Preliminary Study on the Development of a Closed-Loop Control System Using Optical Feedback

machines

MDPI

Article

Preliminary Study on the Development of a Closed-Loop Control System Using Optical Feedback

Vincenzo Marra *, Andrea De Martin, Antonio Carlo Bertolino and Massimo Sorli

Department of Mechanical and Aerospace Engineering, Politecnico di Torino, 10129 Torino, Italy; andrea.demartin@polito.it (A.D.M.); antonio.bertolino@polito.it (A.C.B.); massimo.sorli@polito.it (M.S.)
* Correspondence: vincenzo.marra@studenti.polito.it

Abstract

In recent decades, there has been a rapid development of computer vision techniques that have found applications in various engineering fields such as automation and robotics. This topic can also be developed and further explored in servosystems, a field dominated by physical sensors and transducers. This study proposes the development of a closed-loop control architecture with optical feedback starting from an already existing test bench. It suggests replacing the potentiometric transducer used for position feedback with a camera-based system. A primary contribution of this work lies in the selection of components for the development of the system and the creation of a customized dataset for training the object detection algorithm using the YOLOv8 neural network.

Keywords: yolo-based object detection; servosystems; machine vision; embedded systems

1. Introduction

Emerging in the early 1960s, machine vision has undergone rapid development in recent decades, driven by the advent of increasingly advanced sensors and high-performance devices. With the integration of deep learning and artificial intelligence algorithms, it has been widely applied in fields such as robotics, automation, autonomous driving, video surveillance, inspection, and quality control. In robotics, various computer vision techniques are employed for robot guidance in industrial environments, such as stereo vision, photogrammetry, light coding, structured light, and laser triangulation [1]. Technological advancements have led to the development of cobots capable of interacting with humans in work environments. Some cobots have simply integrated vision systems based on cameras into their end effectors, in a configuration known as eye-in-hand, for tasks such as pick and place and inspection [2], while others focus on quality control using multiple cameras and a more complex architecture [3,4]. In Human–Robot Collaboration (HRC), the most widespread sensing modality, capable of transmitting the most information, is vision [5]. Thanks to its precision, machine vision is also employed in defect detection [6] and remanufacturing [7]. An interesting review discussing the advancements and main techniques used in industrial inspection is available at [8].

Most of these applications rely on machine learning algorithms, which have gained significant attention in recent years due to their ability to ensure real-time performance. The use of neural networks for tasks such as object detection, object segmentation, and image processing is continuously evolving. A key advancement that has enabled the integration of these networks into contexts requiring short response times has been the transition from two-stage to one-stage detection algorithms. While two-stage object detectors generally

check for updates

Academic Editor: Dan Zhang

Received: 20 October 2025
Revised: 18 November 2025
Accepted: 22 November 2025
Published: 25 November 2025

Citation: Marra, V.; De Martin, A.; Bertolino, A.C.; Sorli, M. Preliminary Study on the Development of a Closed-Loop Control System Using Optical Feedback. *Machines* **2025**, *13*, 1085. https://doi.org/10.3390/machines13121085

preserve higher accuracy, their one-stage counterparts provide shorter inference times. A comprehensive review of the evolution of object detection algorithms can be found in [9]. YOLO (You Only Look Once), first introduced in [10], is of paramount importance in this context. Unlike other algorithms, YOLO reformulates the object detection task as a single regression problem that, starting from the image pixels, directly predicts the bounding box coordinates and the corresponding class probabilities. From an architectural perspective, YOLO employs a single convolutional network capable of simultaneously predicting multiple bounding boxes and their associated classes. The model is trained on full images and optimized directly for detection performance, without the need for complex processing pipelines. A detailed description of the evolution of the algorithm over the past years, highlighting how YOLO has become the standard for real-time object detection, can be found in [11,12]. In particular, the analysis also addresses its industrial applications and its current role in defect detection, automotive systems, and safety. YOLO has achieved accuracy and response times adequate for use in a wide range of contexts. In robotics, it is employed for recognition tasks to accelerate and automate processes [13] or to enhance the hand-eye coordination of robotic manipulators [14]. Furthermore, it is utilized in scenarios that involve the identification and classification of multiple objects simultaneously, under various environmental conditions, and over extended ranges from the targets [15,16].

The introduction of such solutions into the domain of servosystems is particularly interesting for several technical and practical reasons. The aim of this work is not to compare transducers and camera modules, but to discuss their substitution within a servo system, as it offers several significant advantages.

First, using a camera-based system as position feedback represents a non-intrusive and contactless alternative to traditional transducers employed in servosystems. This is particularly advantageous, as it removes the need for direct contact between the sensor and the actuator, thereby reducing the systematic errors to which conventional measuring elements are typically prone. In control and regulation tasks, the use of cameras enables the development of advanced feedback strategies based on image processing, allowing real-time adaptation to changing environmental conditions and external disturbances. Moreover, cameras enable position measurements along multiple spatial directions, unlike conventional linear transducers. Several works have integrated vision systems into control applications. In [17], a position control system for an actuator was developed using feedback provided by a Raspberry Pi Camera Module. Image processing for position identification and calibration was performed on a Jetson Nano, while the control signals were transmitted to the motor driver through an Arduino 2. In [18], a machine learning model was proposed to estimate the position of a mobile robot through a vision system employing Vicon cameras. Alternative perspectives have been provided by [19], who proposed a multi-sensor fusion approach for autonomous robot docking, in which position detection is performed using an RGB sensor based on a YOLO network. A Lidar group-symmetry mechanism is combined with visual feedback in a hybrid PID controller to correct angular errors. There are applications in which high accuracy is achieved, and hybrid control proves robust even under external disturbances, as in [20], where motion control of gallium-based liquid metal droplets is implemented through an industrial camera and an AC/DC power supply. There is also an example that does not involve motion control, but rather force control, with feedback always based on machine vision [21].

Although various approaches have been proposed in the literature, only a few have addressed the overall configuration and performance of the system. Moreover, all these developed systems are based on complex architectures involving multiple sensors and different hardware platforms. The novelty of this research lies in developing a system that

enables position feedback through a simplified architecture, using only a single camera and machine learning tools for target recognition.

As shown, machine learning is continuously expanding across different domains, and most related solutions are often implemented using embedded systems. The selection criteria for the hardware system and the camera module depend on the application type and can vary considerably according to the system's specific requirements. Ref. [22] presents a comprehensive review of the embedded systems and sensors employed in a wide range of machine vision applications, highlighting their performance characteristics and the key specifications to be evaluated when selecting a specific solution. Currently, the most widespread hardware systems are NVIDIA Jetson and Raspberry Pi. Several examples of the use of these platforms for object detection can be found in the literature [23–25]. One study proposes a framework for developing an object detection algorithm on NVIDIA Jetson, suggesting appropriate trade-offs between accuracy and latency [26]. In general, however, it is uncommon to find studies in which the system development itself constitutes an integral part of the research.

To address this limitation, the present work proposes the development of a closed-loop control architecture with optical feedback starting from an already existing test bench. It suggests replacing the potentiometric transducer used for position feedback with a camera-based system, aiming to explore the potential of modern machine vision techniques as an alternative to traditional feedback. The first contribution of this work concerns the selection of the components required to develop the system, including the choice of sensors and hardware platform, justified according to the intended application. This study also involves the construction of a custom dataset for training a YOLO neural network aimed at object recognition. A rigorous methodology for dataset development is presented, along with the results obtained from training in terms of confidence, precision, and algorithm accuracy. An additional important aspect of this research is inference time, which can represent a true bottleneck for this type of application. The evaluation of the YOLO algorithm on single-board computers with limited computational power is a crucial topic to explore, as it requires finding a trade-off among CPU load, inference speed, memory usage, and other factors. This topic is of particular interest in the literature, for instance, a benchmarking analysis comparing different embedded systems—such as Raspberry Pi and NVIDIA Jetson—and assessing YOLO performance in terms of frame rate, resolution, and other characteristics is provided in [27]. An additional contribution of this work involves the evaluation of inference time across different YOLO model export formats, aiming to provide insight into the delay introduced by the inference process in this context.

This paper is organized as follows. Section 1 provides an introduction, outlining the context, objectives, and overall structure of the work. Section 2 describes the methodology adopted for the study, including the system design, component selection, test bench layout, and YOLO network training process. Section 3 presents the results, focusing on detection evaluation and inference time analysis for different YOLO model export formats. Moreover, a simulation study has been conducted to illustrate how the presence of the camera and the inference process influence the system's stability. Finally, Section 4 summarizes the conclusions and outlines future developments of the work.

2. Methodology

This section describes the architecture of the position control system and presents the components selected for the implementation of optical feedback, together with the reasoning behind their selection. In addition, it details the procedure adopted to create the custom dataset used to train the neural network based on the YOLOv8n object detection algorithm. By relying on video recordings of the moving carriage, the images

required for the custom dataset were extracted and sampled. At this early stage, the algorithm is expected to reliably recognize the target, while subsequent work should establish a correlation between the detection results and the actual position of the target. Before the development of the detection algorithm, preliminary activities were undertaken to configure the hardware and software environment required for camera operation and control.

2.1. System Description

The controlled variable is the position of the carriage of a standard computer printer. Motion is provided by a DC motor that drives a toothed pulley coupled to a belt, with one branch rigidly attached to the carriage. The servo actuator incorporates a position transducer to continuously monitor the carriage displacement. The analog position control is housed in a rack that also integrates the function generator, the transducer conditioning stage, and the armature voltage regulator of the DC motor.

The experimental test bench on which this study is based consists of the following components:

- direct current (DC) motor;
- carriage driven by a toothed belt transmission;
- potentiometric position transducer;
- data acquisition and control system.

The motor used in the test bench is a brushed direct current (DC) motor. As shown in Figure 1, the motor is powered through two supply wires connected to its terminals. The motor rotor is rigidly coupled to a toothed pulley with a diameter of 14.5 mm, which drives the transmission belt. The belt, engaged at the opposite end with an idler pulley, is rigidly connected to the carriage along one of its branches. Consequently, the rotation of the motor rotor is converted by the belt into a translational motion of the carriage. The translating slide has a stroke of approximately 200 mm. The position transducer is a potentiometric type, model HLP 190/200/8k2 by Penny and Giles, featuring a hybrid track. It consists of a cylindrical body with one end fixed and the other attached to the carriage. The control unit is responsible for driving the electric motor and for closed-loop position feedback control. The command signal can be generated internally through a function generator module, or it can be supplied by an external generator. Closed-loop position control is implemented using an analog board with a PID controller, where the gain of each individual component can be adjusted. The unit also includes the motor driver and the conditioning board for the transducer signal. The control unit is housed in a standard 19″ rack, three units in height. The modules are designed as 19″ Eurocard plug-in boards with 32 + 32 + 32 connectors, each performing the following functions:

- electrical power supply module;
- function generator module;
- PID controller module;
- conditioning module for the transducer;
- motor driver module.

A National Instruments MyRio-1900 device (National Instruments (NI), Austin, TX, USA) and a PC equipped with LabVIEW (LabVIEW 2024 Q3) are employed to evaluate the performance and control the test bench.

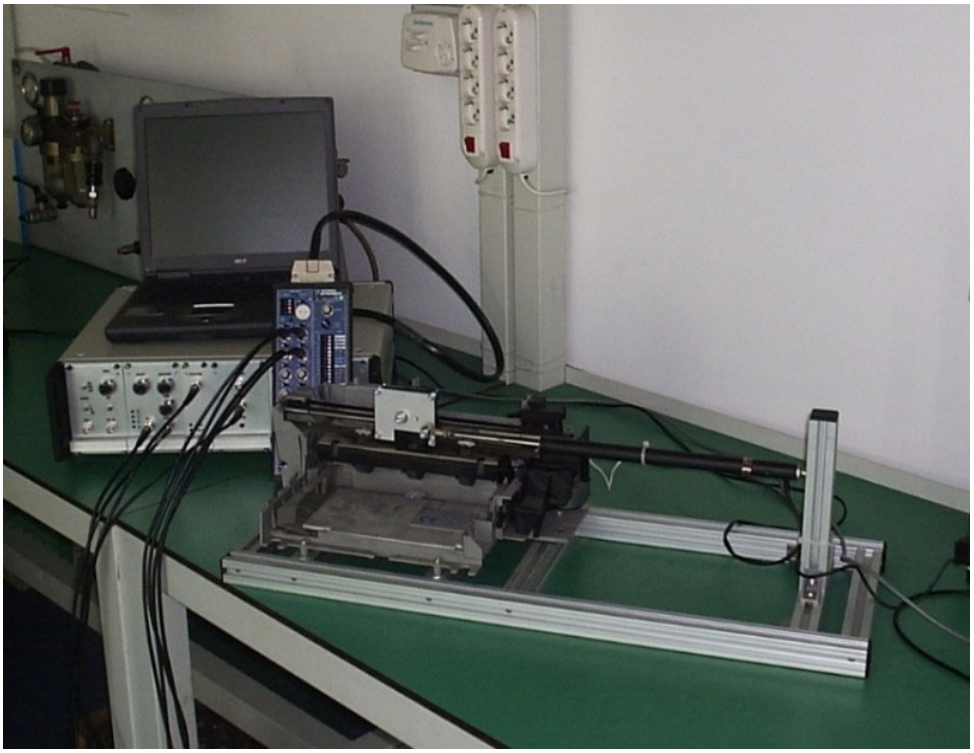

Figure 1. Test bench.

2.2. Component Selection and Test Bench Layout

The architecture conceived for the development of a closed-loop control system using optical feedback consists of cameras capable of performing spatial triangulation of the target, together with a hardware platform dedicated to image processing, object detection, and the correlation between the images acquired by the camera modules and the position. This device must also be able to transmit the appropriate signals to the driver of the electric motor for controlling the carriage.

The most relevant factors in selecting the hardware environment are the CPU clock speed, the number of GPIOs, and the number of cameras that can be connected. Several devices were considered, including Arduino R1 GIGA, Raspberry Pi 5, NVIDIA Jetson AGX Orin, and Renesas EK-RA6M4. After a detailed evaluation of performance and cost, Raspberry Pi 5 (Raspberry Pi Ltd. (commercial arm: Raspberry Pi Holdings), Cambridge, UK—specifically the Compute Module version (CM5)—was chosen as the hardware platform for system development. This board offers one of the highest CPU clock speeds among the solutions analyzed, 2.4 GHz. From the software side, it also represents one of the most advantageous options, as it can be programmed in Python—an accessible and effective programming language—and is fully supported by the OpenCV library, which provides extensive documentation for object tracking and pose estimation. Although the CM5 provides fewer GPIOs than other platforms, the available pins are suitable for the needs of this system. Cost considerations also play a crucial role in the choice: Jetson AGX Orin provides superior performance in terms of pins and camera connectivity, but it is considerably more advanced and much more expensive than Raspberry Pi 5. Compared with Renesas and Arduino boards, CM5 offers significantly higher performance and shorter response times, which makes it preferable over these alternatives. While Renesas and Arduino provide a high number of pins and support for multiple cameras, they are not specifically designed for image processing and machine vision applications. Table 1 reports all the main specifications of the different hardware platforms considered.

Table 1. Hardware platforms.

Device	Language	Number of Cameras	Clock Speed	GPIOs
Raspberry Pi 5 (Raspberry Pi Ltd., Cambridge, UK)	Python, C++, Java, Rust, Scratch	4	2.4 GHz	40
NVIDIA Jetson AGX Orin (NVIDIA Corp., Santa Clara, CA, USA)	Python, C, C++, Scratch, Rust	6	2.2 GHz	40
Renasas EK RA6M4 (Renesas Electronics Corp., Tokyo, Japan)	C++, Python, Rust	4	200 MHz	144
Arduino R1 Giga (Arduino AG, Turin, Italy)	C++	4	480 MHz	103

Apart from NVIDIA Jetson, all the evaluated solutions are reasonably affordable, and all are compatible with MATLAB and Simulink, which could be used to integrate additional models if needed. The final decision was strongly influenced by the need to develop a system with short response times and by the availability of a solution already widely adopted in object tracking applications, namely the Raspberry Pi platform.

The camera serves as a primary device for acquiring information about the target's position, with key parameters such as frame rate and sensor resolution directly influencing the accuracy of position estimation and the system's bandwidth. Other important factors include the sensor type (Global or Rolling Shutter), color mode, field of view (FOV), and the maximum number of cameras supported by a given hardware environment. Several camera models compatible with the evaluated hardware environments were considered. In line with the hardware platform evaluation, different solutions were analyzed with respect to their technical specifications, leading to the final selection. All the evaluated solutions are summarized in Table 2, where the main features relevant to the final selection are highlighted. The entry number of cameras refers to the maximum number of cameras that can be connected to a given platform, also considering the use of accessories such as multiplexers. Since the cameras support different video modes with variable resolutions and frame rates, the FPS column reports the maximum number of frames per second achievable in a specific mode. FOV is, in most cases, the horizontal field of view, except for multi-camera solutions, where the diagonal FOV is provided. For MEGA cameras, frame rate is not specified in the technical documentation.

The analysis shows that, among the cameras, the most capable option in terms of technical specifications is the one associated with the NVIDIA Jetson platform. MEGA cameras, on the other hand, offer limited performance, especially in terms of resolution and frame rate, since the FPS of the individual units cannot be determined. All the evaluated cameras rely on a Rolling Shutter sensor, with the only exception being OG02B10, which—despite its lower resolution—offers faster sensor dynamics. IMX477 sensor demonstrates excellent performance, with the only drawback of a relatively narrow field of view compared with the other options. Finally, the Quad Camera solution for Raspberry Pi 5 achieves the highest resolution, although it should be noted that the value reported in Table 2 refers to the combined output of the four sensors, each featuring 16 MP.

Table 2. Camera sensors.

Camera	Number of Cameras	FPS	FOV	Resolution
Quad camera Kit RP5	4	120	120°	64 MP
MEGA	4	/	90°	5 MP
Arducam IMX477	4	120	78°	12 MP
Arducam OG02B10	4	120	110°	2 MP
Pi camera module 3	4	120	66°	12 MP
Multi-camera Kit JN	6	60	120°	12 MP

The camera modules employed for the system development are: Arducam IMX477, Arducam OG02B10, and Pi camera module 3. The reasons behind this selection are manifold. One relevant factor is the use of a Global Shutter sensor: among the evaluated solutions, OG02B10 is the only camera module equipped with this feature, making it an attractive option as it ensures faster sensor dynamics and shorter image processing times. Another consideration concerns the differences in sensor characteristics. OG02B10 offers a wide field of view and faster image processing, whereas IMX477 provides higher resolution at the expense of a narrower FOV. Raspberry Pi Camera Module 3 represents a reasonable trade-off between these two extremes, although its wide-angle version introduces image distortion, which ultimately led to the adoption of the standard model. This approach highlights the rationale for selecting cameras with different specifications, so that each module provides distinct advantages in specific contexts: the IMX477, with its higher resolution, enhances accuracy, whereas OG02B10, with its Global Shutter and high frame rate, enables faster system dynamics despite its lower resolution. In this way, it becomes possible to evaluate different camera placements depending on the sensor characteristics, ultimately identifying the most effective configurations. Finally, some limitations emerged when considering multi-camera kits, such as those available for Raspberry Pi or Jetson Nano. While these solutions are advanced and capable of achieving excellent performance, they are less suitable for the present work. The main limitation is that the cameras within each kit are identical, leading to low flexibility in sensor selection; moreover, they considerably increase the overall system cost. Two camera modules were purchased for each selected type, in order to enable a broader analysis and to combine them by exploiting the strengths of each device. The development of the system also requires additional accessories, including a cooling system to handle the significant computational load sustained by the board, as well as a mouse, keyboard, and Raspberry Pi HDMI cable to allow programming through an external monitor. An SD card must be provided for Raspberry Pi Compute Module 5 (CM5), since, unlike Raspberry Pi 5, it does not include a built-in microSD slot or any default accessible storage system. In addition, an Analog-to-Digital Converter (ADC) module is required whenever analog signals need to be acquired using a digital system such as Raspberry Pi, which lacks native analog inputs.

2.3. Customize Dataset Construction for YOLOv8 Object Detection

The methodology adopted for custom dataset construction for the object detection algorithm is described in this section. To improve robustness, a custom dataset was built from video acquisitions of the carriage under its operating conditions. This dataset is intended for training the YOLO neural network to enable the recognition of moving objects. During the preliminary phase, the goal is to validate the proposed method; therefore, a single camera module was selected and installed facing the object so it can capture all its possible configurations. Among the available options, Pi Camera Module 3 was chosen. The carriage can operate under different reference signals, while the camera

module has several video modes. For this reason, video acquisitions were carried out under different commands and frame rate settings. For each command type, two recordings were performed: one at the minimum frame rate of 60 FPS and one at the maximum frame rate of 120 FPS. All videos were recorded in MP4 format with a duration of 25 s per acquisition. The reference signals consist of a sinusoidal waveform at 1 Hz and a step input. Figure 2 shows the setup used to perform the video recordings.

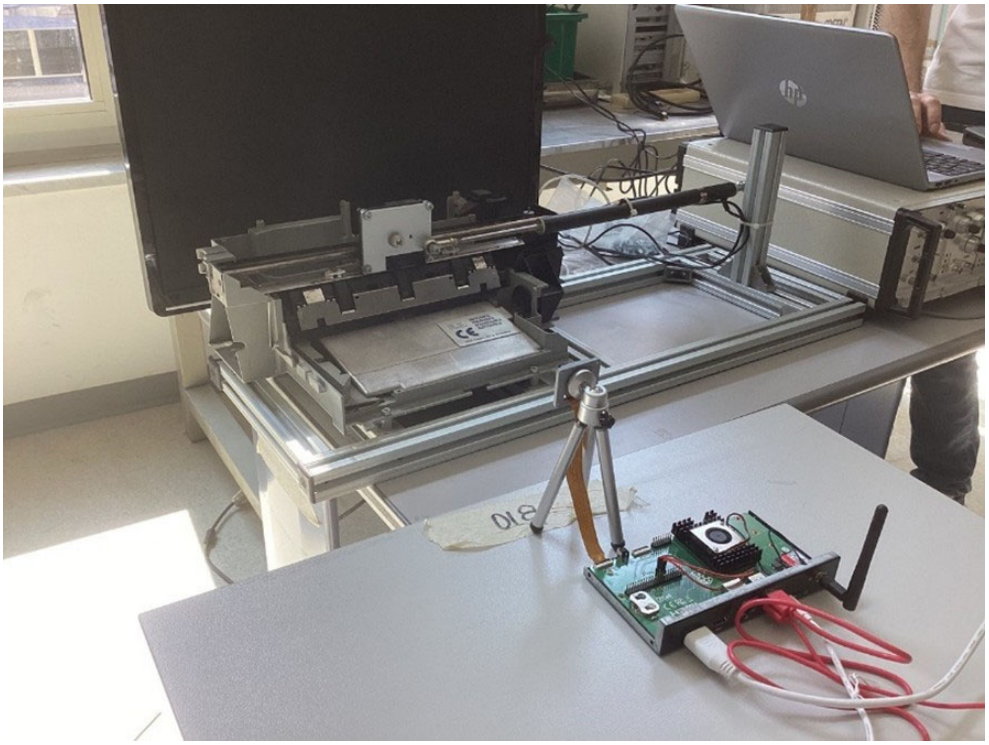

Figure 2. Video acquisition setup.

For the creation of the custom dataset used to train the network, the Roboflow platform was employed. Roboflow is an online tool widely adopted in computer vision projects, particularly for object detection and classification tasks. Its purpose is to streamline the entire workflow, from image collection to model training and deployment, thereby reducing both the time and the specialized expertise typically required for these activities. The platform provides several key functionalities that proved valuable in this work. It allows the upload and selection of images and videos in different formats to support the construction of custom datasets. Through an intuitive graphical interface, users can annotate objects of interest by creating bounding boxes of various shapes, including rectangular, polygonal, or user-defined. In addition, Roboflow offers preprocessing and augmentation, enabling adjustments to parameters such as brightness, exposure, and saturation within user-defined ranges. These transformations not only increase the size of the dataset but also enhance network robustness, allowing the model to recognize objects under a variety of conditions rather than being restricted to those present in the original images. Another important feature is dataset splitting, which enables the images to be separated into training, validation, and test subsets according to either predefined or user-specified percentages. This guarantees that the model is trained on one set of data while being evaluated on previously unseen examples. The platform also supports dataset export in standard formats, fully compatible with all YOLO versions up to the most recent, and provides the option of training and deploying the model directly in the cloud without the need for local execution through environments such as Python.

In this study, the dataset was constructed using frames extracted from video recordings. Once a video is uploaded, Roboflow allows the user to select the frame rate at which images are sampled. Since several video recordings were available for dataset construction and the free version of Roboflow restricts the dataset to 500 images, the frame rate was reduced to keep within this limit. The supported formats are MP4 and MOV. A baseline of 200 images was selected to construct the dataset, excluding preprocessing and augmentation steps, which eventually increased the dataset size to 500 images. Once the images were uploaded, labeling was carried out. Afterward, preprocessing and augmentation can be applied to the annotated images. Preprocessing ensures that the images are transformed into a coherent, standardized format compatible with the network. For example, YOLO algorithms operate with an input size of 640 $\times$ 640 pixels, which requires the selected images to be resized accordingly. Augmentation, on the other hand, introduces artificial variations into the dataset in order to simulate a broader range of real-world conditions than those originally captured. This step is essential to mitigate overfitting, a phenomenon in which the network memorizes the limited examples provided and fails to generalize effectively to unseen images. The next step is data splitting. Although there is no strict rule for selecting the division percentages, standard ratios are commonly adopted. A typical split, also suggested by Roboflow, is 70% for training, 20% for validation, and 10% for testing. When the dataset is relatively small, it is recommended to increase the training percentage, thereby increasing the amount of information available to the network during training. The dataset consists of 500 labeled images, split into 350 for training, 100 for validation, and 50 for testing. After dataset splitting, the images were exported for YOLO-based training. Although all versions up to YOLOv12 are supported, YOLOv8 was selected, given its optimization for platforms such as Raspberry Pi. The network was trained for 50 epochs with input image dimensions of 640 $\times$ 640, implemented through a Python script with Ultralytics.

After training, the PyTorch model files required for inference on the Raspberry Pi were converted into NCNN format. This format enables significantly shorter inference times compared with PyTorch, making it suitable for real-time applications such as the one addressed in this work.

3. Results

This section presents the evaluation of training results, describing the precision and accuracy of the object detection algorithm. Focus is also on inference time analysis based on several export formats and a closed-loop simulation, which highlights the influence of the vision system and how latency affects the loop's stability.

3.1. Detection Evaluation

To assess the robustness of the network after the training phase, several tools are provided, including charts, matrices, and collages, which offer key information for evaluating the quality of the process. YOLO assigns each class a specific ID number; since this work involves a single class to be detected, the ClassID number is 0. In the bounding box corresponding to the object, the class name is displayed along with the probability of the object belonging to that class. The carriage is successfully identified with high confidence scores. Figure 3 shows examples of detection, highlighting how effectively the network recognizes the target.

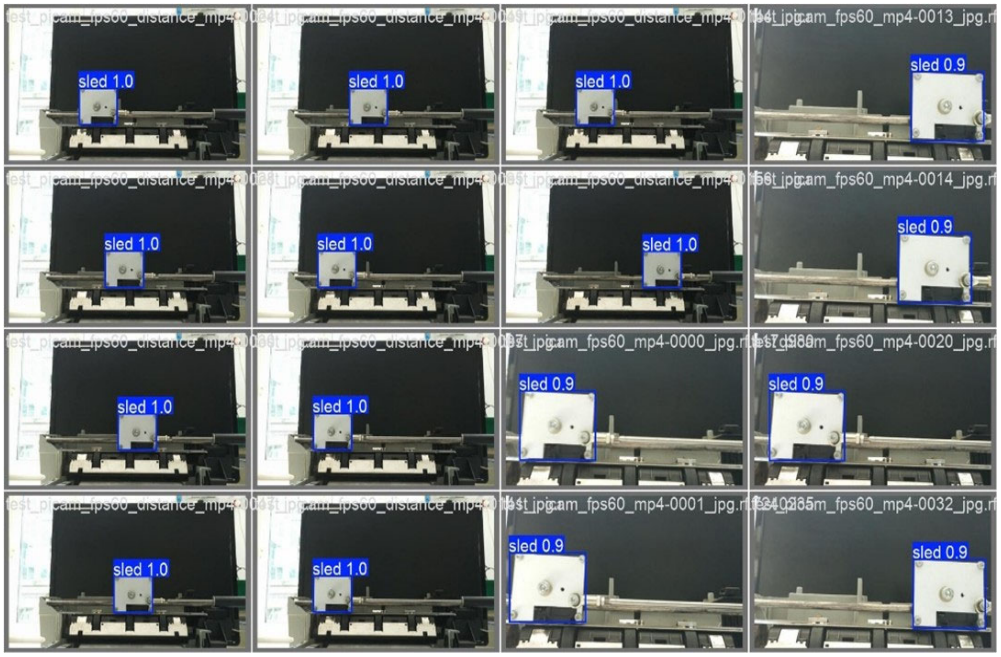

Figure 3. YOLOv8 detection results. (In blue there is the bounding box with his detection confidence score).

Figure 4 presents the Precision–Recall (PR) curve, which shows that the model achieved excellent performance in terms of precision and recall for the target class, with an average precision of 0.995. The results are consistent with training performed in a single context. The focus was on demonstrating the feasibility of detection rather than creating a robust dataset capable of generalizing recognition across different environments and under varying exposure and lighting conditions. The results show a high confidence score for detection. On the other hand, it should be noted that the system does not generalize well, and when inference is performed in different contexts, the detection performance drops significantly. Future work will address this limitation.

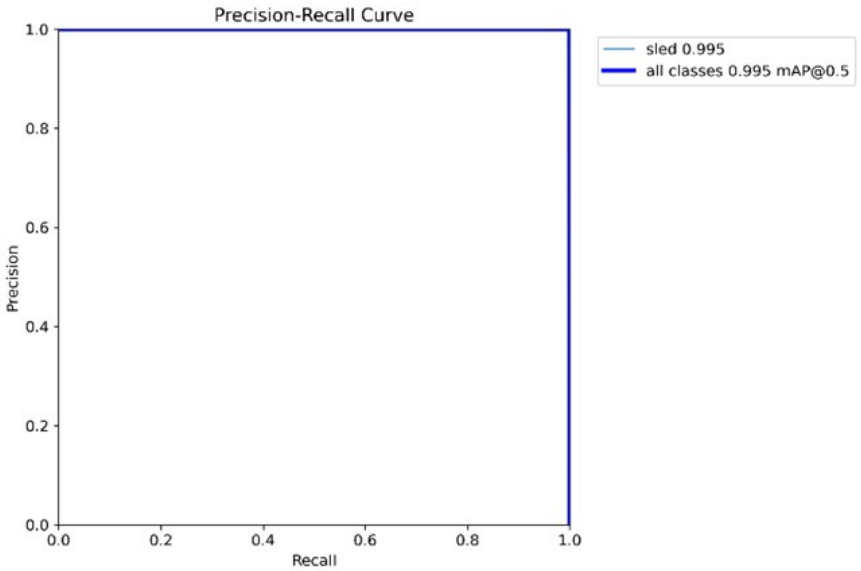

Figure 4. Precision–recall curve of the YOLOv8-based object detection model.

The Precision–Confidence curve is also reported in Figure 5, showing a confidence threshold value of 0.986. Confidence represents the probability that a given bounding box assigned by the network contains an object of the predicted class, while precision refers to

the percentage of correct predictions made by the model. Such a high threshold value arises because training was performed using video acquisitions always recorded in the same scenario; as a result, the model reliably recognizes the object in that environment. Since the network never encountered conditions leading to false positives, this curve captures a limited aspect of the network's performance and does not provide an evaluation of its capability to correctly discriminate object-free scenarios. Unlike other applications, in this case, the algorithm is designed to operate within that specific configuration. Precision, therefore, reflects the model's capability to minimize false positives, and it remains at a high level across the entire confidence range, confirming the robustness of the network in performing correct detections.

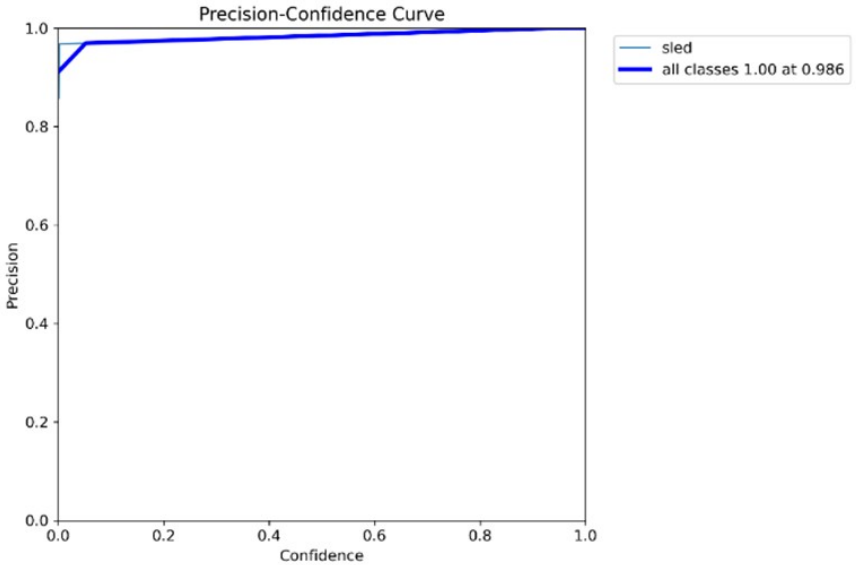

Figure 5. Precision-confidence curve of the YOLOv8-based object detection model.

The Recall–Confidence curve illustrates the network's ability to minimize false negatives. As shown in Figure 6, the curve remains close to 1.00 across a wide range of confidence values. This highlights the model's ability to detect the object even under varying conditions, while avoiding the generation of false negatives.

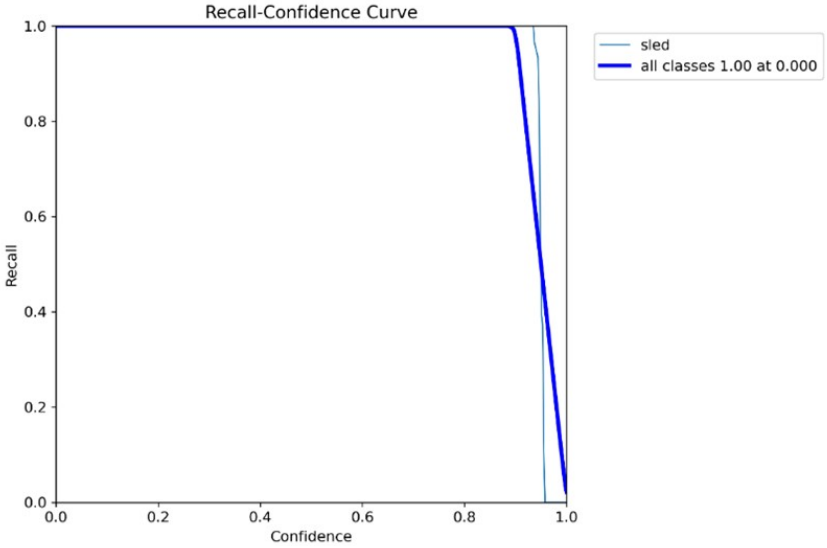

Figure 6. Recall-confidence curve of the YOLOv8-based object detection model.

The F1 curve illustrates the balance between precision and recall achieved by the trained network. As shown in Figure 7, the network provides high precision and reliability in identifying the target across different confidence thresholds.

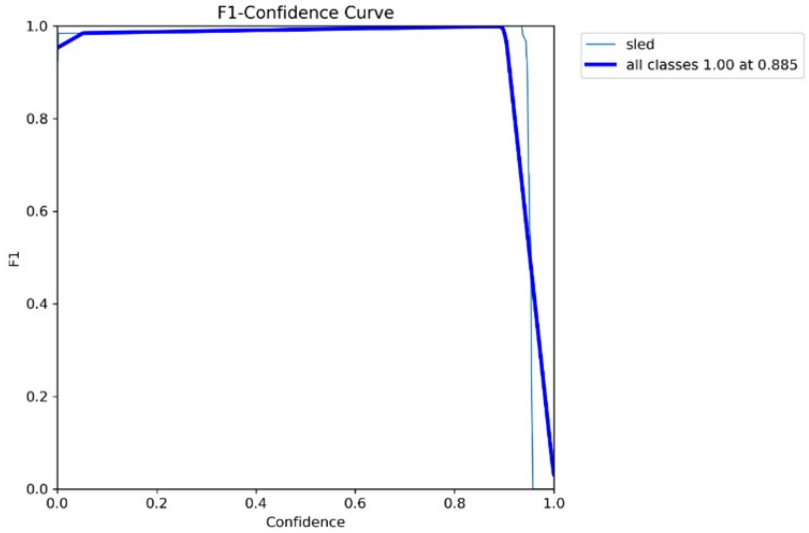

Figure 7. F1 curve of the YOLOv8-based object detection model.

Finally, additional metrics can be examined to assess the robustness of the network during training. Figure 8 shows that the mean average precision (mAP50–95) increases progressively throughout training before stabilizing at a high value. Since the dataset always depicts the same scenario, the mAP50 metric remains constant at a value close to 1, as the model quickly learns to recognize the objects in that specific context. In contrast, mAP50–95 serves as a tougher benchmark than mAP50, and its improvement during training is therefore more indicative of the network's actual progress. These results validate the reliability of the YOLOv8 model as an object detection algorithm for the system under study.

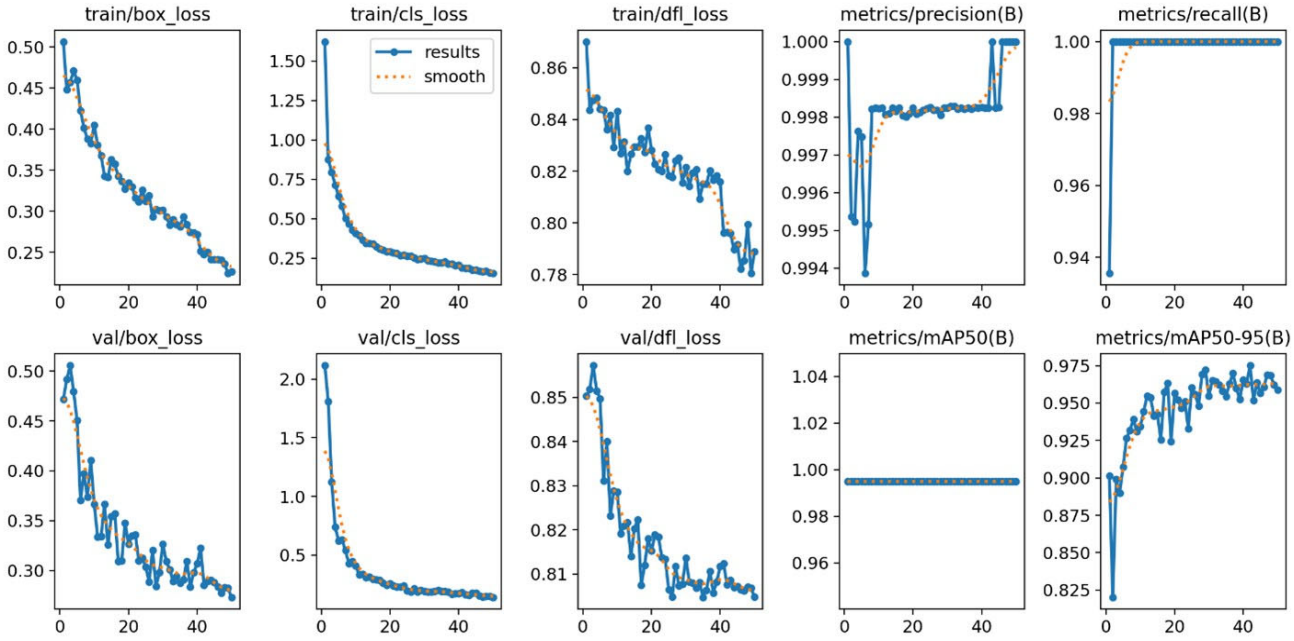

Figure 8. Training results.

3.2. Inference Time Evaluation

To evaluate the inference time of the network, data was collected directly from the platform and processed to estimate the delay introduced into the system. Ultralytics allows the YOLO model to be exported in several formats, including PyTorch, TorchScript, ONNX, OpenVINO, TF SavedModel, TF GraphDef, TF Lite, PaddlePaddle, MNN, and NCNN, each associated with different values of mean average precision (mAP) and inference time. Among these, NCNN proved to be the most suitable format for applications on embedded systems such as Raspberry Pi, as it provides a good trade-off between accuracy and short inference time. On the other hand, PyTorch is considerably slower and unsuitable for the development of this system. Figure 9 presents two probability density histograms of inference times for PyTorch and NCNN formats. As reported, both distributions are asymmetric, with markedly different inference time values. The PyTorch format introduces a delay of at least 400 ms, whereas NCNN achieves inference times in the range of 10–20 ms.

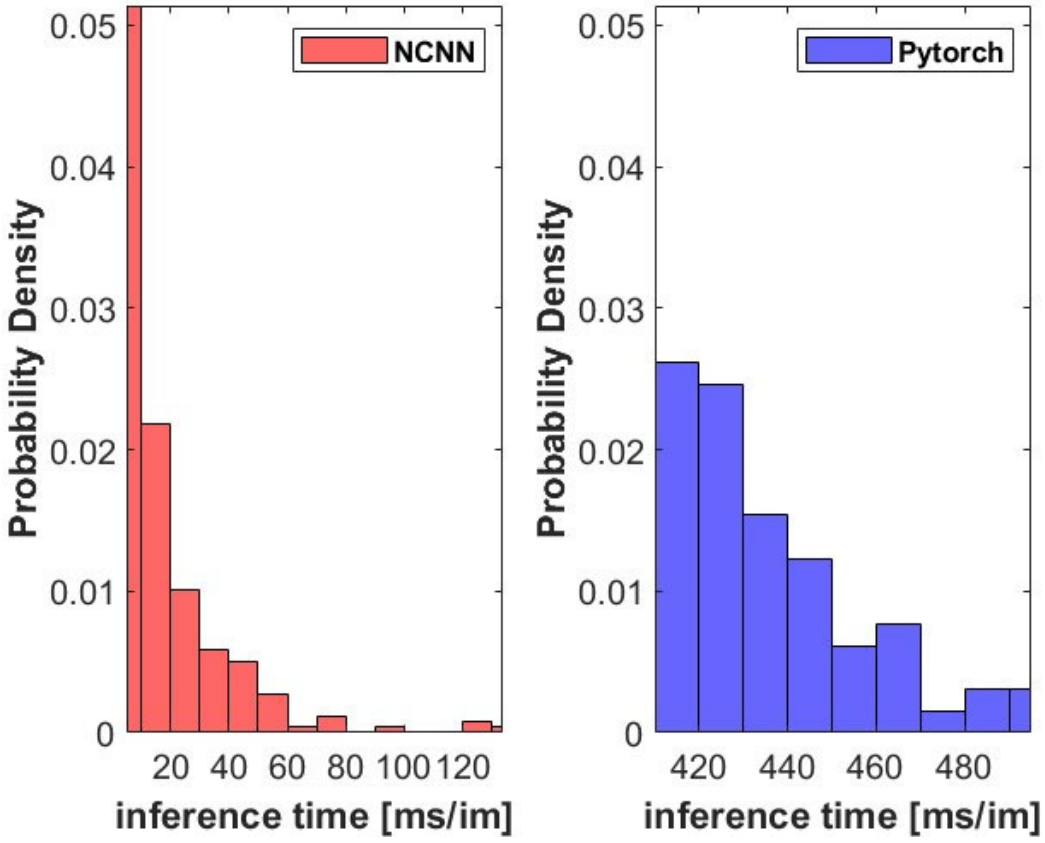

Figure 9. Probability density of inference time for the object-detection model deployed with NCNN and PyTorch.

Figure 10 shows the box plot of inference times obtained during the validation of the object detection algorithm for the two export formats. As shown, 50% of NCNN's inference times (between the first and third quartile) fall in the range of 6 to 20 ms, whereas with PyTorch, the times exceed 400 ms. In the latency analysis, only the inference time is considered, as it represents the most significant contribution. It is worth mentioning that the preprocessing and post-processing times range between 1 and 2 ms per inference. The presence of outliers in both formats cannot be attributed to changes in the scenario or to target motion, since the data was collected with the object stationary and correctly detected by the network. Instead, these anomalies are likely due to hardware-related issues.

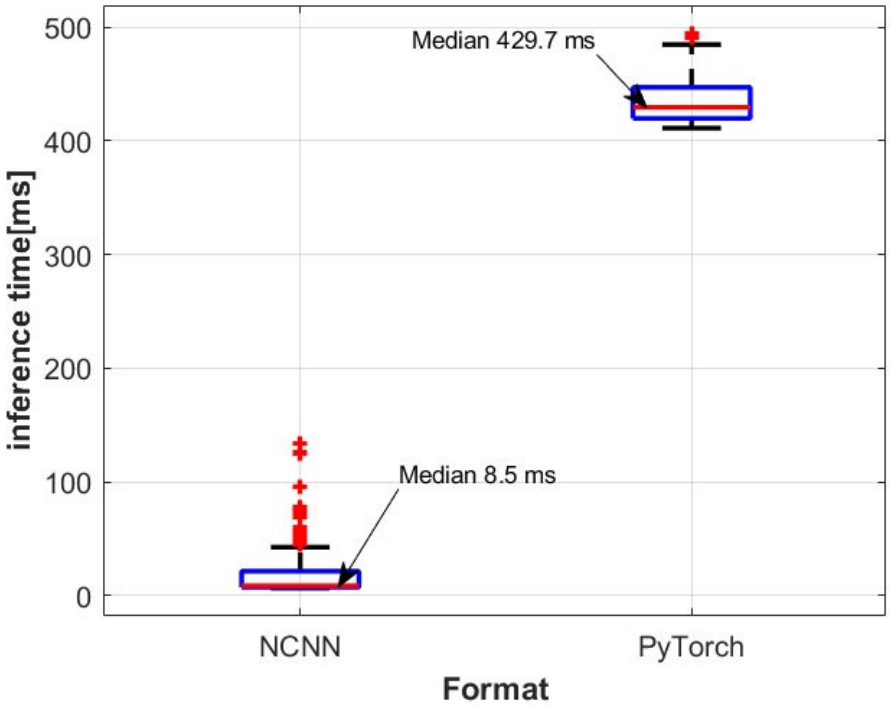

Figure 10. Box plot inference time for NCNN and Pytorch formats. (The red crosses represent the outliers of the distribution, while the black line indicates the upper whisker).

The obtained results highlight a clear difference between the two formats: the NCNN implementation achieves p95 and p99 latencies of 53 ms and 123 ms, respectively, whereas the PyTorch version reaches 481 ms and 494 ms. Overall, the results provide insight into the delays introduced by inference and indicate which export format is more suitable for developing this system using Raspberry Pi.

Although the results are promising, the inference times show significant variability, leading to an unstable frame rate that is clearly undesirable for this type of application. This aspect requires further investigation and optimization to improve the effectiveness of the solution. In every selected format, inference introduces a delay into the system that cannot be neglected and represents one of the most critical factors of this work. Nevertheless, the results confirm the feasibility of the proposed approach and enable additional steps aimed at developing the complete system by introducing the correlation with the position.

3.3. Closed-Loop Simulation Results

In this subsection, the results of a closed-loop simulation are presented to quantify how typical latencies affect the closed-loop frequency response of the system in terms of phase and bandwidth margins. This approach is useful to justify the feasibility of the proposed system.

The Simulink model includes the camera within the feedback loop, specifically the IMX477 camera module, which is modeled as a measurement device capable of sampling the signal according to its frame rate and quantizing it based on the pixel resolution provided by the sensor. The inference process is represented as a transport delay before closing the position loop. The inference time considered refers exclusively to the NCNN format, as simulations with the PyTorch format would not be meaningful due to the excessive latency that prevents the system from operating properly.

The simulations allow deriving the frequency response of the system by reporting both the magnitude (in dB) and phase (in degrees) as a function of a simulation parameter. In particular, the first simulation aims to assess the control feasibility. By considering a

constant inference time equal to the median value of the NCNN format (10 ms, including pre- and post-processing times), it is possible to obtain Bode diagrams of the closed-loop response for different values of the proportional gain K_p. Figure 11 shows the magnitude and phase of the closed-loop transfer function as K_p varies. Since this is a position control system, only a proportional gain is acceptable, while the other control gains are set to zero.

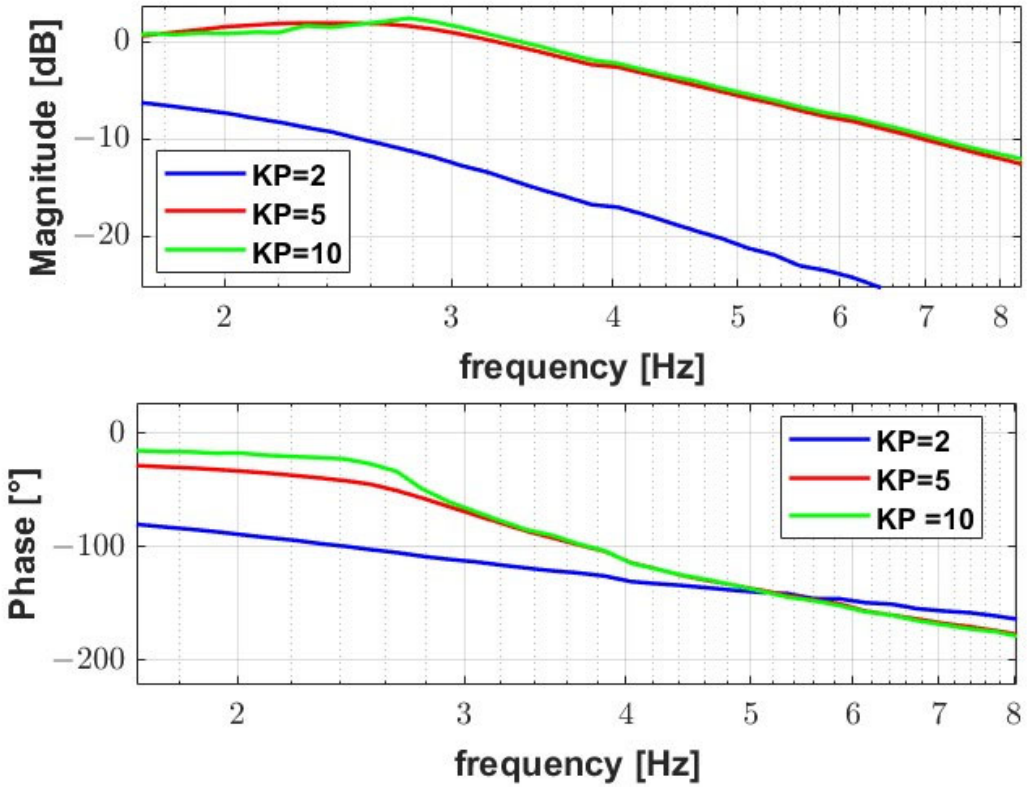

Figure 11. Effects of varying K_p on the closed-loop Bode diagrams.

As we can see, increasing K_p leads to a higher gain crossover frequency and a wider system bandwidth, but it also reduces stability by decreasing the corresponding margins. With $K_p = 2$, the response is slow and highly attenuated, whereas for $K_p = 5$, the system shows a prompt response at low frequencies. With a properly tuned proportional gain, the system bandwidth reaches approximately 3–4 Hz.

The second simulation highlights how the latency introduced by the inference process affects the system's stability and control performance. In the transport delay of the model, different latency values were considered, namely the NCNN median value (10 ms), the value corresponding to the 75th percentile (20 ms), and the upper limit of the distribution (40 ms). The proportional gain K_p is set to 5.

Figure 12 shows how the delay introduced by the inference process affects the closed-loop frequency response, particularly its magnitude and phase. Considering the median value of the NCNN format's distribution, the system remains stable with an acceptable phase margin. As the delay increases, the phase margin decreases, and a noticeable peak in the magnitude diagram appears near the gain crossover frequency. For a latency of 40 ms, the system operates close to instability. Peaks can be observed in the magnitude diagram as the latency increases, while the phase, as expected, drops much more rapidly.

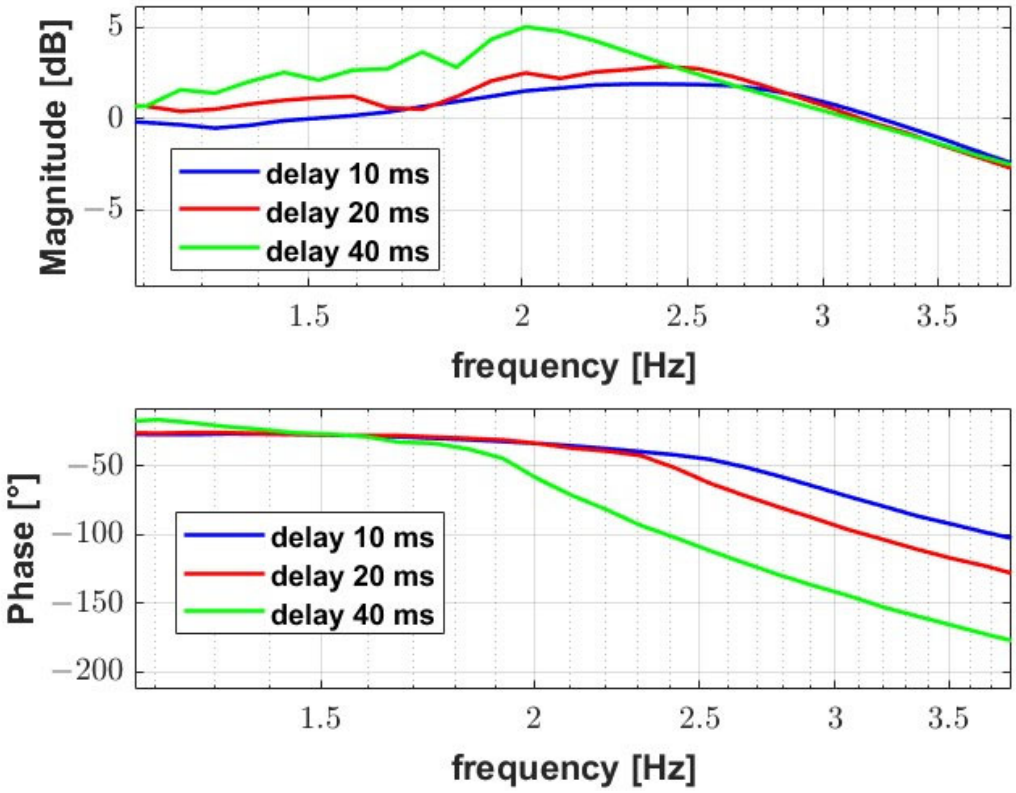

Figure 12. Effects of varying latency on the closed-loop Bode diagrams.

Simulation results indicate that the system responds effectively at frequencies matching those of the reference signals used during the video acquisitions (1 Hz). These results demonstrate the feasibility of the proposed system, highlighting the range of latency values that can be considered acceptable to ensure system stability.

4. Conclusions

This paper presents a preliminary study on the development of a closed-loop control system using optical feedback, introducing a new layout for an already existing test bench. The replacement of the potentiometric transducer with a vision-based system employing camera modules was investigated.

One of the main contributions of this work lies in the selection of hardware components for the development of the system, among several possible solutions, with the choice motivated by the performance provided by each option. In addition, a methodology was outlined for the development of the object detection algorithm based on a YOLO neural network. A custom dataset was constructed to enhance the robustness of the network used for object recognition. The detection results demonstrated a highly accurate network capable of recognizing the object with high confidence scores, as long as the camera module captures it in configurations included in the training set.

A key focus of this study was the inference time and the delay it introduces into the system. The YOLO model can be exported in several formats, each with different mAP values and response times. NCNN proved to be the most suitable format for the development of the system on Raspberry Pi. In this work, an indication of the delay introduced by inference has been provided. The simulations highlighted the range of latency values within which the system remains stable.

Although the outcomes are promising, the work presents some limitations. In terms of component selection, the proposed system represents a proof-of-concept prototype, while more advanced and efficient solutions may involve different costs and performance levels.

Thanks to the NCNN export, the inference time of the object detection algorithm is suitable for real-time performance; however, inference itself remains an aspect to be improved, as it produces a variable frame rate that requires stabilization.

Future work could focus on stabilizing the frame rate or improving dataset construction, enabling recognition under different camera configurations, and further enhancing robustness. A key issue that still needs to be addressed is the correlation with the position in order to finalize the development of the system.

Author Contributions: Conceptualization, V.M., A.D.M., and A.C.B.; methodology, V.M., A.D.M., and A.C.B.; software, V.M.; validation, V.M.; formal analysis, V.M.; investigation, V.M.; resources, V.M.; data curation, V.M.; writing—original draft preparation, V.M.; writing—review and editing, V.M., A.D.M., and A.C.B.; visualization, V.M.; supervision, V.M., A.D.M., A.C.B., and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Restrictions apply to the datasets presented in this article. The datasets are not readily available because the data are part of an ongoing study which limits public sharing. Requests to access the datasets should be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

YOLO	You Only Look Once
RP5	Raspberry Pi 5
JN	Jetson Nano
CM5	Computer Module 5
PR	Precision–Recall
FOV	Field Of View

References

1. Pérez, L.; Rodríguez, Í.; Rodríguez, N.; Usamentiaga, R.; García, D.F. Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review. *Sensors* **2016**, *16*, 335. [CrossRef] [PubMed]
2. Yang, X.; Zhou, Z.; Sørensen, J.H.; Christensen, C.B.; Ünalan, M.; Zhang, X. Automation of SME Production with a Cobot System Powered by Learning-Based Vision. *Robot. Comput. Integr. Manuf.* **2023**, *83*, 102564. [CrossRef]
3. Kohut, P.; Skop, K. Vision Systems for a UR5 Cobot on a Quality Control Robotic Station. *Appl. Sci.* **2024**, *14*, 9469. [CrossRef]
4. Prezas, L.; Michalos, G.; Arkouli, Z.; Katsikarelis, A.; Makris, S. AI-Enhanced Vision System for Dispensing Process Monitoring and Quality Control in Manufacturing of Large Parts. *Procedia CIRP* **2022**, *107*, 1275–1280. [CrossRef]
5. Semeraro, F.; Griffiths, A.; Cangelosi, A. Human–Robot Collaboration and Machine Learning: A Systematic Review of Recent Research. *Robot. Comput. Integr. Manuf.* **2022**, *79*, 102432. [CrossRef]
6. Maculotti, G.; Giorio, L.; Genta, G.; Galetto, M. Traceability and Uncertainty of Defects Automated Measurements by CNN-Powered Machine Vision Systems. *CIRP Ann.* **2025**, *74*, 661–665. [CrossRef]
7. Mangold, S.; Steiner, C.; Friedmann, M.; Fleischer, J. Vision-Based Screw Head Detection for Automated Disassembly for Remanufacturing. *Procedia CIRP* **2022**, *105*, 1–6. [CrossRef]
8. Ren, Z.; Fang, F.; Yan, N.; Wu, Y. State of the Art in Defect Detection Based on Machine Vision. *Int. J. Precis. Eng. Manuf.-Green Technol.* **2022**, *9*, 661–691. [CrossRef]
9. Diwan, T.; Anirudh, G.; Tembhurne, J.V. Object Detection Using YOLO: Challenges, Architectural Successors, Datasets and Applications. *Multimed. Tools Appl.* **2023**, *82*, 9243–9275. [CrossRef]
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Denver, CO, USA, 2–6 June 2026.
11. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo Algorithm Developments. *Procedia Comput. Sci.* **2021**, *199*, 1066–1073. [CrossRef]

12. Hussain, M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines* **2023**, *11*, 677. [CrossRef]
13. Claudia Carvalho, A.; Isabel Carvalho, A.; Correia, R. Robot with Vision Guidance System for Unscrewing Operation. In Proceedings of the 2024 International Conference on Decision Aid Sciences and Applications, DASA 2024, Manama, Bahrain, 11–12 December 2024; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2024.
14. Wei, Y.; Liao, C.; Zhang, L.; Zhang, Q.; Shen, Y.; Zang, Y.; Li, S.; Huang, H. Enhanced Hand–Eye Coordination Control for Six-Axis Robots Using YOLOv5 with Attention Module. *Actuators* **2024**, *13*, 374. [CrossRef]
15. Luo, W.; Zhang, G.; Shao, Q.; Zhao, Y.; Wang, D.; Zhang, X.; Liu, K.; Li, X.; Liu, J.; Wang, P.; et al. An Efficient Visual Servo Tracker for Herd Monitoring by UAV. *Sci. Rep.* **2024**, *14*, 10463. [CrossRef] [PubMed]
16. Zhang, G.; Wang, G.; Chen, J.; Jiang, W.; Hao, X.; Deng, T. An Automatic Control System Based on Machine Vision and Deep Learning for Car Windscreen Clean. *Sci. Rep.* **2025**, *15*, 4857. [CrossRef] [PubMed]
17. Rahsepas, K.; Sadighi, A. Use of Machine Vision for Feedback Computation in Motion Control Systems. 2024. Available online: https://www.researchgate.net/publication/387314825_Use_of_Machine_Vision_for_Feedback_Computation_in_Motion_Control_Systems (accessed on 12 December 2024).
18. Nowak, T.; Große-Kreul, A.; Boshoff, M.; Kuhlenkötter, B. Enhancing Mobile Robot Position Estimation with Machine Learning Methods Using Camera-Based Tracking. *Procedia CIRP* **2024**, *130*, 964–968. [CrossRef]
19. Dai, Y.; Lee, K. Multi-Sensor Fusion for Autonomous Mobile Robot Docking: Integrating LiDAR, YOLO-Based AprilTag Detection, and Depth-Aided Localization. *Electronics* **2025**, *14*, 2769. [CrossRef]
20. Ma, Y.; Feng, B.; Li, K.; Zhang, L. Motion Control of Gallium-Based Liquid Metal Droplets in Abrasive Suspensions Within a Flow Channel. *Actuators* **2025**, *14*, 456. [CrossRef]
21. Ren, Z.; Zhang, T. Automatic Control System of Manipulator Clamping Angle Based on Machine Vision. *J. Phys. Conf. Ser.* **2023**, *2479*, 012052. [CrossRef]
22. Biglari, A.; Tang, W. A Review of Embedded Machine Learning Based on Hardware, Application, and Sensing Scheme. *Sensors* **2023**, *23*, 2131. [CrossRef]
23. Dong, X.; Dukes, X.; Littleton, J.; Neville, T.; Rollerson, C.; Quinney, A. Object Detection on Raspberry Pi. In Proceedings of the 2022 ASEE Gulf Southwest Annual Conference, Prairie View, TX, USA, 16–18 March 2022.
24. Budiyanta, N.E.; Sereati, C.O.; Manalu, F.R.G. Processing Time Increasement of Non-Rice Object Detection Based on YOLOv3-Tiny Using Movidius NCS 2 on Raspberry Pi. *Bull. Electr. Eng. Inform.* **2022**, *11*, 1056–1061. [CrossRef]
25. Islam, R.B.; Akhter, S.; Iqbal, F.; Saif Ur Rahman, M.; Khan, R. Deep Learning Based Object Detection and Surrounding Environment Description for Visually Impaired People. *Heliyon* **2023**, *9*, e16924. [CrossRef] [PubMed]
26. Shin, D.J.; Kim, J.J. A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform. *Appl. Sci.* **2022**, *12*, 3734. [CrossRef]
27. Feng, H.; Mu, G.; Zhong, S.; Zhang, P.; Yuan, T. Benchmark Analysis of YOLO Performance on Edge Intelligence Devices. *Cryptography* **2022**, *6*, 16. [CrossRef]