Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/108/2025.LD21155

Adaptive Robust Learning Control for a 6-DOF Robotic Arm with Real-Time Object Detection Using YOLO v10

Jiace Zhao[1,a,*]

[1]Tsinghua High School Daoxianghu School, Beijing, China
a. krypoto@outlook.com
**corresponding author*

Abstract: This paper proposes using the YOLO object detection algorithm to accurately and efficiently detect nuts and screws using a custom-built 6-DOF robotic arm equipped with a gripper and a depth camera running the YOLO v10 object detection model in real-time, addressing the challenges of traditional detection methods in fast-paced production environments. This capability enables subsequent robot control to dynamically adjust actions based on precise component localization. Dynamic control in robotic automation has traditionally depended on model-based approaches, where performance depends on accurate system modeling. However, as systems grow more complex, precise modeling becomes increasingly challenging, often leading to suboptimal control. To address these limitations, we propose an Adaptive Robust Learning (ARL) Control algorithm. By integrating a Disturbance Observer (DOB) within the ILC framework, the ARL Control algorithm enhances adaptability and robustness, compensating for real-time disturbances. This work highlights the potential of combining advanced control algorithms with state-of-the-art visual recognition in robotics, paving the way for robust learning solutions in dynamic environments.

Keywords: 6-DOF robotic arm, Adaptive Robust Learning (ARL) Control, YOLO v10, object detection, Iterative Learning Control (ILC), Disturbance Observer (DOB), real-time control, PID control, trajectory tracking, robotics automation, dynamic environments, model training, system robustness, control performance.

1. Introduction

With the increasing demand for automation in manufacturing and assembly lines, the accurate and efficient detection of small components such as nuts and screws has become critical for ensuring quality control and process optimization. Traditional methods for component detection, such as manual inspection or the use of basic computer vision techniques, are often time-consuming, error-prone, and unable to scale in fast-paced production environments. As a result, there is a growing need for advanced detection algorithms capable of real-time performance with high accuracy.

We propose to detect nuts and screws using the YOLO (You Only Look Once) [1] object detection algorithm. YOLO is a fast and efficient object detection algorithm that processes images in one pass, allowing for real-time identification. It can detect and classify multiple objects in a single frame simultaneously. YOLO formulates detection as a regression problem, simultaneously predicting bounding boxes and class probabilities for various objects within an image. This makes it especially suitable for high-throughput industrial applications.

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

Integrating YOLO's real-time detection with robotic control enhances automation by providing precise feedback for grasping nuts and screws. YOLO enables robots to dynamically adjust actions based on accurate component localization, improving precision and flexibility in tasks such as fastening and sorting. Dynamic control for robotic automation and other autonomous systems is traditionally designed and optimized using a model-based approach, where performance relies heavily on accurate system modeling. However, accurately modeling the true dynamics of increasingly complex robotic systems is extremely challenging, often resulting in the automation system operating under non-optimal conditions.

Specifically, the most widely used traditional control method is known as PID control [2]. It takes the tracking error from an autonomous system, such as a robotic arm, drone, or autonomous vehicle, and computes the optimal control action for the system. The PID controller is often designed based on the system model, represented either in state-space form or transfer function form. However, in real-world settings, obtaining models for unknown systems can be difficult and time-consuming. Additionally, inaccurate model-based controller design may not provide satisfactory performance. In light of these challenges, learning-based methods have been developed that use data generated from the actual system instead of relying on the robot system model.

Iterative Learning Control (ILC) is a concept that focuses on enhancing system performance with each repetition of a task by adjusting control inputs based on error information gathered from previous attempts. It's akin to a basketball player practicing free throws, where each attempt provides insights that inform adjustments to their grip and posture for the next shot. Initially introduced in the 1980s, the concept of ILC has expanded significantly. Early contributions by Arimoto and colleagues laid the groundwork for ILC, concentrating on repetitive control tasks aimed at consistently achieving a desired trajectory. Subsequent research has broadened this foundational concept, exploring a variety of ILC algorithms and applying them to areas such as robotic systems and industrial automation. The review [3] on ILC comprehensively covers advances in both the theory and practical implementation of the technique. In specific domain applications, it can be found in soft robotics [4], underactuated arms [5], delta robots [6], mobile manipulators [7], ball screw system [8], micro aerial vehicle [9] and more.

However, one of the key drawbacks of ILC is that it only learns an optimal control strategy for a fixed task, without accounting for changes in operating conditions or external disturbances. In industrial automation systems, this limitation is acceptable because conditions usually remain constant. However, in real-world robotics applications, varying conditions must often be considered. For example, a UAV may frequently encounter strong winds as a disturbance, while a mobile robot may experience changes in loading, road surface conditions, or inclines. In such cases, ILC often fails to provide satisfactory performance. This motivates us to develop a learning control approach that is robust and adaptive to changing conditions and disturbances. In this project, I propose an Adaptive Robust Learning (ARL) Control algorithm to address these challenges. Specifically, a Disturbance Observer (DOB) control [10-12] loop is integrated within the ILC framework, enabling the system to compensate for sudden disturbances and adapt to changing conditions in real-time. The experimental results show that the proposed ARL Control outperforms traditional PID control and standard ILC approaches in terms of accurate tracking.

2. System Architecture and Hardware Configuration

2.1. System Overview

The robot system consists of a 6-DOF robotic arm, a custom-designed gripper, a depth camera, and a controlling computer. The robotic arm is applied for object manipulation in three-dimensional space; its motor control boards and the main controller (ref-core) are controlled using the open-source project by Peng Zhihui [13] and the PCB board from the open-source project EleBoard [14]. This gripper (Figure. 1), installed on the end-effector, serves for grasping objects recognized by the vision system. A depth camera (Gemini Pro) on the end-effector terminal joint, providing real-time visual data, enables object recognition based on the YOLO v10 model [1]. As for the robot arm, it is a 6-DOF one and allows manipulating in three-dimensional space flexibly and precisely. The issued commands from the control computer are sent to the robotic arm. However, its control algorithms by Peng Zhihui [13] are calculated in its own control board, including motor controls, trajectory generation, kinematics, inverse kinematics, etc. The robot uses three 35 stepper motors and three 42 stepper motors, and it uses a CAN bus for communication between motors. The depth camera, "Gemini Pro", is mounted on the end effector.

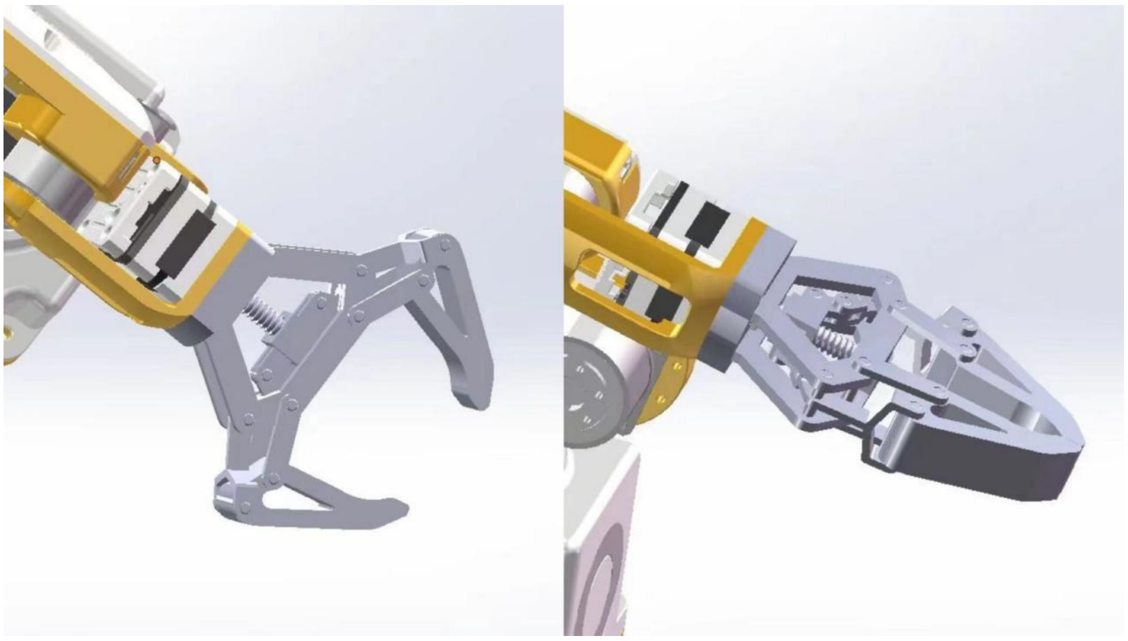
Figure 1: Gripper

The following steps were used to set up the environment:

- Python installation: https://python.org
- YOLOv10 installation: https://github.com/THU-MIG/yolov10
- Camera SDK installation: https://github.com/orbbec/pyorbbecsdk
- ROS driver: https://github.com/orbbec/OrbbecSDK_ROS1
- The camera’s intrinsic parameters were calibrated using the ROS camera calibration package.
- Ubuntu22.04
- LabelImg [15]
- X-AnyLabeling [16]
- YOLO v10(v10m) [17]

2.2. Robotic Arm Architecture

Given that this robotic arm is relatively small in size, it requires a very compact design with high precision and accuracy. While the use of servo motors may lead to the challenge of complex drivers, stepper motors are well-suited for the current project. Stepper motors are not only compact but also have simpler drivers compared to servo motors. However, one significant disadvantage of stepper motors is their inability to generate high torque. Adding harmonic reducers, however, can help overcome this limitation and produce higher torque. Figure 2 illustrates the harmonic reducers in the robotic arm, and Figure 3 shows the overall structure of the robotic arm.

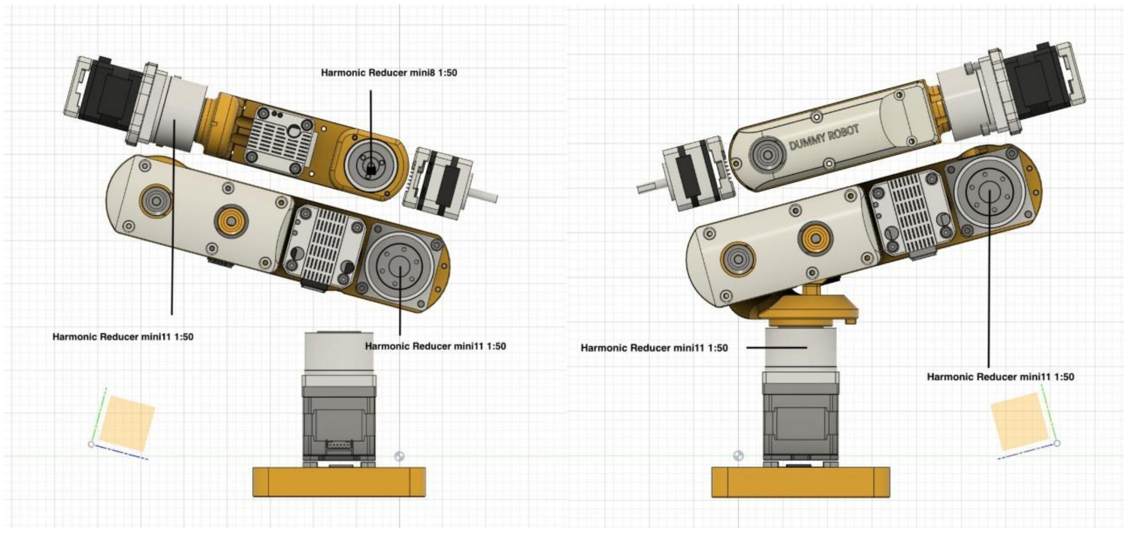
Harmonic Reducer mini8 1:50
Harmonic Reducer mini11 1:50
Harmonic Reducer mini11 1:50
Harmonic Reducer mini11 1:50
Harmonic Reducer mini11 1:50
DUMMY ROBOT

Figure 2: Harmonic Reducer in the Robotic Arm

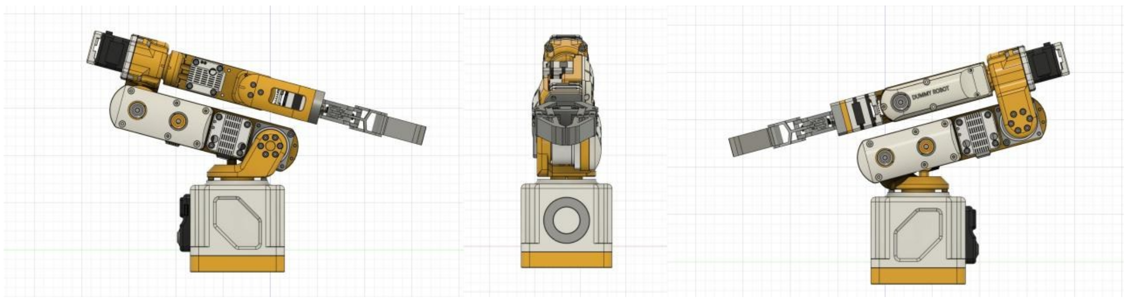

Figure 3: Robotic Arm Architecture in Home Position

3. Visual Recognition and Object Detection

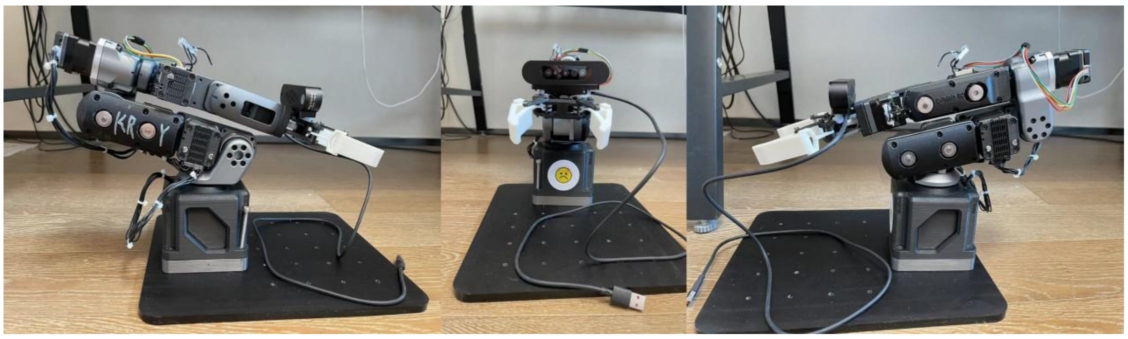

Figure 4: Robotic Arm Architecture in Real World

3.1. YOLO v10 Model Overview

YOLO (You Only Look Once) [17] is a state-of-the-art real-time object detection system that has significantly influenced the field of computer vision and robotics. Unlike traditional object detection methods that utilize a two-stage process, YOLO unifies object detection into a single regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation. This approach enables YOLO to achieve high detection speeds, making it suitable for real-time applications in robotics.[1]

YOLO v10 uses regression to detect objects, transforming the image into a standard input size of 640 × 640 pixels. The image is divided into a grid of 80 × 80 cells, each of which has a size of 8 × 8 pixels. This allows YOLO v10 to process the image efficiently by assigning object detection tasks to each cell based on the object's center falling within that cell. For each grid cell, the model predicts several outputs, represented as a vector:

$$\mathbf{p}_i = \left[x_i, y_i, w_i, h_i, C_i, P_i^1, P_i^2, \dots, P_i^N\right]$$

where x_i, y_i, w_i, h_i are the bounding box parameters, C_i is the confidence score, and P_i^k are the class probabilities for class k, where $k \in \{1,2,\dots,N\}$, with N being the total number of object classes. The total loss function in YOLO v10 consists of three parts:

1. Localization Loss
2. Confidence Loss
3. Classification Loss

3.2. Frame Transformation

The robotic arm is equipped with a depth camera at the end joint. The camera is calibrated to determine its intrinsic and extrinsic parameters. **Intrinsic** parameters include the focal length f_x, f_y, and the optical center c_x, c_y. They form the camera's intrinsic matrix:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Finding intrinsic parameters via depth camera calibration(ros-noetic-camera-calibration) using a chess set. Here's the real Intrinsic Matrix:

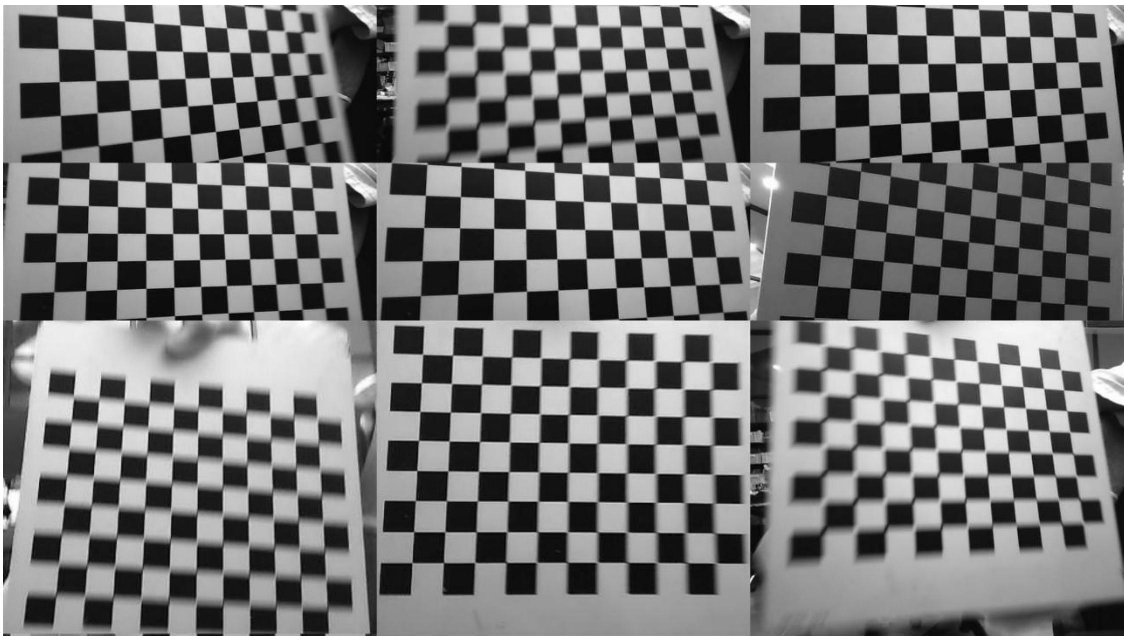

Figure 5: Depth Camera Calibration Using Chess set

$$K = \begin{bmatrix} 459.900903 & 0 & 328.262984 \\ 0 & 460.278379 & 245.203520 \\ 0 & 0 & 1 \end{bmatrix}$$

Here's the distortion coefficient of the camera:

$$D = [0.088048, -0.132798, -0.002091, -0.000172, 0.000000]$$

Extrinsic parameters describe the camera's orientation and position relative to the world frame, represented by a rotation matrix R and a translation vector t. When the camera captures an object, YOLO v10 will detect the 2D bounding box $(x_{min}, y_{min}, x_{max}, y_{max})$. The pixel coordinates of the object's center in the image plane are (u, v), and the depth camera provides the depth Z_c. To convert the 2D pixel coordinates to 3D camera coordinates, we use:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = Z_c \cdot K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

We transform the 3D camera coordinate (X_c, Y_c, Z_c) into **world frame** (robotic arm's frame) by applying the extrinsic transformation.

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = R \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} + t$$

This equation gives the object's coordinates in the robot's reference frame, (X_r, Y_r, Z_r), which can now be used for robotic control.

3.3. Model Training

YOLO don't need a large amount of data in training, since it's pretrained. The data is from [18] and real date show in the Fig.7 shot by robotic arm.

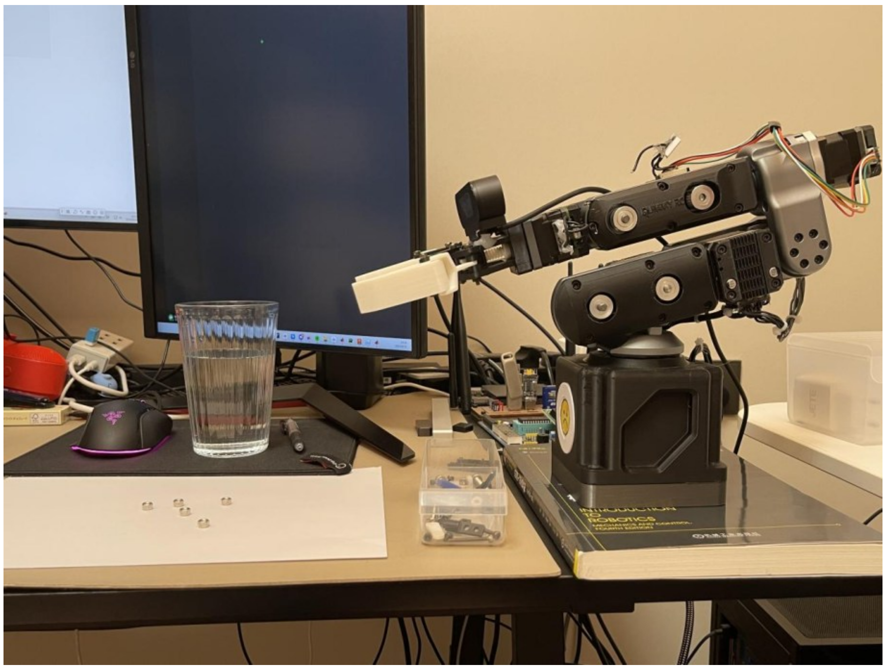

Figure 6: Capturing images using Gemini Pro

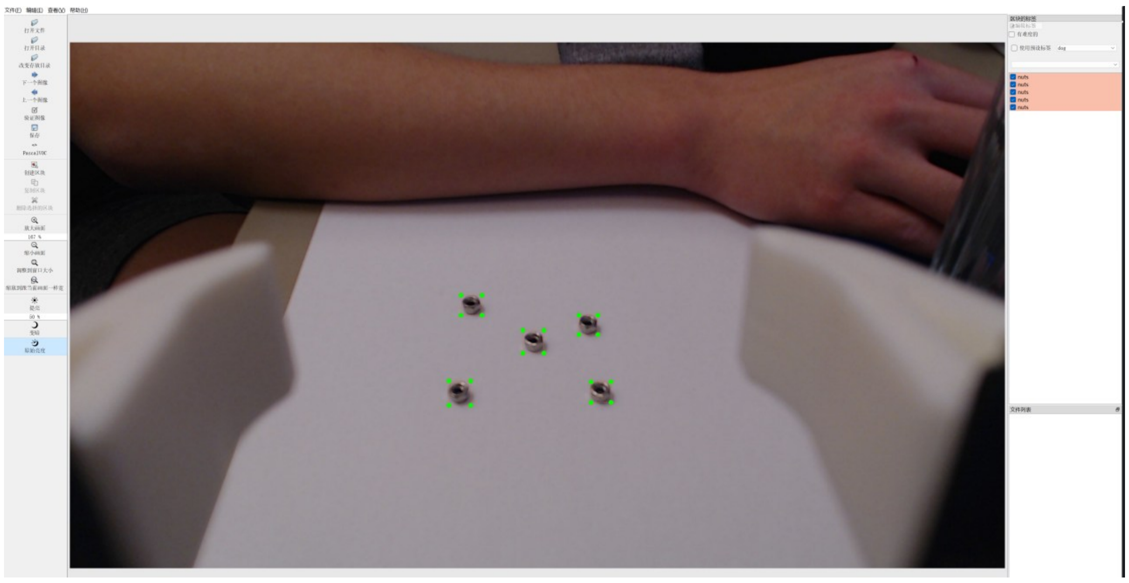

Figure 7: labelImg [15] using real data

After capturing the images, the data needs to be labeled and then provided to the model for training. The data is divided into three parts:

- Data used in the model is from an open-source dataset.
- Val: Data used during training to tune the model and avoid overfitting (after each epoch).
- Test: Data used after training to evaluate how well the model generalizes to unseen data.

Afterwards, labeling the data with LabelImg [15] shown in Figure. 7. In the training process the YOLO v10m model will automatically argument the data as shown in Figure. 8. The coco128.yaml configuration file was modified to reflect the dataset, which included images of nuts and screws.

```
path: /yolov10/datasets/
train: data1/train
val: data1/test
test:

# Classes
names:
  0: nuts
  1: screw
```

```
yolo detect train data=coco128.yaml model=yolov10m.yaml batch=2 epochs=500 imgsz=480 device=0
```

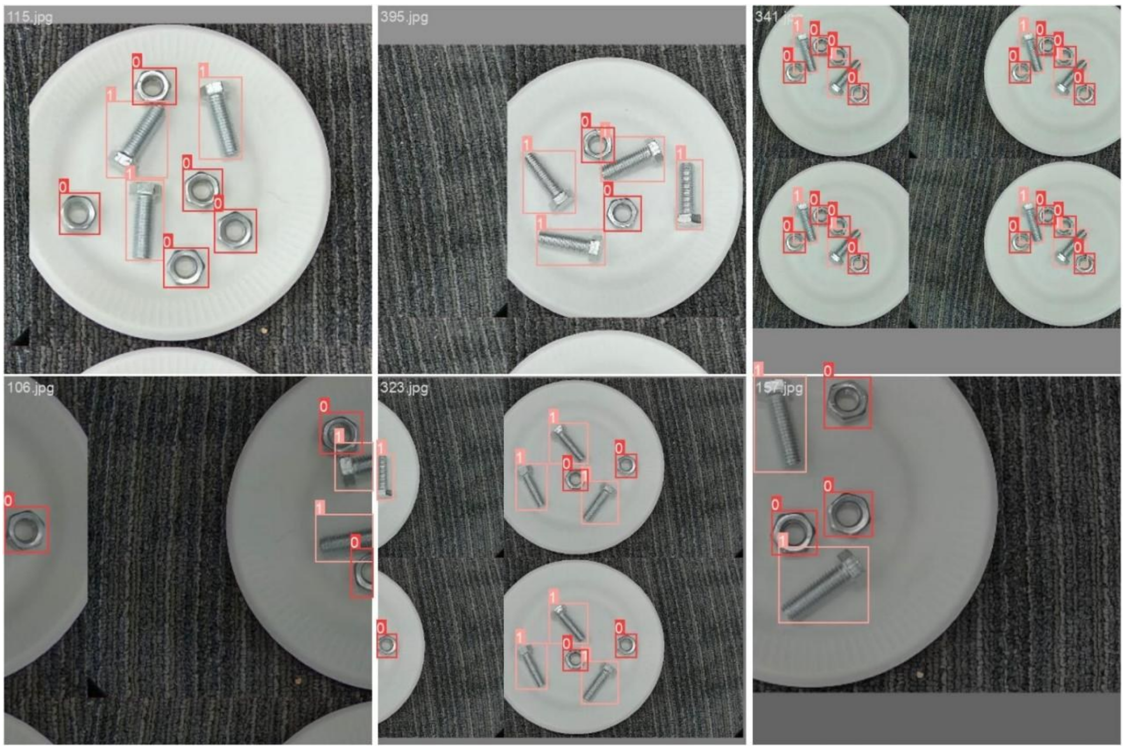

Figure 8: Train Batch in YOLO v10m

The following command was used to start training the YOLOv10 model:

3.4. Model Result

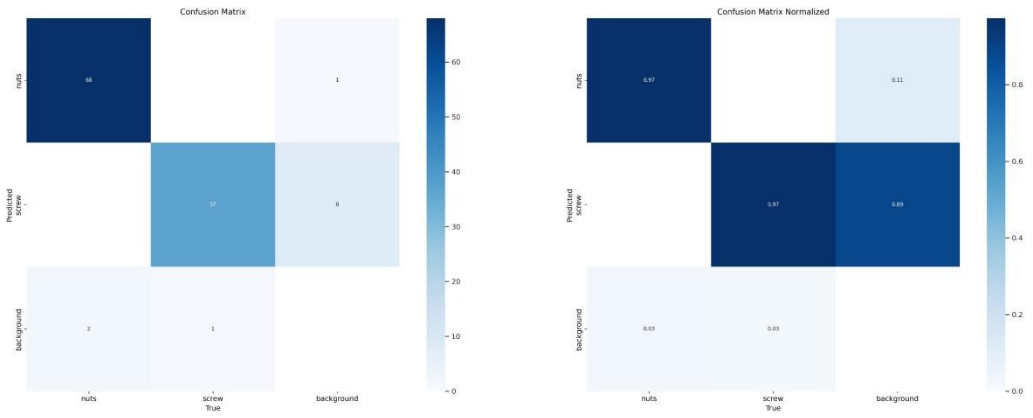

Figure 9: Confusion Matrix

The confusion matrix in Figure. 9 shows how well the model is predicting different object classes (nuts, screws, and background). The matrix shows that nuts and screws are classified with high accuracy 97% for nuts, 97% for screws.

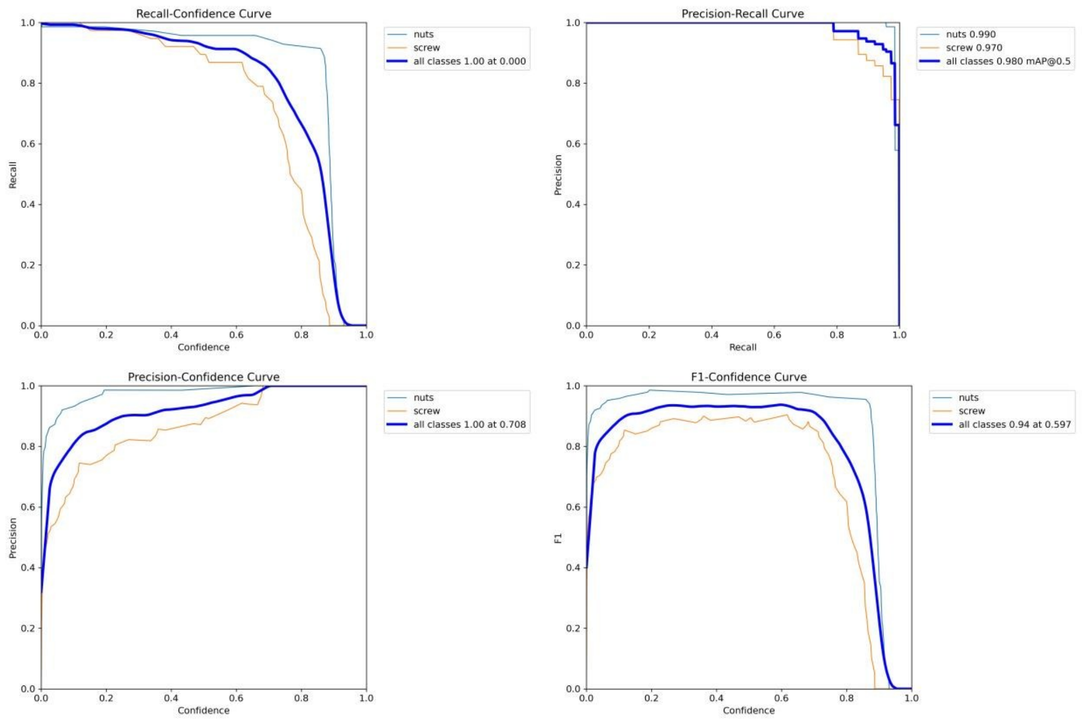

Figure 10: Recall-Confidence, Precision-Recall, Precision-Confidence, F1-Confidence Curves

The high mAP@0.5 of 0.98 means that the model is very reliable when predicting both classes.

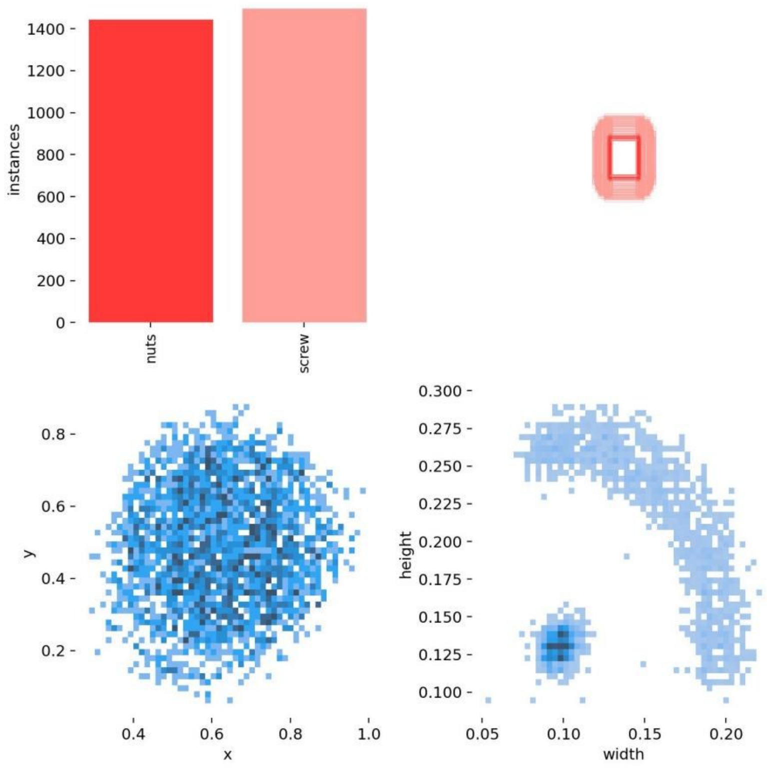

Figure 11: Labels

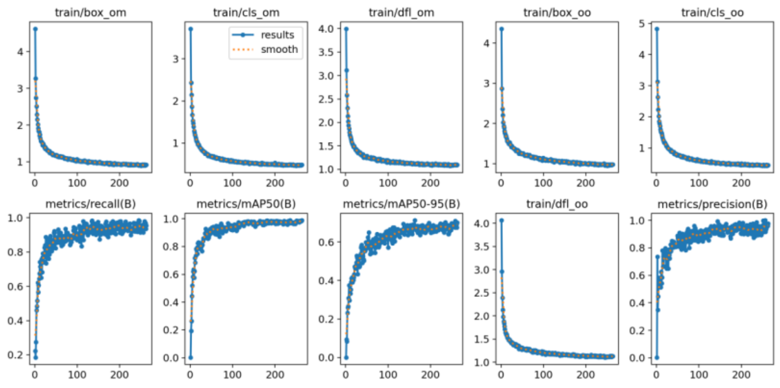

Figure 12: Results

The Figure. 11 shows four graphs. The Figure. 12 shows model's performance improves steadily.

3.5. Middle Ware

The camera(powered by ROS [19] and SDK [20]) is connected to the computer via USB and the robotic arm is connected to computer via a usb-c with serial communication using the CLI-Tool upper SDK written by Peng-zhihui [13]

Returning .pt model file is used in middle ware to label the box of the detected object using opencv. Then do the frame transformation, and output the coordinate of the object to the CLI-Tool (final position + rotation degree of the end-effector.)

3.6. Robotic Arm Control by Peng-zhihui

All the kinematic algorithms are from the open source project [13] designed by Peng-zhihui. After receiving the position of the desired object, the robotic need to reach that point and grasp the object using the glitter. In order to do that the angles of each joint are computed using the inverse kinematics (IK).

Inverse kinematics is the process of calculating the necessary joint angles for the robotic arm to reach a specified position and orientation. For a 6-DOF system like this, the desired end-effector pose consists of six variables (three for position: x, y, and z, and three for orientation: roll, pitch, and yaw). The IK equations solve for the angles $\theta_1, \theta_2, \dots, \theta_6$.

4. Traditional Control

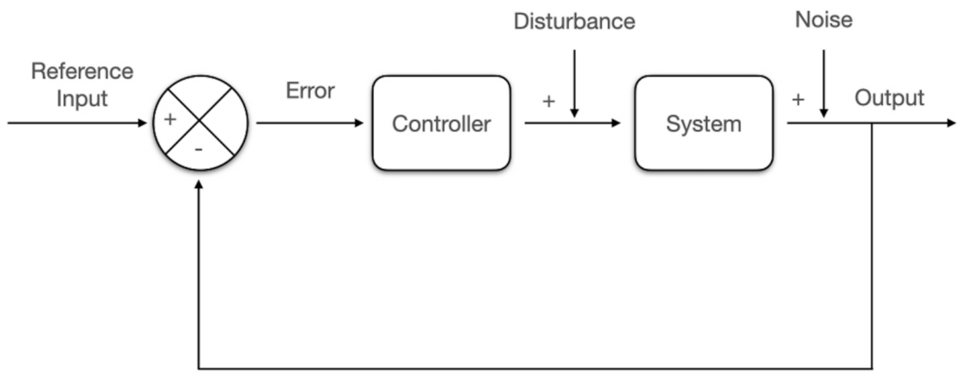

Figure 13: Feedback Control Diagram

Traditional control is based on the idea of modeling dynamic systems using Laplace Transform to derive the Transfer Function. The traditional control system relies on feedback control, a fundamental approach used to maintain the desired performance of a system.

$$u_p(t) = K_P\left(r(t) - y(t)\right) \tag{1}$$

$$u_d(t) = K_D\frac{de(t)}{dt} \tag{2}$$

$$u_i(t) = K_I\int e(t)dt \tag{3}$$

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)\,d\tau + K_d\frac{de(t)}{dt} \tag{4}$$

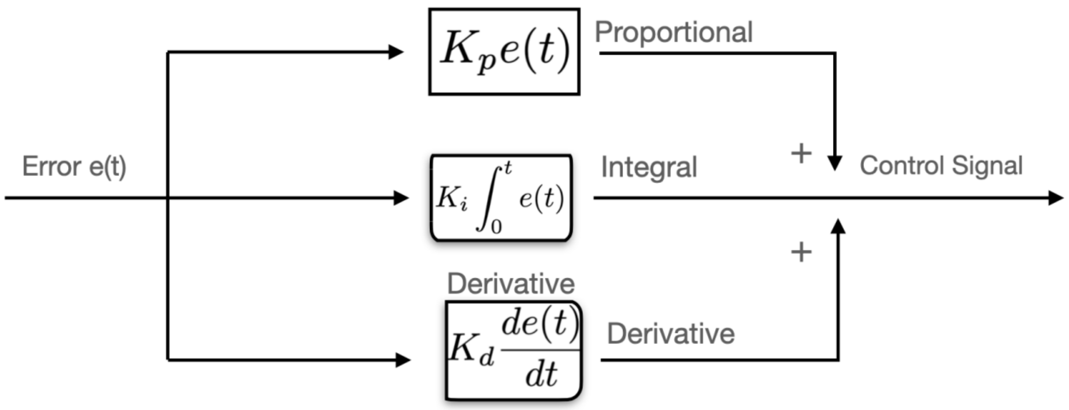

Figure 14: PID Controller Block Diagram

5. Adaptive Robust Learning Control Algorithm

To address the performance limitations of traditional PID control, I plan to develop a learning-based control approach that leverages past motion data and information to achieve superior performance. Simultaneously, the robotic system must be endowed with adaptiveness and robustness to effectively manage the uncertainties and disturbances commonly encountered in robotic systems.

5.1. Iterative Learning Control (ILC)

Iterative Learning Control (ILC) is a learning control strategy designed to enhance the performance of systems that execute repetitive tasks. By continually refining the control input, ILC can achieve near-perfect tracking for systems that repeatedly perform the same task.

$$e_k(t) = r(t) - y_k(t) \tag{5}$$

$$u_{k+1}(t) = u_k(t) + Le_k(t) \tag{6}$$

For convergence, we require that the spectral radius of $(I - CBL)$ is less than 1.

5.2. Adaptive Robust Learning (ARL) Control

In real-world robotics applications, constantly changing conditions are a common challenge. The Disturbance Observer (DOB) control method is used to enhance the performance of control systems by actively estimating and compensating for external disturbances.

5.2.1. Principle of the DOB Method.

$$y(s) = G(s)u(s) + d(s) \tag{7}$$

$$\hat{d}(s) = Q(s)\left[G^{-1}(s)y(s) - u(s)\right] \tag{8}$$

$$u_{DOB}(s) = u(s) - \hat{d}(s) \tag{9}$$

5.2.2. ARL Control Signal Analysis.

$$y(s) = G(s)u_k(s) + d(s) \tag{10}$$

$$u_k(s) = u_{k-1}(s) + Le_k(t) \tag{11}$$

$$\hat{d}(s) = Q(s)\left[G^{-1}(s)G(s)\left[u_{k-1}(s) + Le_k(t)\right] + d(s) - u_k(s)\right] \tag{12}$$

$$u(s) = \left[u_{k-1}(s) + Le_k(t)(s)\right] - \hat{d}(s) \tag{13}$$

6. ARL Results and Analysis

6.1. Control Algorithm Implementation

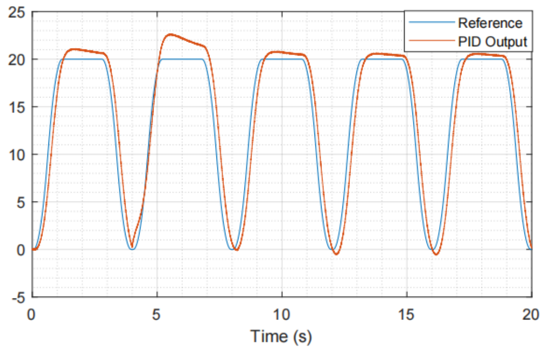

Figure 15: Tracking Performance with Tradition PID Feedback Control

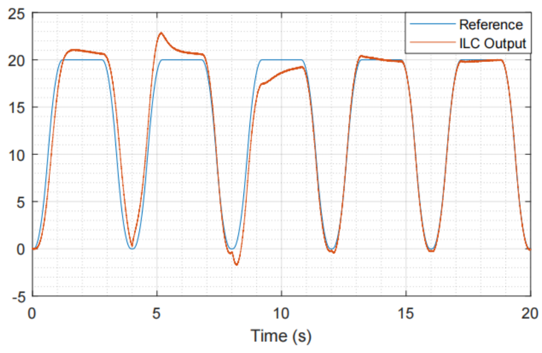

Figure 16: Tracking Performance with Standard Iterative Learning Control

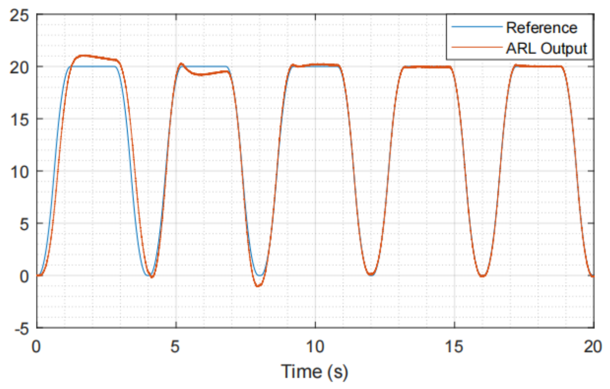

Figure 17: Tracking Performance with Adaptive Robust Learning(ARL) Control

6.2. Results

- **Method 1: Classical PID Control**
- **Method 2: Iterative Learning Control**
- **Method 3: Adaptive Robust Learning(ARL) Control**

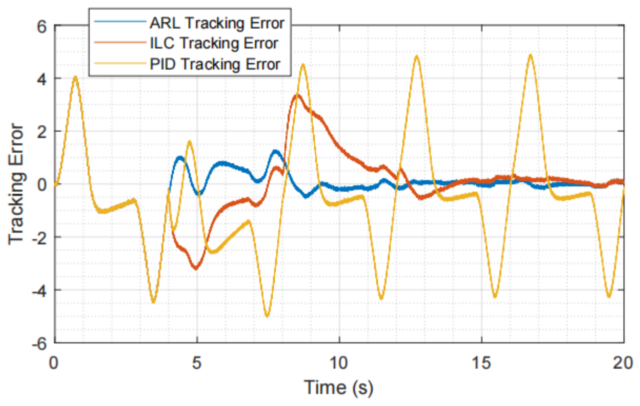

Figure 18: Tracking Error Comparison of Different Algorithms

6.2.1. RMSE Comparison.

$$\text{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2} \tag{14}$$

Table 1: RMSE value and percentage improvement of different control algorithms

No.	Algorithm	RMSE Value	Percentage Improvement
1	PID	2.2502	
2	ILC	1.4697	34.68%
3	ARL	1.004	55.39%

7. Conclusion

This paper presents the development of an Adaptive Robust Learning (ARL) Control algorithm and its potential application in robotics. The ARL algorithm combines a disturbance observer (DOB) with the Iterative Learning Control (ILC) concept to enhance the adaptability and robustness of the process. Results show that the proposed ARL Control significantly outperforms PID and standard ILC methods in both tracking accuracy and disturbance rejection. With ARL control the less rigid 3D printed materials can also attain a high precision and accuracy compared to traditional methods when conducting repeating tasks, which can make this project more affordable and energy efficient. Additionally, a 6-DOF robotic manipulator equipped with a gripper and a depth camera running the YOLO v10 object detection model was developed. However, the integration of robotic arm, YOLO model, and ARL control is still ongoing due to abruptive hardware problem. Future work will focus on addressing integration challenges to achieve a fully operational system capable of precise object manipulation in dynamic environments.

Acknowledgments

My tutor, Wang RuJie, a computer science teacher from my high school. He teaches me contexts of robotics, machine learning, and control theory. He guided me the concepts of Closed Loop Control, PID control and etc. I also attend his Artificial Intelligence class in our school for two years, where he teaches us basic statistics, linear algebra, and machine learning in python.

References

[1] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," 2024. [Online]. Available: https://arxiv.org/abs/2405.14458

[2] M. A. Johnson and M. H. Moradi, PID control. Springer, 2005.

[3] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," IEEE control systems magazine, vol. 26, no. 3, pp. 96–114, 2006.

[4] M. Hofer, L. Spannagl, and R. D'Andrea, "Iterative learning control for fast and accurate position tracking with an articulated soft robotic arm," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) . IEEE, 2019, pp. 6602–6607.

[5] M. Pierallini, F. Angelini, R. Mengacci, A. Palleschi, A. Bicchi, and M. Garabini, "Iterative learning control for compliant underactuated arms," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 6, pp. 3810–3822, 2023.

[6] C. E. Boudjedir, M. Bouri, and D. Boukhetala, "Model-free iterative learning control with non-repetitive trajectories for second-order mimo nonlinear systems—application to a delta robot," IEEE Transactions on Industrial Electronics, vol. 68, no. 8, pp. 7433–7443, 2020.

[7] J. Meng, S. Wang, G. Li, L. Jiang, X. Zhang, C. Liu, and Y. Xie, "Iterative-learning error compensation for autonomous parking of mobile manipulator in harsh industrial environment," Robotics and Computer-Integrated Manufacturing, vol. 68, p. 102077, 2021.

[8] T. Hayashi, H. Fujimoto, Y. Isaoka, and Y. Terada, "Projection-based iterative learning control for ball-screw-driven stage with consideration of rolling friction compensation," IEEJ Journal of Industry Applications, vol. 9, no. 2, pp. 132–139, 2020.

[9] W. He, T. Meng, X. He, and C. Sun, "Iterative learning control for a flapping wing micro aerial vehicle under distributed disturbances," IEEE transactions on cybernetics, vol. 49, no. 4, pp. 1524–1535, 2018.

[10] H. Shi, J. Zeng, and J. Guo, "Disturbance observer-based sliding mode control of active vertical suspension for high-speed rail vehicles," Vehicle System Dynamics, pp. 1–24, 2024.

[11] W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related meth- ods—an overview," IEEE Transactions on industrial electronics, vol. 63, no. 2, pp. 1083–1095, 2015.

[12] Y.-C. Liu, S. Laghrouche, D. Depernet, A. N'Diaye, A. Djerdir, and M. Cirrincione, "Disturbance-observer-based speed control for spmsm drives using modified super-twisting algorithm and ex- tended state observer," Asian Journal of Control, vol. 26, no. 3, pp. 1089–1102, 2024.

[13] peng zhihui. (2024) Github peng-zhihui. Accessed: 2024-09-03. [Online]. Available: https: //github.com/peng-zhihui/Dummy-Robot

[14] l. xin.li. (2024) Gitee switchpi-dummy. Accessed: 2024-09-03. [Online]. Available: https: //gitee.com/switchpi/dummy

[15] Tzutalin, "Labelimg," https://github.com/tzutalin/labelImg, 2023.

[16] W. Wang, "Advanced auto labeling solution with added features," https://github.com/ CVHub520/X-AnyLabeling, CVHub, 2023.

[17] W. Ao. (2024) Gitee switchpi-dummy. Accessed: 2024-09-03. [Online]. Available: https: //github.com/THU-MIG/yolov10.git

[18] Benben, "Screw and nuts detection dataset," https://aistudio.baidu.com/aistudio/datasetdetail/ 6045, 2019.

[19] O. Technology, "Orbbecsdk ros1: Ros1 sdk for orbbec devices," https://github.com/orbbec/ OrbbecSDK ROS1, 2024, accessed: 2024-09-14.

[20] "pyorbbecsdk: Python sdk for orbbec devices," https://github.com/orbbec/pyorbbecsdk, 2024, accessed: 2024-09-14.