acurácia após 2000 ciclos de treinamento e teste, realizado contra um fundo verde para minimizar ruído. O *TensorBoard* foi utilizado para monitorar e visualizar o processo de treinamento, aumentando a eficácia das ferramentas de aprendizado de máquina empregadas.

C. Grasping and Positioning Tasks for Selective Compliant Articulated Robotic Arm using Object Detection and Localization: Preliminary Results

Este estudo utilizou um robô SCARA, modelo GLOBOT KSS-1500, equipado com uma garra de dois dedos para realizar tarefas de *pick and place*. O objetivo foi melhorar a eficácia na detecção de bordas de objetos para facilitar a identificação e manipulação precisa de objetos cilíndricos [8].

Para a detecção de bordas, foi instalada uma câmera Flir Point Grey Chamaleon 1.3 modelo CM3-U3-12S2C-CS-SET na flange do robô, alinhada paralelamente ao eixo Z do manipulador. A câmera, compatível com o software *Matlab*, suportou múltiplos formatos de imagem, facilitando o processamento em tempo real. A calibração da câmera foi realizada utilizando 40 imagens de um tabuleiro de calibração assimétrico, corrigindo a distorção da lente angular. Diversas técnicas de detecção de bordas, incluindo Sobel, Prewitt, Roberts, Log, Zero-cross, e Canny, foram comparadas para determinar a mais eficaz [8].

Os modelos Log, Zero-cross e Canny mostraram-se superiores, alcançando 100% de precisão o qual indica , *recall* definindo e *F-measure*(). O robô foi programado para identificar objetos cilíndricos, mover-se para um ponto fixo de observação da câmera, detectar o objeto e medir a posição do centro do objeto em relação à coordenada de referência *World*. A posição foi ajustada às coordenadas do robô, que então realizava a operação de *pick and place*. O método permitiu estimar o erro médio de posicionamento após várias amostragens, demonstrando a precisão e eficácia do sistema SCARA em operações de montagem guiadas por *VC* [8].

D. Research on Robotic Arm Based on YOLO

Este estudo utilizou a rede YOLOv5 para detecção de objetos em linhas de montagem, com o objetivo de facilitar o envio de informações sobre os alvos e suas localizações para manipuladores robóticos. A rede YOLOv5 trata a classificação de objetos como um problema de regressão, utilizando o erro quadrático médio da função de perda [9].

O treinamento da rede utilizou 100 imagens para treino, 10 para teste e 10 para validação do modelo de aplicação, com a rotulagem realizada por um software específico. O hardware empregado incluiu uma CPU-i5-8500 e uma GPU RTX3070, com o treinamento ocorrendo ao longo de 100 épocas e um tamanho de *batch* igual a 4. A função de perda total incluiu o erro das coordenadas centrais do *bounding box*, o erro de altura e largura do *bounding box*, o erro de confiança no alvo detectado e o erro de classificação da célula que contém o objeto [9]. O grau de confiança é medido como a probabilidade de que o *bounding box* contenha um objeto e que esse objeto detectado é da classe prevista. Esta medida varia de 0 a 1, com 1 indicando a total confiança na detecção.

A estabilidade do modelo foi analisada sob dois cenários distintos: um com *background* similar ao conjunto de dados de treinamento e outro com um *background* diferente. O modelo com *background* similar obteve um grau de confiança de 0.82 e um tempo médio de detecção de 26.6 ms, detectando todos os alvos na imagem. Em contraste, o modelo com *background* diferente teve um tempo de detecção médio de 37.1 ms e um grau de confiança de 0.79, com dois erros de amostra e oito detecções corretas. Para cenários onde não há *targets* específicos, múltiplos erros foram observados, indicando a necessidade de aprimoramento do modelo para melhorar sua capacidade de generalização, especialmente em ambientes não treinados [9].

E. Robotic Grasping Position of Irregular Object Based Yolo Algorithm

O estudo desenvolveu um método utilizando o *framework PyTorch* e uma versão adaptada da rede YOLO para detectar e localizar objetos variados como canetas, borrachas, apontadores, varas, garrafas e brinquedos. O objetivo foi abordar os desafios associados à detecção e posicionamento para apreensão de objetos por robôs [10].

A técnica gera um retângulo circunscrito ao redor do objeto detectado, indicando a posição ideal para a apreensão por uma garra de dois dedos. Além disso, a implementação de uma segunda rede neural, a *Visual Geometry Group*(*VGG16*), permitiu uma melhoria significativa no processo. Esta rede, com 13 camadas convolucionais ativadas por *LeakyReLU*, 4 camadas de *max pooling* e duas camadas totalmente conectadas com

funções de ativação sigmoidais, proporciona a regressão das quatro coordenadas (x1, x2, y1, y2) que definem a posição de apreensão. Para evitar o *overfitting*, foram aplicadas técnicas de *dropout* e ajustado o tamanho do *batch-size* nos dados de treinamento [10].

Os resultados demonstraram que a abordagem é mais eficaz e robusta na detecção da posição de apreensão em objetos de formas irregulares em comparação com métodos clássicos, que processam a imagem para determinar o centroide do menor retângulo circunscrito ao objeto. Esta técnica proporcionou uma detecção mais precisa e eficiente para operações de apreensão robótica [10].

F. Research on Moving Arm Grasping Based on Computer Vision

O estudo abordou a adaptação de manipuladores robóticos industriais a ambientes externos, visando aumentar a flexibilidade e eficiência na execução de múltiplas tarefas. Foi identificada uma limitação nos braços robóticos que dependem exclusivamente da memória programada e da interação humano-máquina para aprender posições, resultando em menor eficiência e flexibilidade no posicionamento de objetos [11].

Para melhorar a interação entre o ambiente externo e o manipulador, foi proposta a utilização da técnica *Canny Edge Detection* para eliminar informações indesejadas e obter um *bounding box* mínimo ao redor dos objetos de interesse. Redes neurais convolucionais (CNNs) combinadas com a rede YOLOv3 foram usadas para localizar objetos. A implementação utilizou uma câmera de resolução 640x480 *pixels*, fixada em relação ao plano de observação e às flanges do braço robótico. O algoritmo YOLOv3 refinou o *bounding box*, melhorando a acurácia da rede e reduzindo ruídos de fundo. O algoritmo de detecção de bordas de Canny aplicou um filtro gaussiano para remover ruídos, calcular a direção e magnitude do gradiente, e implementar a supressão não máxima para melhorar os contornos dos objetos [11].

Após determinar os contornos dos objetos, o contorno principal foi selecionado com base no maior número de coordenadas, seguido pelo algoritmo de *Sklansky* para modelar a geometria do objeto. As informações foram transmitidas como tópicos no sistema *Robot Operating System* (*ROS*). A movimentação do robô foi ajustada para posicionar a garra no centro da imagem, alinhando a abscissa do objeto detectado no centro da garra e posicionando-o diretamente abaixo do centro do braço. A inclinação do objeto foi informada via *subscribe*, permitindo que a quinta junta do robô se movesse em um ângulo de 90 graus em relação à inclinação do alvo, facilitando a captura do objeto pelas demais juntas do manipulador [11].

G. Towards automatic generation of imagem recognition models for industrial robot arms

O estudo abordou a necessidade de soluções adaptativas nos processos de manufatura devido à crescente customização e frequentes mudanças que impactam a programação dos manipuladores robóticos. Para lidar com a demanda por dados volumosos e precisamente rotulados, o estudo propôs o uso de *Cycle GANs* (Generative Adversarial Networks) para gerar dados sintéticos realistas a partir de imagens tridimensionais e processá-las usando a rede neural YOLOv5 para reconhecimento de imagens [12].

O processo envolveu a conversão de modelos 3D para o formato *STL* e o desenvolvimento de um script em Python para importar essas imagens e rotacioná-las, gerando novas imagens sob diferentes perspectivas do mesmo objeto. Para aumentar o realismo, foi criado um sistema de translação que alterava a aparência das imagens para que se assemelhassem a peças tridimensionais reais. Fundos com filtros foram adicionados para evitar que as redes focassem mais nos fundos brancos das imagens geradas. Um rotulador automático melhorou a qualidade do treinamento, gerando um conjunto de 12.000 imagens a partir de 4.000 imagens relacionadas a cada peça 3D, treinadas ao longo de 150 épocas [12].

O reconhecimento de imagem foi realizado com o auxílio de uma câmera *RealSense D40*, acoplada na flange do robô *MZ07*, que possui seis graus de liberdade. A geração de dados sintéticos por meio das *Cycle GANs* melhorou significativamente o reconhecimento de imagem, tornando as simulações mais realistas e aplicáveis às tecnologias da indústria 4.0. A utilização de redes neurais rápidas permitiu a localização de objetos em tempo real, aumentando a flexibilidade e adaptabilidade das configurações robóticas e reduzindo a necessidade de intervenção humana na movimentação dos braços robóticos [12].

IV. Materiais e Métodos

Esse tópico é destinado a demonstrar os materiais e métodos utilizados para compor o diagrama da figura 1, que representa o sistema de controle de ponta-a-ponta.

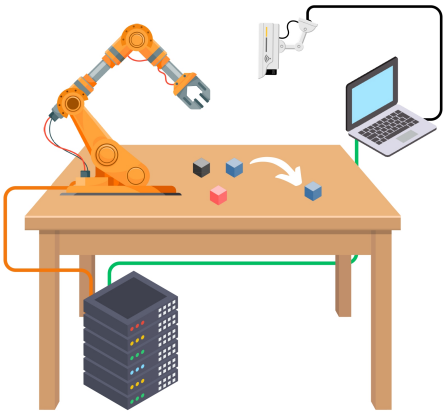

Fig. 1: Relação entre cada componente do sistema

A. Objetos de Movimentação

Os blocos de plástico impressos em 3D, figura 2, nas cores branca, preta e cinza, com dimensões de 2.6 cm x 2.6 cm x 5.5 cm, foram os objetos empregados como modelos físicos dos objetos manipulados durante as atividades de teste e desenvolvimento do sistema robótico.

Fig. 2: Blocos utilizados como material a ser movimentado

B. Mesa de trabalho

A mesa de trabalho, feita de madeira e com dimensões de 185 cm x 185 cm x 83 cm, proporciona uma superfície plana para a movimentação tanto dos blocos quanto do robô durante as atividades de manipulação.

C. Robô

O manipulador robótico utilizado é do modelo *Staubli TS60 FL 200*, conforme demonstrado na figura 3, composto por 4 graus de liberdade, posicionado sobre a mesa de trabalho, é equipado com uma garra pneumática, presente na figura 4, impressa em 3D acoplada em sua flange. Essa garra é capaz de realizar o movimento de pinça comandada pelo controlador *CS8C*, que recebe comandos de movimento do robô e controle da garra. Esse controlador utiliza a linguagem *VAL3* para a elaboração de programas de movimento e controle do robô, estando conectado a um *PC* para execução do algoritmo de controle.

Fig. 3: Robô Staubli TS60

Fig. 4: Garra pneumática

D. Câmera

A captura de imagens é realizada pela câmera genérica, com resolução de 1920x1080 *pixels*, está fixada por uma haste acima do robô, a 1.04m acima da mesa de trabalho, e é responsável por capturar as imagens e transmiti-las ao PC para processamento.

E. Modelo de detecção de objetos

Para a detecção de objetos, foi utilizado o avançado algoritmo YOLO em sua oitava versão nos modelos *YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x*, que, ao processar a imagem em uma única passagem, prevê diretamente as caixas delimitadoras e classes dos objetos, destacando-se por sua eficiência e precisão. O modelo foi treinado com um *dataset* próprio, construído a partir de 43 imagens capturadas pela câmera, posicionada por uma haste fixa acima do robô e a 1.04[m] acima da mesa de trabalho, e rotuladas manualmente utilizando o Roboflow. O treinamento do modelo foi realizado com os seguintes parâmetros: 200 épocas, 43 imagens e 3 classes (Bloco branco, Bloco preto, Bloco cinza). Essas imagens foram usadas apenas para a construção do modelo durante a fase de treinamento. Para a realização da inferência, novas fotos eram tiradas a cada inferência, garantindo que o modelo tivesse informações atualizadas e precisas sobre a cena atual.

F. Mesa de trabalho virtual

Foi desenvolvida uma réplica virtual da mesa de trabalho, utilizando o software de simulação RoboDK, com o objetivo de fazer deste um gêmeo digital do ambiente real. A mesa virtual possui as mesmas dimensões da mesa real e é composta por um plano de trabalho, blocos virtuais, câmera virtual e robô virtual, que são modelados de acordo com as especificações dos componentes reais.

G. Algoritmo de controle

O algoritmo de controle desenvolvido em Python desempenha o papel fundamental de processar as imagens capturadas pela câmera, realizar a identificação dos objetos presentes na cena e tomar decisões de movimento para o robô. Para facilitar a execução do algoritmo, foram criadas algumas classes para separar as lógicas:

- **cam**: Controle da câmera, realiza ações como abertura da câmera e captura da imagem.
- **frame**: Atua sobre os *bounding boxes*, realizando ações como inferência utilizando o YOLOv8, cálculo de distâncias entre dois pontos e centro de *bounding box*, cálculo da proporcionalidade entre milímetros e pixels.
- **network**: Estabelece uma conexão socket com o controlador CS8C e envia mensagens a ele.

H. VAL3

Para o controle do robô, junto do controlador CS8C, foi utilizado um script escrito em VAL3, garantindo a coordenação precisa das tarefas automatizadas. VAL3 é a linguagem de programação específica utilizada para controlar os robôs da Stäubli, incluindo seus manipuladores e controladores, como o CS8C. Esta linguagem foi projetada para oferecer um ambiente de programação flexível e eficiente, permitindo aos usuários desenvolver aplicativos complexos de automação robótica.

A estrutura do VAL3 é baseada em scripts que descrevem sequências de movimentos e operações de controle, facilitando a integração com diversos sensores e dispositivos de entrada/saída.

V. Desenvolvimento

Este trabalho propõe uma maneira de controlar um sistema de forma autônoma, utilizando visão computacional para a identificação de objetos e a tomada de decisões. O sistema pode ser dividido em três componentes principais: o sistema de visão computacional, o sistema de controle e o sistema de manipulação. Esta seção descreve detalhadamente os componentes do sistema e os algoritmos utilizados em cada um deles.

A. Sistema de visão computacional

Foi usado o algoritmo YOLOv8, um modelo de detecção de objetos treinado para identificar blocos dimensões de 2.6 cm x 2.6 cm x 5.5 cm em uma mesa de trabalho.

O modelo foi treinado a partir de um conjunto de dados próprio, construído a partir de 43 imagens capturadas pela câmera a qual foi posicionada por uma haste fixa acima do robô e a 1.04 m acima da mesa de trabalho e rotuladas manualmente utilizando o *Roboflow*.

O modelo foi treinado com 200 épocas e 3 classes (Bloco branco, Bloco preto, Bloco cinza), sendo que dentre as 200 épocas, é sempre selecionado o arquivo em que o modelo apresentou melhor resultado.

Além disso, um segundo modelo foi treinado com para detectar a mesa branca usada como plano de referência e estimar métricas. Esse modelo foi composto por 49 imagens e apenas uma classe.

Ambos os conjuntos de dados foram criados pelos próprios autores e estão disponíveis para acesso público em *universe.roboflow.com/tcc-qhnu9/mesa-djgi1*.

B. Sistema de controle

Este trabalho busca propor uma maneira de controlar um sistema de forma autônoma, utilizando visão computacional para a identificação de objetos e a tomada de decisões.

O sistema pode ser dividido em três componentes principais: o sistema de visão computacional, o sistema de controle e o sistema de manipulação.

Esta seção descreve detalhadamente os componentes do sistema e os algoritmos utilizados em cada um deles.

O sistema de controle é composto por um *script* responsável por processar as imagens capturadas pela câmera, realizar a localização dos objetos presentes na cena e tomar decisões de movimento para o robô.

O algoritmo de controle foi desenvolvido em *Python* e desempenha o papel fundamental de processar as imagens capturadas pela câmera. O código está disponível em *github.com/guimroque/staubli-control*.

A partir da identificação dos objetos presentes na cena, utilizando a função de inferência do YOLOv8, o algoritmo identifica a posição dos blocos, seguindo a seguinte lógica:

Para realizar o processo de manipulação, o sistema começa com a identificação dos blocos presentes na cena. Nesta etapa, são retornadas as coordenadas em *pixels* de cada uma das arestas das caixas delimitadoras dos blocos. Em seguida, realiza-se o cálculo das coordenadas dos centros dos blocos, utilizando as coordenadas das arestas para determinar o centro de cada bloco.

Após essa etapa, é criada uma coordenada de referência (C.R) a partir do centro da mesa de trabalho, estabelecendo assim o centro da mesa. Com isso, calcula-se as distâncias dos centros dos blocos em relação à C.R. Esse cálculo é feito a partir do centro da mesa, determinando a distância de cada bloco em coordenadas lineares *(x, y)*.

Para calcular o centro de cada bloco e da mesa, as coordenadas das arestas fornecidas pelo YOLOv8 são utilizadas. Essas coordenadas são geralmente dadas como (x_min, y_min) para o canto superior esquerdo e (x_max, y_max) para o canto inferior direito da caixa delimitadora. O centro do bloco (x_{center}, y_{center}) pode ser calculado usando a fórmula:

$$x_{center} = \frac{x_{min} + x_{max}}{2}$$

$$y_{center} = \frac{y_{min} + y_{max}}{2}$$

Conhecendo o tamanho físico da mesa de trabalho, é possível converter essas distâncias para milímetros, fornecendo a distância precisa de cada bloco em relação à C.R. Com as distâncias calculadas, o algoritmo envia as instruções de movimento para o robô. Essas instruções são então executadas pelo robô, que se encarrega de posicionar os blocos conforme as coordenadas recebidas.

Dessa forma, o sistema consegue identificar os blocos, calcular suas posições relativas ao centro da mesa, e mover os blocos para as posições desejadas de maneira eficiente e precisa.

C. Sistema de manipulação

O sistema de manipulação é composto pelo Staubli, uma garra pneumática e um controlador CS8C.

Conforme apresentado na figura 5, o controlador foi conectado ao sistema por meio do protocolo de comunicação *socket*, Para isso, deve-se conectar um cabo com extremidade RJ45 no controlador e a outra extremidade USB-C no PC. Feito isso, basta configurar um endereço IP e uma porta, para que o sistema de controle possa se comunicar com o controlador. Para enviar os comandos ao controlador, foi desenvolvido um algoritmo em Val3 [13]. Algumas variáveis foram criadas para o controle do sistema, as mais relevantes são:

- **home**: do tipo Point, que representa a posição equivalente a origem do sistema de coordenadas (centro da mesa de trabalho).
- **photo**: do tipo Point, uma pose que indica onde o robô deve estar, quando a foto é tirada.
- **cradle appr**: do tipo Point[], que representa as posições conhecidas para onde os blocos devem ser movidos.
- **input**: do tipo sio, responsável por receber as informações via socket no formato de bytes.
- **msg**: do tipo num[], responsável por armazenar as informações recebidas pela variável input.
- **garra**: do tipo gripper, responsável por controlar a garra pneumática.

Após iniciada a execução, o robô se posiciona no ponto photo, e o controlador fica aguardando as instruções de movimento do sistema de controle, são esperadas mensagens no formato:

S	X	xxx	Y	yyy	D	dd	E

Fig. 5: Formato da mensagem esperada pelo controlador

- **S**: Representa o início de uma instrução válida, se valer algo diferente de "G", o controlador ignora a instrução.
- **X**: Indica que os próximos caracteres são referentes à coordenada X.
- **xxx**: A coordenada destino no eixo X, onde X é a distância em milímetros.
- **Y**: Indica que os próximos caracteres são referentes à coordenada Y.
- **yyy**: A coordenada destino no eixo Y, onde Y é a distância em milímetros.
- **D**: Indica que os próximos caracteres são referentes ao índice do array de posições conhecidas.
- **dd**: O índice do array de posições conhecidas, onde D é um número inteiro, e o valor da posição representa para onde o bloco vai ser movido depois de capturado.
- **E**: Representa o fim de uma instrução válida.

Após identificada uma instrução válida, o controlador a executa, como explicado no fluxograma presente na figura 6:

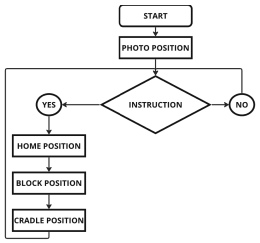

Fig. 6: Fluxograma de execução do script em val3

VI. Experimentos e Resultados

Esta seção analisa a performance dos modelos de detecção com o objetivo de identificar o melhor modelo para o sistema proposto. As métricas de detecção foram calculadas utilizando a biblioteca *scikit-learn*, baseando-se nas matrizes de confusão de cada modelo durante o processo de validação. Para determinar o modelo com o comportamento mais adequado para um sistema real, também foi considerado o tempo de inferência para cada instância em todos os modelos.

A aplicação desenvolvida visa fazer o manipulador robótico capturar o objeto de interesse detectado pela câmera. Isso envolve a obtenção da posição de captura do objeto e o envio das coordenadas para o controlador, que realiza os movimentos necessários. A rede neural convolucional YOLOv8 mostrou-se eficaz na extração da localização e classificação das peças e da mesa usada como sistema de referência.

Para avaliar qual modelo YOLOv8 é o mais eficaz para o ambiente proposto, foram calculadas diversas métricas de desempenho para diferentes modelos da oitava versão do YOLO. Essas métricas incluíram a acurácia, precisão, recall e f1-score, baseando-se nas matrizes de confusão de validação. Além disso, foram desconsideradas as perdas do *background* presentes nessas matrizes para uma avaliação mais precisa. O tempo de inferência foi outra métrica crucial, pois um modelo mais rápido é essencial para aplicações em tempo real.

A tabela 1 apresenta a relação entre os modelos avaliados, mostrando as métricas de desempenho e o tempo de inferência para cada um. Esses dados são fundamentais para escolher o modelo mais adequado para a aplicação proposta, balanceando precisão e eficiência de processamento.

TABELA I: Métricas de desempenho dos modelos YOLOv8.

Modelo	Acurácia	Precisão	Recall	F1-Score
Yolov8n	0.933	0.933	0.933	0.933
Yolov8s	0.903	0.902	0.903	0.901
Yolov8m	0.935	0.942	0.935	0.934
Yolov8l	0.967	0.971	0.967	0.967
Yolov8x	0.933	0.933	0.933	0.933

Além disso, foram registrados os tempos de inferência de cada um dos modelos para peças de cores pretas, cinzas e brancas, sob as mesmas condições de altura de câmera e iluminação. O código de inferência foi executado a partir de um computador com 16 GB de RAM e processador *Apple M1 Pro*. O tempo de inferência foi observado a partir de ensaios em um cenário real posicionando individualmente cada peça em uma mesma posição na mesa branca e executando o código de detecção para cada modelo.

TABELA II: Tempos de inferência dos modelos (ms).

Cor da Peça	Yolov8n	Yolov8s	Yolov8m	Yolov8l	Yolov8x
Peça preta	40.1	76.1	138	207	280
Peça branca	-	105.5	119	180	263.4
Peça cinza	-	75.3	153.9	185.5	270.7

Ademais, observou-se também o gráfico de desempenho do modelo presente na figura 7 YOLOv8l durante a etapa de treinamento, indicando que esse modelo aprendeu e melhorou seu desempenho ao longo de 200 épocas de treinamento. A queda nas linhas de perda e o aumento nas linhas de precisão, *recall* e *mAP* demonstraram que o modelo tornou-se mais preciso na identificação e classificação de objetos devido ao menor número de falsos positivos. Através das curvas de *metrics/precision(B)* e *metrics/recall(B)* observou-se que o YOLOv8l se tornou mais preciso na identificação de objetos corretos e menos propenso a cometer erros. Ainda, observou-se uma tendência de queda nas linhas de perdas de treino e validação.

Devido ao decrescimento no erro de localização *Box loss* das*bounding boxes* dos objetos detectados e também no erro de classificação *cls loss* foi possível notar que o modelo foi capaz de classificar e localizar objetos dentro dessa quantidade de épocas. Com a perda *DFL loss* tendendo a valores próximos de zero demonstrou-se que o modelo está se tornando mais eficaz em discriminar objetos reais de falsos positivos.

VII. Conclusão e trabalhos futuros

Este estudo atingiu seu principal objetivo de integrar a visão computacional para a execução de tarefas de *pick and place* utilizando um robô SCARA TS 60 FL. Com base nas métricas calculadas e no tempo de inferência analisado, concluiu-se que o modelo de detecção YOLOv8l apresentou resultados positivos devido à sua alta acurácia, precisão, recall e f1-score, embora não seja o modelo mais rápido, com um tempo de inferência de 190.83 ms para classificar e localizar peças pretas.

Além disso, apesar das métricas satisfatórias do modelo YOLOv8n em termos de acurácia, precisão, recall e f1-score serem idênticas ao modelo YOLOv8x, o modelo YOLOv8n apresentou perdas na localização dos objetos, uma vez que apresentou dificuldades de detectar experimentalmente peças brancas e cinzas devido ao baixo contraste entre as cores da mesa e as cores da peça. As métricas de ambas versões são semelhantes pois foram calculadas a partir de um conjunto de dados de validação enquanto o tempo de inferência foi estimado a partir ensaios em um cenário real.

Com relação à conversão de pixels em milímetros, o modelo de detecção da mesa branca apresentou algumas perdas na localização devido à sua curvatura irregular e à distorção da lente da câmera, provocando uma irregularidade entre as caixas delimitadoras e o objeto real. Por consequência, as imprecisões das caixas delimitadoras implicaram em cálculos errôneos do tamanho do pixel, resultando em coordenadas imprecisas dos objetos.

Entretanto, mesmo com a imprecisão na posição de captura, o manipulador robótico conseguiu capturar as peças de forma satisfatória, atingindo seu principal objetivo, embora de forma irregular e com objetos não rotacionados sobre a mesa. Para futuras aplicações, propõe-se a utilização do modelo orientado *Yolo OBB*, capaz de identificar as rotações da mesa e dos cubos, melhorando a precisão e eficiência do sistema.

Referências

[1] E. Teixeira, B. Araujo, V. Costa, S. Mafra, and F. Figueiredo, "Literature review on ship localization, classification, and detection methods based on optical sensors and neural networks," *Sensors*, vol. 22, no. 18, 2022.

[2] M. H. F. Afonso, E. H. Teixeira, M. R. Cruz., G. P. Aquino, and E. C. Vilas Boas, "Vehicle and plate detection for intelligent transport systems: Performance evaluation of models yolov5 and yolov8," in *2023 IEEE International Conference on Computing (ICOCO)*, 2023, pp. 328–333.

[3] M. W. Spong, *Robot dynamics and control*. São Paulo: Editora Exemplo, 1989.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[7] Y. Çapkan, C. B. Fidan, and H. Altun, "Robotic arm guided by deep neural networks and new knowledge-based edge detector for pick and place applications," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2021, pp. 1–4.

[8] M. Farag, A. N. Abd Ghafar, and M. H. Alsibai, "Grasping and positioning tasks for selective compliant articulated robotic arm using object detection and localization: Preliminary results," in *2019 6th International Conference on Electrical and Electronics Engineering (ICEEE)*, 2019, pp. 284–288.

[9] N. Aggarwal, M. Deshwal, and P. Samant, "A survey on automatic printed circuit board defect detection techniques," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2022, pp. 853–856.

[10] S. Zhang, Z. Guo, J. Huang, W. Ren, and L. Xia, "Robotic grasping position of irregular object based yolo algorithm," in *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, 2020, pp. 642–646.

[11] X. Zhang, J. Xu, H. Fu, and S. Hu, "Research on moving arm grasping based on computer vision," in *2022 5th International Conference on Robotics, Control and Automation Engineering (RCAE)*, 2022, pp. 298–301.

[12] H. Arnarson, B. A. Bremdal, and M. F. Hanssen, "Towards automatic generation of image recognition models for industrial robot arms," in *IEEE Conference on Emerging Technologies and Factory Automation*, 2023, pp. 1–6.

[13] N. Waytowich, A. Henderson, D. Krusienski, and D. Cox, "Robot application of a brain computer interface to staubli tx40 robots - early stages," in *2010 World Automation Congress*, 2010, pp. 1–6.

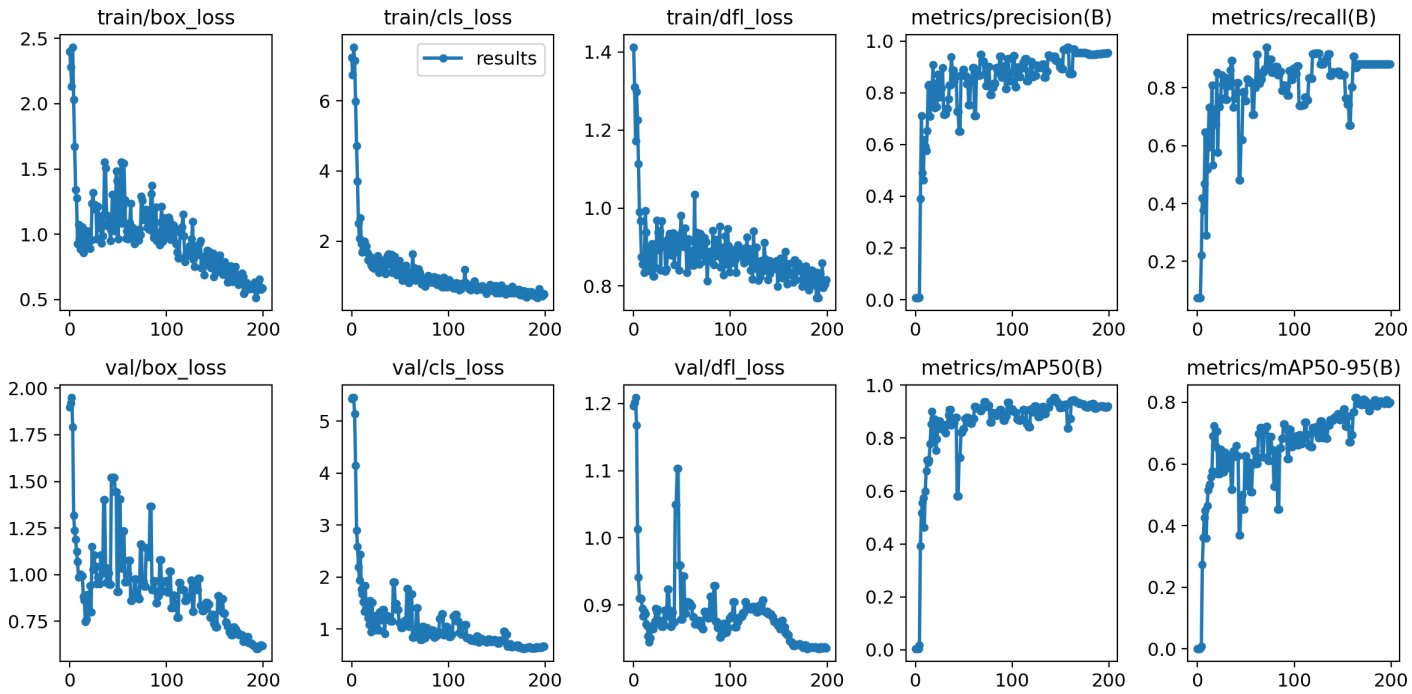

Fig. 7: Progressão de treinamento YOLOv8l

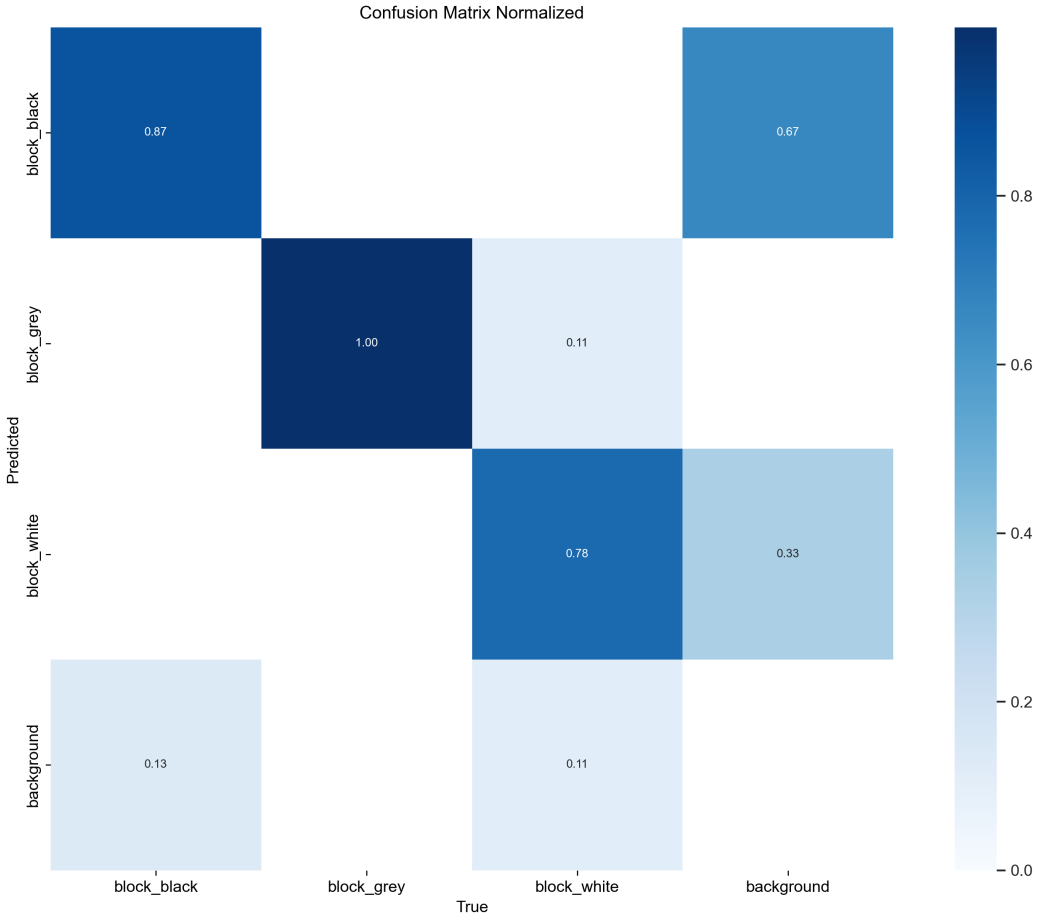

Fig. 8: Matriz de confusão YOLOv8l