Muhammad Zaeem Sarfraz

Robotic Attachment System for Hospital Bed and Trolley Transportation

A Hybrid Computer Vision Approach for Autonomous Hospital Logistics

In collaboration with Done Robotics Ab Oy

School of Technology and Innovations

Master of Science in Computing Sciences

Discipline: Sustainable and Autonomous Systems

Vaasa 2026

UNIVERSITY OF VAASA

School of Technology and Innovations

Author: Muhammad Zaeem Sarfraz

Thesis title: Robotic Attachment System for Hospital Bed and Trolley Transportation : A Hybrid Computer Vision Approach for Autonomous Hospital Logistics

Degree: Master of Science in Computing Sciences

Discipline: Sustainable and Autonomous Systems

Supervisors: Mahmoud Elsanhoury (principal supervisor), Thomas Höglund, Kannan Selvan

Year of graduation: 2026 **Number of pages:** 87

ABSTRACT:

Autonomous mobile robots (AMRs) in hospital logistics require precise visual alignment systems for trolley docking. This thesis presents a hybrid computer vision approach that combines deep learning with geometric edge-based processing to detect both large trolley handles and small mounting holes in real time. The central challenge is multi-scale detection: handles occupy approximately 35% of the frame, while holes occupy approximately 0.14%. A YOLOv8n model trained on 39 annotated images (32 training and 7 validation, without a separate held-out test set) achieved reliable handle localization, but practical hole detection from machine learning alone remained unstable. To address this limitation, YOLO-based handle detection was used to define a region of interest, and Canny edge detection with contour circularity filtering was used for hole localization. Subsystem-level evaluation showed 96% hole detection success in ROI-constrained tests and approximately 19–20 FPS processing on CPU hardware, with centimeter-level alignment estimation in controlled calibration cases. Due to the thesis time frame, full physical end-to-end docking integration (perception, motion execution, and mechanical engagement in one autonomous sequence) was not completed; perception and hardware subsystems were validated separately and full integrated docking trials are planned as future team work. Results indicate that combining context-aware machine learning for larger objects with geometry-driven methods for small features is more robust than pure machine-learning-only or pure-classical-only detection pipelines for this task.

Keywords: Computer vision, YOLOv8, robotic docking, hybrid vision, small object detection, hospital logistics, ROS, and visual servoing.

Contents

List of Figures

List of Tables

Abbreviations

AGV	Automated Guided Vehicle
AMR	Autonomous Mobile Robot
ArUco	Augmented Reality University of Cordoba fiducial marker family
CAD	Computer-Aided Design
CLAHE	Contrast Limited Adaptive Histogram Equalization
CPU	Central Processing Unit
CV	Computer Vision
IBVS	Image-Based Visual Servoing
IMR	Intelligent Mobile Robot
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mAP	mean Average Precision
PBVS	Position-Based Visual Servoing
RaaS	Robotics as a Service
RGB-D	Red Green Blue plus Depth
ROI	Region of Interest
ROS	Robot Operating System
ToF	Time of Flight
UV-C	Ultraviolet-C germicidal spectrum
YOLO	You Only Look Once

1 Introduction

1.1 Background and Motivation

The healthcare sector is undergoing a significant transformation driven by automation and robotics technology. Autonomous mobile robots (AMRs) are increasingly deployed in hospitals and healthcare facilities to address critical challenges including global labor shortages, the demands of aging populations, and the need for rigorous infection control in clinical environments (Holland et al., 2021; Kim, Kang, Lee, & Shim, 2024). These robots automate repetitive and hazardous tasks, allowing healthcare professionals to focus on patient care while improving operational efficiency and safety.

One of the key challenges in deploying AMRs for hospital logistics is the ability to precisely dock with trolleys, beds, and other equipment for autonomous transportation. Traditional manual docking processes are time-consuming and physically demanding for staff. Autonomous docking requires robust computer vision systems capable of detecting and localizing attachment points with millimeter-level precision under variable lighting conditions, viewing angles, and dynamic hospital environments.

Computer vision has emerged as a critical enabling technology for robotic manipulation and navigation tasks. Recent advances in deep learning, particularly object detection algorithms such as the YOLO (You Only Look Once) family, have demonstrated remarkable success in detecting objects in real-time (Redmon, Divvala, Girshick, & Farhadi, 2016; Ultralytics, 2023). However, these systems face significant challenges when dealing with small objects that occupy less than 1% of the image area, such as the mounting holes required for trolley docking mechanisms (Lin et al., 2014; Singh & Davis, 2018).

1.2 Case Study: Done Robotics and the Drivey Robot

This thesis work was conducted in collaboration with Done Robotics Ab Oy, a Finnish technology startup headquartered in Vaasa, Finland. Founded in 2020 by Thomas Höglund and Samuel Broman, Done Robotics operates as a subsidiary of Done Enterprises Ab

Oy and specializes in the design, development, and deployment of autonomous service robots tailored primarily for the healthcare and public sectors.

The company is driven by the core vision ”Robots for People,” aiming to create a global robotic ecosystem that enhances safety and quality of life by automating repetitive or hazardous tasks. Done Robotics' technological foundation is built on fifth-generation Intelligent Mobile Robots (IMRs), which utilize AI-driven context awareness, 360-degree 3D perception including LiDAR and stereo vision, and natural human-robot interaction to navigate complex, dynamic spaces alongside people.

The organization's current flagship solutions include Charlie, an autonomous UV-C disinfection robot; Drivey, a holonomic autonomous bed transporter and delivery robot; and Betty, a social assisting robot in development. This thesis focuses on the Drivey robot, which is designed to reduce the physical burden on hospital staff and improve internal logistics through precise autonomous navigation and docking capabilities.

Done Robotics operates under a Robotics as a Service (RaaS) business model, offering subscription-based deployment that allows healthcare providers to implement automation as a predictable operational expense. The company handles all training, maintenance, and software updates throughout the robot's lifecycle, ensuring reliable and adaptable systems that can scale across various industries beyond healthcare.

Organizationally, Done Robotics operates through cross-functional teams including software/ROS development, hardware and mechanical integration, and product operations. During this thesis project, my practical contribution was focused on ROS integration and hardware subsystem development to ensure the perception pipeline could be prepared for full docking-system integration.

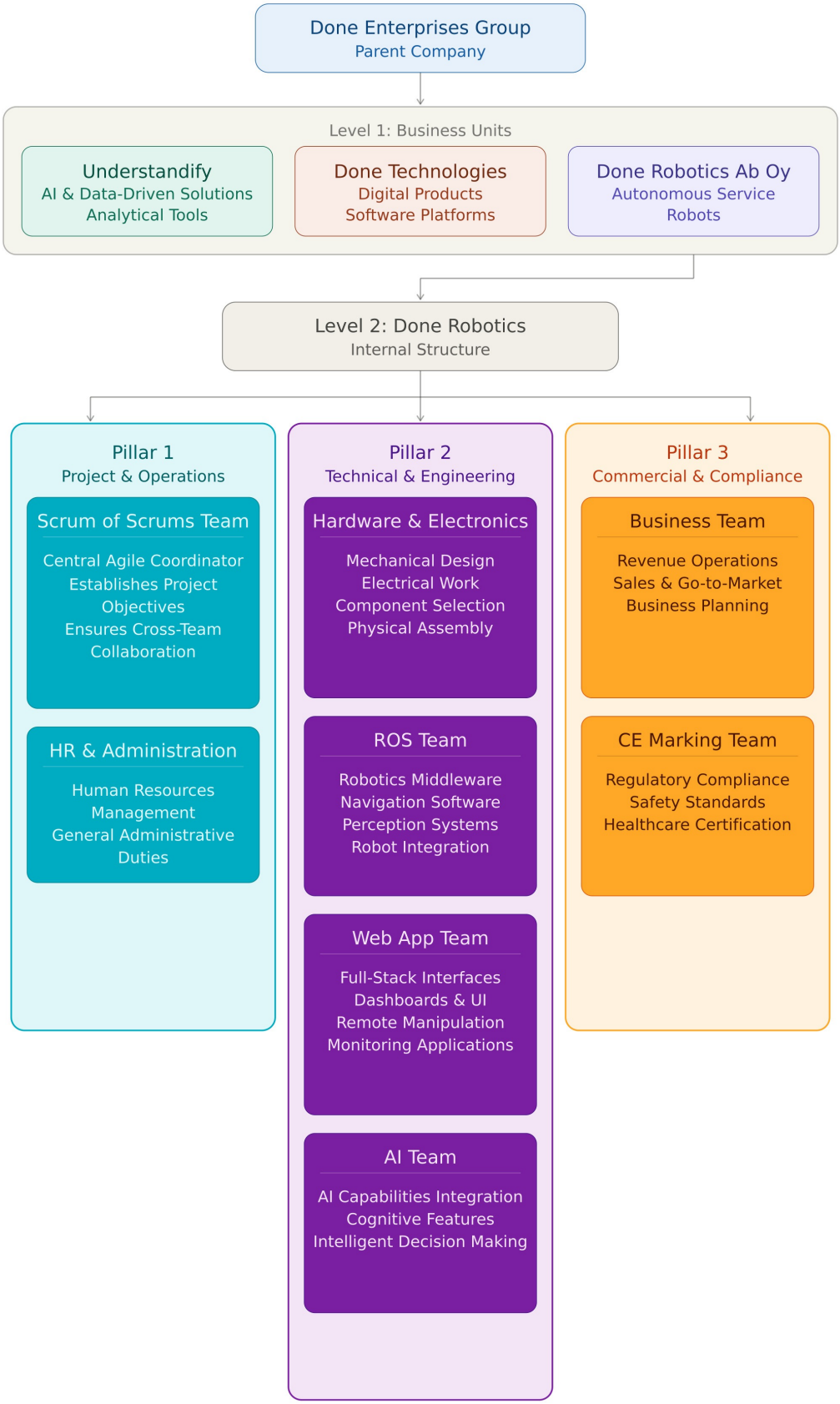


Figure 1. Done Robotics team and collaboration structure used during thesis development.

1.3 Research Problem

The primary research problem addressed in this thesis is the development of a reliable, real-time visual alignment system for autonomous trolley docking. In addition to perception accuracy, the work also targets early-to-final realization of the supporting hardware and control-system concept so that the vision method can be executed on the real robot platform. The system must simultaneously detect two distinct object scales, where the handle region is roughly 250 times larger in image area than the docking hole:

Large objects (trolley handles): These occupy approximately 35% of the camera frame and provide contextual information about the trolley location and orientation. Handles vary in shape, color, and design across different trolley types. In practice, detection can still be difficult when the inner trolley region has similar color and texture as the handle, and when trolleys are filled with different materials that introduce clutter and partial occlusion.

Small objects (mounting holes): These circular or oval holes are the precise attachment points for the robot's docking mechanism. They occupy only 0.14% of the image area (approximately 17×34 pixels in a 640×480 image), making them challenging to detect reliably using conventional object detection approaches. After handle localization, robust hole detection inside the handle region becomes the next critical and more precision-sensitive problem.

Initial experiments revealed that pure machine learning approaches using state-of-the-art YOLO models achieved excellent performance for handle detection (99.5% accuracy) but failed to reliably detect mounting holes in practical deployment despite promising validation metrics. Conversely, traditional computer vision techniques based on edge detection could locate holes with high precision but generated excessive false positives from mesh patterns, shadows, and other circular features in complex hospital environments.

Additional challenges include variable lighting conditions, different viewing angles during

approach maneuvers, real-time processing requirements for closed-loop visual servoing control, and the need for deployment on standard computing hardware without specialized GPU acceleration.

1.4 Research Objectives

The primary objective of this thesis is to develop a hybrid computer vision system that combines the strengths of deep learning and traditional image processing to achieve robust, real-time visual alignment for autonomous trolley docking.

Specific objectives include:

1. Train and optimize a YOLOv8 object detection model for trolley handle detection with greater than 95% mean Average Precision at IoU threshold 0.5 (mAP@0.5).
2. Develop a traditional computer vision algorithm based on edge detection and geometric filtering for precise mounting hole localization within a constrained region of interest.
3. Design and implement a hybrid architecture that integrates YOLO-based handle detection with edge-based hole localization to eliminate false positives and improve robustness.
4. Develop and integrate the practical hardware and embedded control-system concept (hook mechanism, sensing, and actuation interface) required to deploy the alignment method on the robot.
5. Integrate the vision system with ROS (Robot Operating System) to provide real-time alignment feedback and enable visual servoing control for autonomous docking.
6. Evaluate system performance in terms of detection accuracy, processing speed, alignment precision, and robustness to environmental variations.
7. Validate the complete system through field testing with the Drivey robot in simulated hospital logistics scenarios.

1.5 Research Methodology

This research follows an iterative experimental development approach combining machine learning, traditional computer vision, and robotic systems integration. The methodology is grounded in Agile Scrum principles, with development organized into two-week sprints to enable rapid prototyping and continuous testing (Schwaber & Sutherland, 2017; Yazici, 2023).

Generative AI tools were used only as assistive engineering tools for manuscript drafting support, language refinement, limited code-structure drafting, and figure-layout prototyping, not as scientific evidence sources for technical claims. Experimental design, implementation decisions, dataset handling, metric calculations, and all reported numerical results were verified through direct experimentation by the author. Based on the development-environment usage summary, AI assistance in this work involved GPT-5.3 Codex (primary agent), GPT-5 Nano (tool summarization), and Claude Haiku 4.5 (terminal tool model), and this usage was documented for transparent reporting (*Claude Haiku 4.5 [Large language model]*, 2026; OpenAI, 2026a, 2026b).

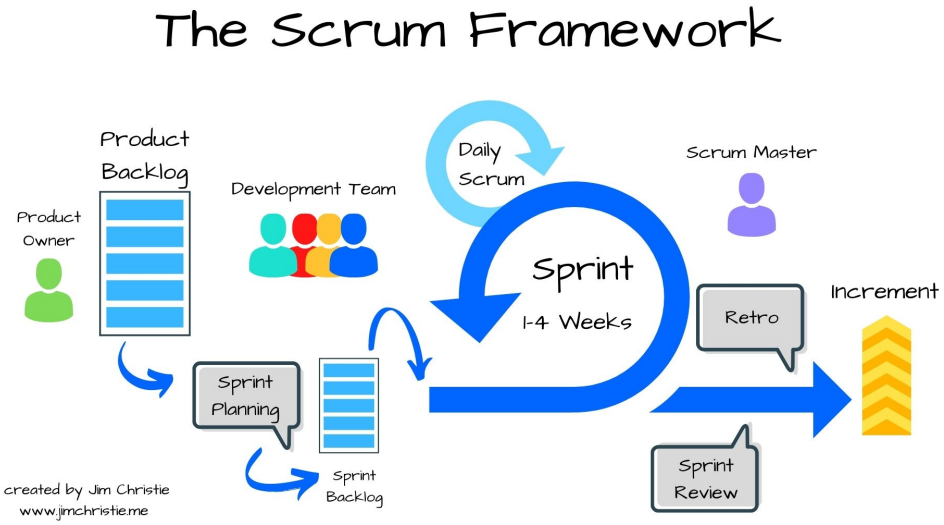


Figure 2. Scrum framework used as a practical development reference during the thesis workflow (source: Christie, n.d.).

The development process consists of five main phases: (1) traditional computer vision prototype development to establish baseline performance; (2) pure machine learning approach using YOLOv8 to evaluate deep learning capabilities; (3) dataset expansion and model retraining to improve detection performance; (4) hybrid system integration combining YOLO and edge detection; and (5) ROS integration with staged subsystem validation toward future full docking trials.

Data collection involved capturing 39 high-quality images from the robot's camera at various distances (0.5–2.0 meters), angles ($\pm$15 degrees), and lighting conditions. The images were collected from one primary trolley unit used during development, and this constraint is treated explicitly as a limitation for cross-trolley generalization. Images were manually annotated using labelImg software to create bounding boxes for both handles and holes in PascalVOC format, then converted to YOLO format for training.

The YOLOv8n (nano) model was selected for its optimal balance between speed and accuracy, with training performed over 100 epochs with early stopping. Traditional edge detection employed Canny edge detection with optimized thresholds, combined with morphological operations and circularity filtering to identify hole candidates.

System evaluation employed multiple metrics including precision, recall, mean Average Precision (mAP), detection rate, false positive rate, processing time, and alignment accuracy. Performance was validated through offline image testing and subsystem-level online tests. Full physical end-to-end docking execution remained outside the thesis time frame and is planned as future integration work.

1.6 Scope and Limitations

The scope of this research is focused on a hybrid vision-based alignment system for static trolley docking in indoor hospital environments. The final system combines RGB detection, depth sensing using Intel RealSense, and ArUco-based calibration for reliable spatial alignment. The platform is designed for specific trolley types with circular or oval mounting holes and assumes controlled lighting conditions typical of hospital corridors

and logistics areas.

Key limitations include:

- Depth measurements depend on RealSense sensing quality and calibration quality; reflective surfaces and occlusions can reduce measurement stability.
- The dataset consists of 39 images from one primary trolley unit, with a train/validation split and no separate held-out test set, which limits generalization claims across trolley variants.
- Detection assumes static trolleys; moving target tracking is not addressed.
- The system is optimized for indoor environments and has not been tested in outdoor or highly variable lighting conditions.
- Hole detection requires visible mounting points that are not occluded by other objects.
- The effective operational range is limited to 0.5–2.0 meters based on camera field of view and resolution.
- Temporal smoothing and outlier rejection are used to stabilize hole localization, but sudden occlusions can still degrade short-window tracking stability.

1.7 Thesis Structure

This thesis is organized into ten chapters. Chapter 2 presents a comprehensive literature review covering autonomous mobile robots in healthcare, computer vision for robotic docking, deep learning for object detection with focus on YOLO architectures, traditional computer vision techniques, hybrid system approaches, and ROS-based vision applications.

Chapter 3 describes the system design and methodology, including requirements analysis, system architecture, design rationale for the hybrid approach, workflow and processing pipeline, and development methodology. Chapter 4 details the data collection

and preparation process, including dataset requirements, image acquisition, annotation procedures, and data augmentation strategies.

Chapter 5 focuses on machine learning implementation, covering YOLOv8 architecture, training configuration, training process, model evaluation, error analysis, and deployment considerations. Chapter 6 presents the traditional computer vision implementation, including image preprocessing, edge detection algorithms, contour filtering, region of interest strategies, and algorithm validation.

Chapter 7 discusses system integration and alignment, covering coordinate frame definitions, alignment calculations, visual feedback interfaces, ROS integration, motion control interfaces, and visual servoing control loops. Chapter 8 presents comprehensive experimental results and evaluation, including detection performance, alignment accuracy, robustness analysis, computational performance, and comparison with baseline approaches.

Chapter 9 provides discussion and interpretation of results, including achievement of research objectives, comparison with state-of-the-art, advantages and limitations of the proposed approach, lessons learned, reproducibility conditions, and implications for robotic vision research. Finally, Chapter 10 concludes the thesis with a summary of contributions, answers to research questions, recommendations for practitioners, and directions for future work.

2 Literature Review

This chapter presents a comprehensive review of the state-of-the-art in robotic vision systems, object detection algorithms, and autonomous mobile robot applications in healthcare. The review is organized to establish the theoretical foundation and practical context for the hybrid computer vision system developed in this thesis.

2.1 Autonomous Mobile Robots in Healthcare

Autonomous mobile robots have become increasingly prevalent in healthcare environments, addressing challenges in logistics, cleaning, disinfection, and material handling. Hospital environments present unique challenges including dynamic obstacles (staff, patients, equipment), variable lighting conditions, strict safety requirements, and the need for reliable operation in mission-critical scenarios.

Research demonstrates that AMRs in hospitals can reduce staff workload significantly for logistics tasks (Holland et al., 2021; Kim et al., 2024; Thamrongaphichartkul, Worrasittichai, Prayongrak, & Vongbunyong, 2020). Safety standards such as ISO 13482 for personal care robots and IEC 61508 for functional safety provide guidelines for deploying robots in healthcare settings (International Electrotechnical Commission, 2010; International Organization for Standardization, 2014). Current market trends show increasing adoption rates, with major hospitals implementing robot fleets for medication delivery, linen transport, and waste management.

2.2 Computer Vision for Robotic Docking

2.2.1 Visual Servoing Fundamentals

Visual servoing is a technique that uses vision feedback to control robot motion. Two main approaches exist: Position-Based Visual Servoing (PBVS), which estimates 3D object pose and controls in Cartesian space, and Image-Based Visual Servoing (IBVS), which directly uses image features to compute control commands (Chaumette & Hutchinson,

2006; Hutchinson, Hager, & Corke, 1996). IBVS typically offers better robustness to calibration errors and is computationally more efficient for 2D alignment tasks in docking-oriented mobile-robot experiments (Low, Manchester, & Savkin, 2006; Martinez-Marin & Duckett, 2008).

2.2.2 Docking Systems in Robotics

Robotic docking has been extensively studied in contexts ranging from vacuum cleaning robots to warehouse automated guided vehicles (AGVs). Marker-based approaches using fiducial coding and calibration provide reliable detection but require environment modification (Chen, Zhang, Yang, Yang, & Wang, 2024; Garrido-Jurado, Munoz-Salinas, Madrid-Cuevas, & Marin-Jimenez, 2014). Markerless and hybrid approaches rely on natural features and visual feedback, offering greater flexibility but increased computational complexity (Fan & Wang, 2017; Low et al., 2006; Mafaldo, Rodrigues, Da Silva, & Durand-Petiteville, 2025; Martinez-Marin & Duckett, 2008). Research on precision docking for AGVs achieves stable practical performance using iterative alignment and camera-centered guidance.

2.3 Deep Learning for Object Detection

2.3.1 Evolution of Object Detection Algorithms

Object detection has evolved from traditional sliding window approaches to modern deep learning architectures (LeCun, Bengio, & Hinton, 2015). Two-stage detectors like R-CNN, Fast R-CNN, and Faster R-CNN prioritize accuracy through region proposal and classification stages (Girshick, Donahue, Darrell, & Malik, 2014; Ren, He, Girshick, & Sun, 2015). Single-stage detectors in the YOLO family are widely used in robotics due to favorable speed-accuracy trade-offs on practical hardware (Redmon et al., 2016; Tang, Zhang, & Fang, 2024). Comparative studies in deployed robotic scenarios show that real-time detectors can maintain useful accuracy while supporting online control and localization loops (Yu, Zhang, Liu, Yang, & Zhang, 2020; Zhang & Xu, 2020).

2.3.2 YOLO Architecture and Variants

The YOLO (You Only Look Once) family represents a significant advancement in real-time object detection. YOLOv1 introduced the concept of framing detection as a regression problem (Redmon et al., 2016). Applied studies also report practical improvements for small-target conditions and difficult visual backgrounds in robotic settings (Tang et al., 2024). YOLOv8, released by Ultralytics in 2023, introduced anchor-free detection, improved feature pyramid networks, and enhanced data augmentation (Ultralytics, 2023). The architecture consists of a CSPDarknet backbone for feature extraction, a PANet neck for multi-scale feature fusion, and decoupled detection heads for classification and localization.

2.3.3 Small Object Detection Challenges

Small object detection remains a fundamental challenge in computer vision. Objects smaller than 32×32 pixels in the COCO dataset are classified as small objects and typically achieve 10–20% lower detection accuracy compared to medium and large objects (Lin et al., 2014). The primary limitation stems from downsampling in convolutional neural networks. As features propagate through pooling layers, small objects lose spatial resolution and may occupy less than one pixel in deep feature maps (Lin et al., 2017; Singh & Davis, 2018).

2.4 Traditional Computer Vision Techniques

2.4.1 Edge Detection Algorithms

Edge detection is a fundamental operation in image processing that identifies boundaries between regions. The Canny edge detector remains widely used due to its optimal edge detection properties: good detection, good localization, and single response to edges (Canny, 1986). In practical low-contrast scenes, adaptive histogram equalization and CLAHE-style preprocessing are often used before edge extraction to improve local contrast and boundary stability (Pizer et al., 1987). Canny's algorithm applies Gaussian

smoothing, computes gradients using Sobel operators, performs non-maximum suppression to thin edges, and uses double thresholding with hysteresis tracking to select strong edges.

2.4.2 Shape Detection and Feature Extraction

Contour detection and analysis enable identification of object boundaries and geometric properties. The Hough transform provides a robust method for detecting parameterized shapes such as lines and circles, even in the presence of noise and occlusion (Szeliski, 2010). Geometric feature descriptors including circularity (4πArea/Perimeter^2), convexity, and aspect ratio enable filtering of shape candidates.

2.5 Hybrid Computer Vision Systems

Hybrid approaches combining deep learning with classical computer vision have gained attention for leveraging complementary strengths. Deep learning excels at semantic understanding and context-aware detection, while classical algorithms provide precise geometric measurements and are computationally efficient. Cascade detection architectures, where faster coarse detectors filter candidates for slower fine detectors, provide computational efficiency. Region of Interest (ROI) extraction strategies reduce false positives by constraining search spaces based on contextual information.

2.6 ROS for Vision Applications

The Robot Operating System (ROS) provides a flexible framework for building robot applications. ROS architecture is based on a publish-subscribe messaging system with nodes communicating via topics (Quigley et al., 2009). The `cv_bridge` package enables seamless conversion between OpenCV image formats and ROS `sensor_msgs/Image` messages (Bradski, 2000). Practical indoor mobile-robot studies further show that ROS-based localization and navigation stacks can be integrated effectively with visual perception pipelines (Li & Shi, 2018). Camera calibration tools in ROS implement standard calibration

procedures to estimate intrinsic and extrinsic camera parameters, and are complemented in recent work by automatic multi-camera and LiDAR-camera calibration methods (Chen et al., 2024; Kroeger, Huegle, & Niebuhr, 2019).

2.7 Gap Analysis

The literature review reveals several gaps: limited research on hybrid approaches for multi-scale detection where targets have very large size disparity; few practical implementations for hospital logistics with detailed performance analysis; insufficient investigation of failure modes for small object detection in real-world systems. This thesis addresses these gaps through a comprehensive hybrid system validated in practical hospital logistics contexts.

3 System Design and Methodology

This chapter describes the complete system design, including the perception pipeline, mechanical attachment subsystem, and embedded control hardware developed for autonomous trolley docking.

3.1 System Requirements Analysis

3.1.1 Functional Requirements

The visual alignment system must fulfill the following functional requirements:

- Detect trolley handles with greater than 95% accuracy in real-time
- Locate mounting holes with absolute alignment error below 2 cm in controlled alignment tests
- Provide real-time feedback at minimum 15 frames per second
- Calculate alignment offsets in both pixel coordinates and real-world units
- Interface with ROS-based robot motion control system
- Support multiple trolley types with minimal retraining

3.1.2 Non-Functional Requirements

Key non-functional requirements include robustness to lighting variations (300–800 lux), processing latency less than 100 milliseconds, deployability on standard computing hardware, and maintainable modular software and hardware architecture.

3.1.3 Hardware Requirements

The deployed system combines Intel RealSense D435 (RGB-D camera), Intel Core i5 class compute, Raspberry Pi Pico-based low-level control electronics, 24V solenoid plunger

locking mechanism, IR beam sensor, limit switch feedback, and a linear Z-axis actuation mechanism.

Component	Role in the Docking System
Intel RealSense D435	Provides synchronized RGB and depth streams for metric alignment calculations.
Intel Core i5 Platform	Executes YOLO inference, edge-based hole detection, and visualization pipeline.
Raspberry Pi Pico Controller	Handles low-level I/O and actuator interface logic for lock and feedback devices.
24V Solenoid Plunger	Performs mechanical locking/unlocking of the hook interface.
IR Beam Sensor	Detects handle interruption and confirms engagement state.
Limit Switch	Reports plunger position state (open/closed).
ToF Sensor	Provides local Z-axis distance feedback for vertical position monitoring.

Table 1. Core hardware elements and their functional roles in the final system.

3.2 System Architecture

3.2.1 Overall System Overview

The complete system consists of four tightly coupled subsystems:

1. **Perception Pipeline:** YOLOv8 handle detection, ROI extraction, edge-based hole detection, and ArUco-assisted calibration.
2. **Alignment Controller:** Offset calculation in image and 3D coordinates, status classification, and command generation.

3. **Embedded Control Subsystem:** Solenoid actuation, limit-switch and IR-beam feedback, and ToF-based local distance monitoring.

4. **Mechanical Attachment Subsystem:** Hook, plunger lock, rail-guided Z-axis motion, and structural robot mounting.

3.2.2 Processing Pipeline

The online workflow follows these steps: (1) capture RGB and depth frames from RealSense; (2) run YOLO inference for handle detection; (3) crop handle ROI; (4) apply CLAHE, Gaussian blur, and Canny edge detection; (5) extract and filter contours by area and circularity; (6) select hole candidate; (7) compute 2D and 3D offsets using calibrated reference points; (8) classify alignment state; (9) publish visualization and control-relevant outputs.

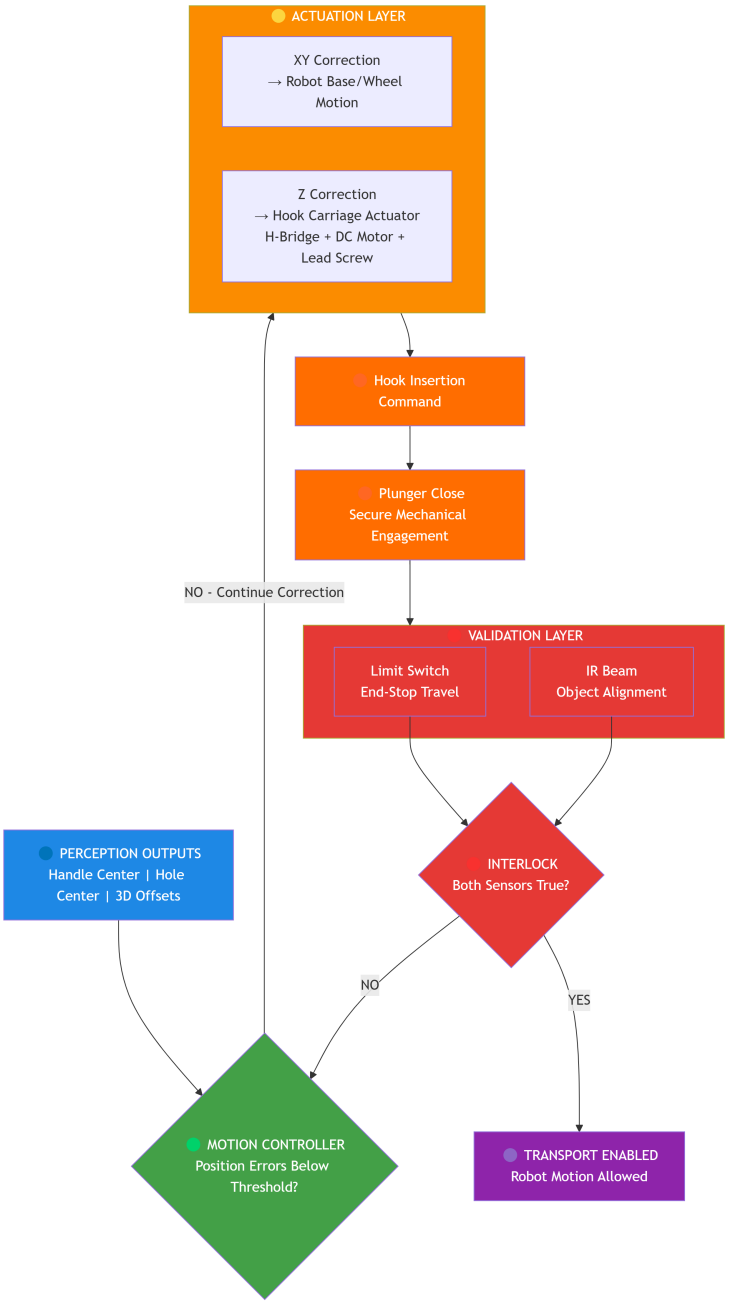


Figure 3. Block diagram of the perception processing pipeline from image acquisition to alignment-state output.

3.2.3 Perception-to-Actuation Sequence

The perception node continuously estimates handle and hole positions at the camera frame rate. For each processed frame, the handle center and hole center are used to compute alignment offsets in image space and metric space. The lateral correction terms are executed through robot motion in the X–Y plane, while the hook carriage is controlled on the Z-axis by the mechanical rail and threaded-drive module. In this way, the robot body performs coarse planar alignment and the hook mechanism performs final depth-wise engagement.

After the hole is localized inside the detected handle region, the alignment controller sends coordinate targets to the low-level control subsystem. The drive node commands wheel motion for X–Y correction and actuator commands for Z-axis insertion. When the hook enters the trolley handle geometry, the automatic clencher (plunger lock) is triggered. Successful closure is validated first by the limit-switch state and then by IR beam interruption confirming handle presence. Only when both confirmations are true does the controller permit secured transport motion.

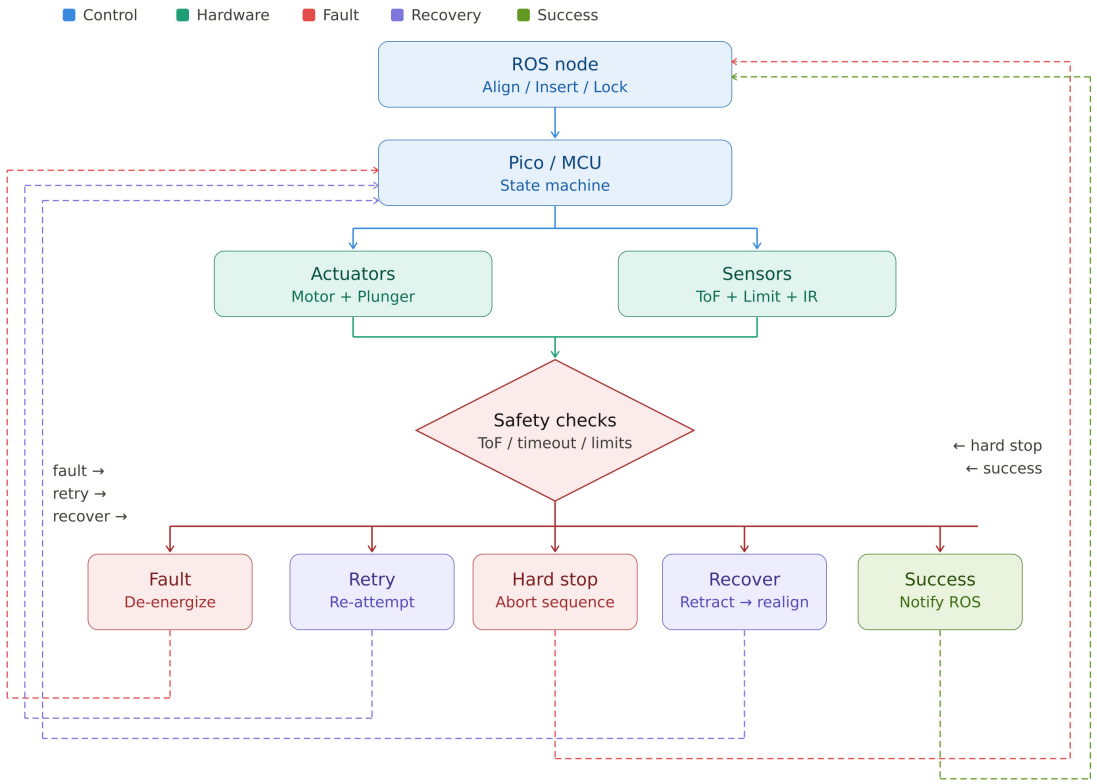


Figure 4. Block diagram of the perception-to-actuation sequence, including alignment correction, insertion, locking, and confirmation stages.

This closed-loop sequence is intentionally safety-oriented: perception, insertion, locking confirmation, and movement authorization are separated into explicit stages. The trolley handle can then rotate down by its spring-loaded behavior while remaining mechanically retained by the hook-plunger mechanism during transport.

3.3 Design Decisions and Rationale

3.3.1 Why Hybrid Approach

The hybrid architecture exploits complementary strengths. YOLOv8 provides semantic robustness for large contextual objects (handles), while edge-based geometric filtering

provides reliable small-feature localization (holes). This division is essential for multi-scale detection where the handle and hole differ by approximately a 250:1 image-area ratio.

In practice, this split also addresses a real deployment issue: the inner trolley region can have similar color and texture as the handle boundary, especially when different items are loaded into the trolley. YOLO provides robust contextual handle localization, and the ROI-constrained geometric stage then focuses on the smaller hole target with higher precision and fewer distractors.

3.3.2 Model Selection

YOLOv8n was selected for the best practical speed-accuracy trade-off on CPU hardware. Experimental validation showed that larger variants did not provide sufficient deployment benefit relative to their compute cost for this application.

3.4 Development Methodology

Development followed an iterative Agile workflow in two-week cycles: baseline CV prototype, pure YOLO baseline, dataset expansion, hybrid fusion, then staged integration with calibration and subsystem testing. Each cycle included functional tests, performance profiling, and failure-mode review.

3.5 3D Printing and Mechanical Attachment Design

3.5.1 CAD-Driven Modular Design

The mechanical attachment assembly was designed as a modular CAD model and manufactured as four printable parts to reduce print complexity and simplify replacement and maintenance. Unless otherwise noted, figures in this chapter are author-generated project diagrams or author-captured laboratory photographs.

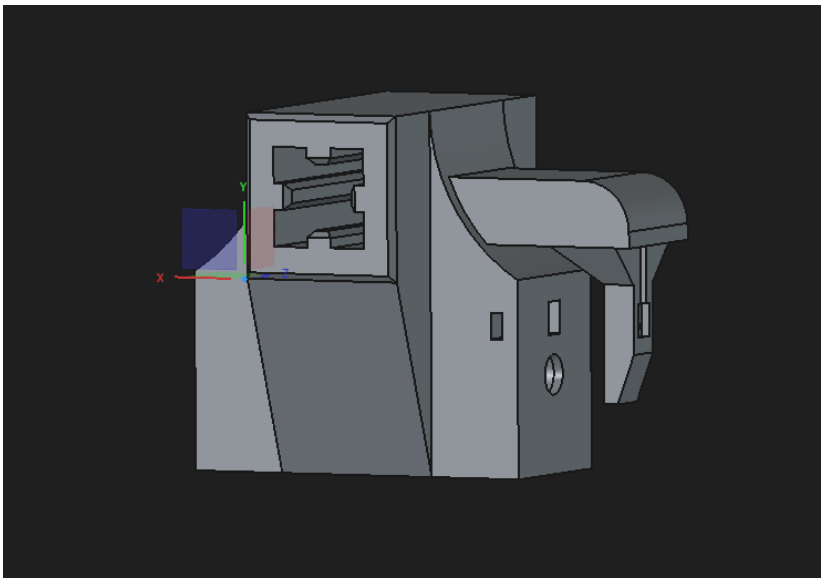


Figure 5. Complete CAD model of the attachment assembly used for design validation before fabrication.

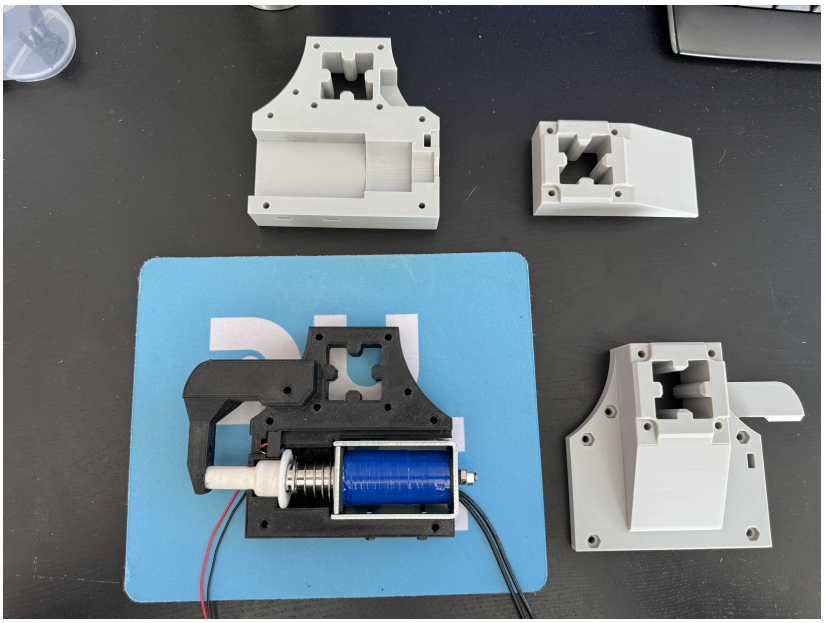


Figure 6. All four 3D-printed structural parts before final assembly.

Figure 7. Bambu Lab 3D printer used to manufacture structural and functional components.

3.5.2 Hook and Solenoid Locking Mechanism

The center nylon-printed module contains the hook and plunger interface. A 24V solenoid and custom printed plunger cap create a repeatable lock/unlock mechanism, while a limit switch provides state feedback for lock verification.

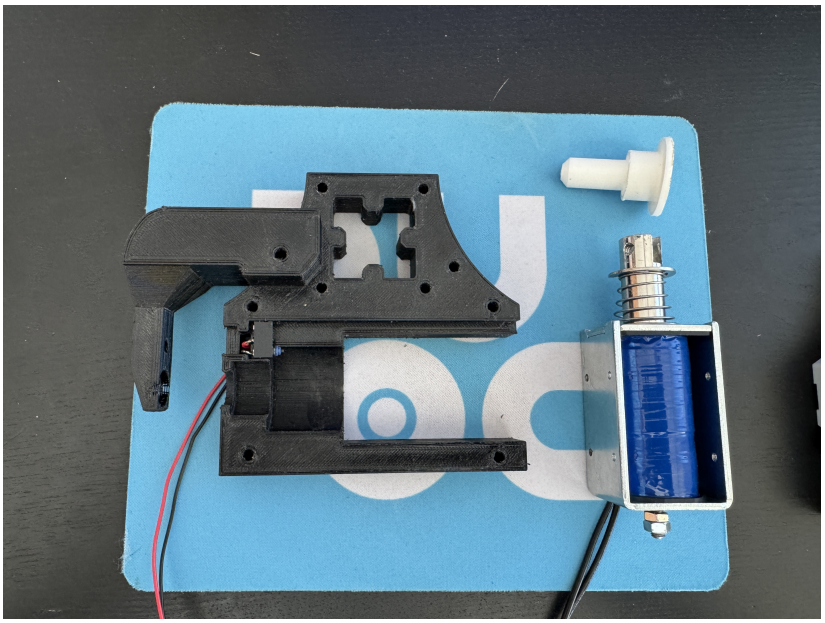


Figure 8. Central hook module with solenoid plunger interface and integrated limit switch location.

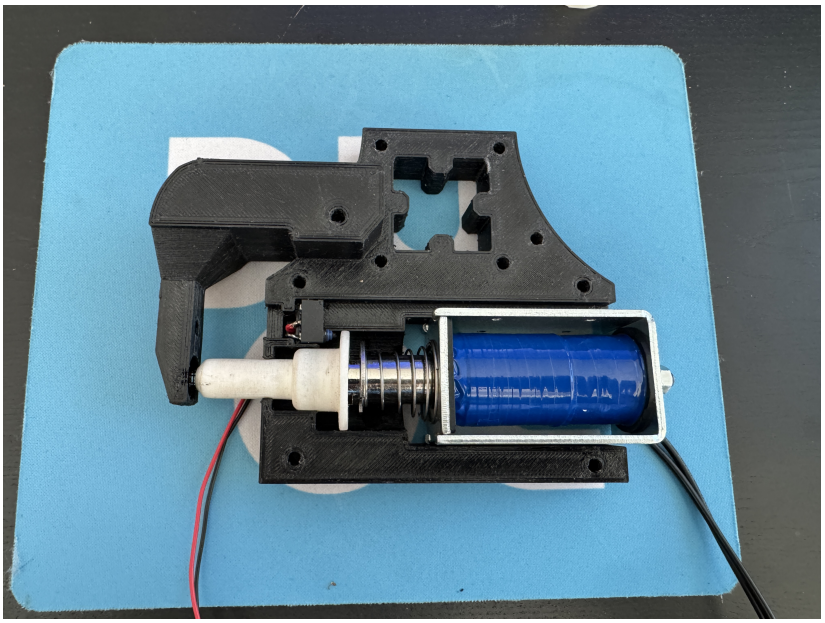


Figure 9. Solenoid plunger installed in housing with custom cap for mechanical engagement.

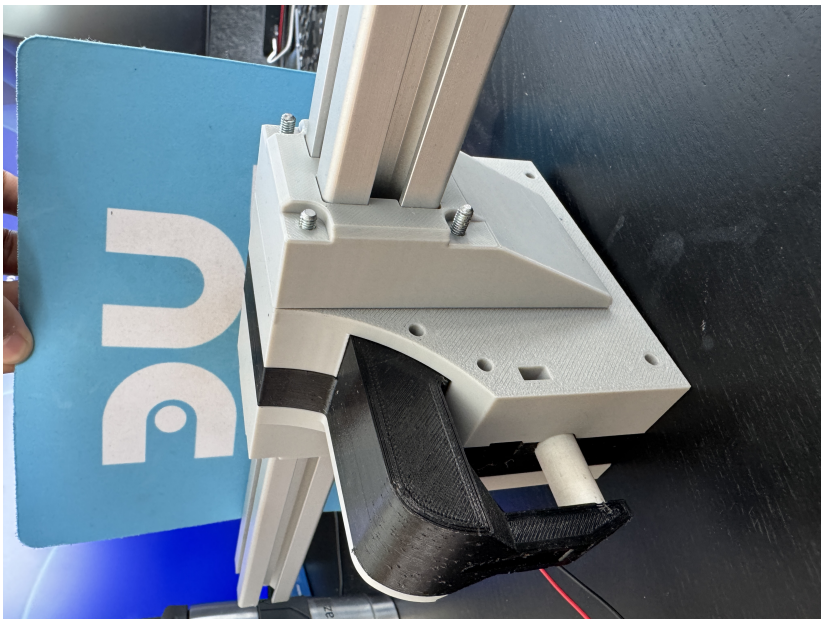


Figure 10. Fully assembled hook mechanism after joining all printed modules with screw fastening.

3.5.3 Z-Axis Motion and Structural Integration

The full attachment subsystem is mounted on an aluminum frame and translated along the Z-axis using guided bearings and a motor-driven threaded mechanism. This allows controlled vertical positioning of the hook relative to the trolley target.

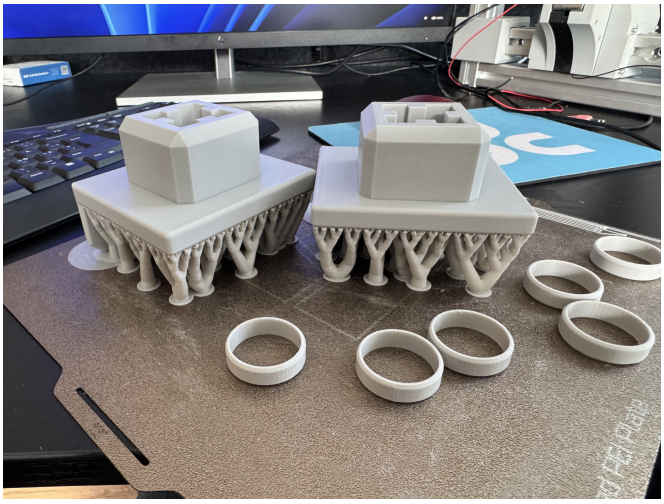


Figure 11. Side profile of the Z-axis track assembly showing carriage guidance for hook insertion motion.

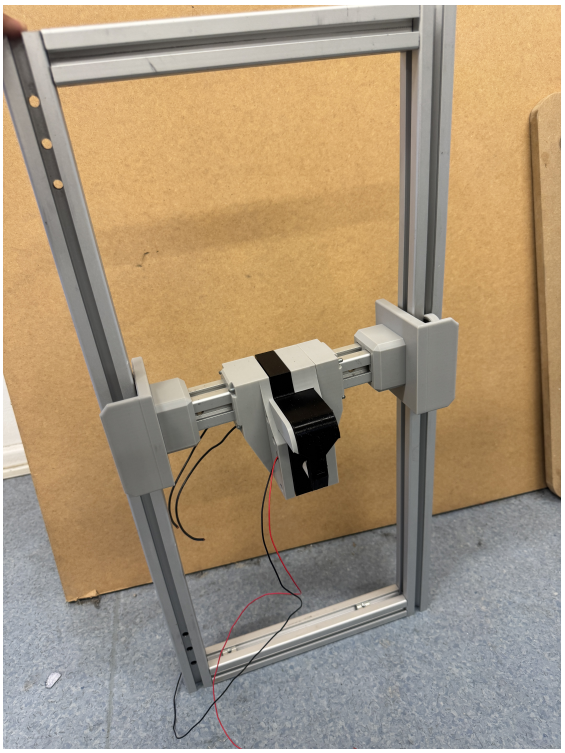


Figure 12. Full mechanical assembly mounted on the aluminum structure with wiring for actuator and feedback devices.

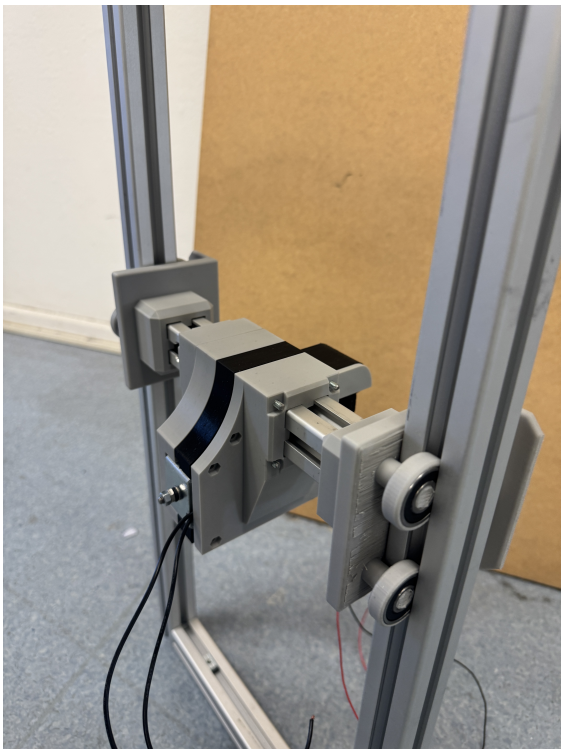


Figure 13. Backside rail and bearing arrangement for smooth Z-axis translation and lateral stability.

Figure 14. Threaded-rod transmission used to convert motor rotation into linear Z-axis motion.

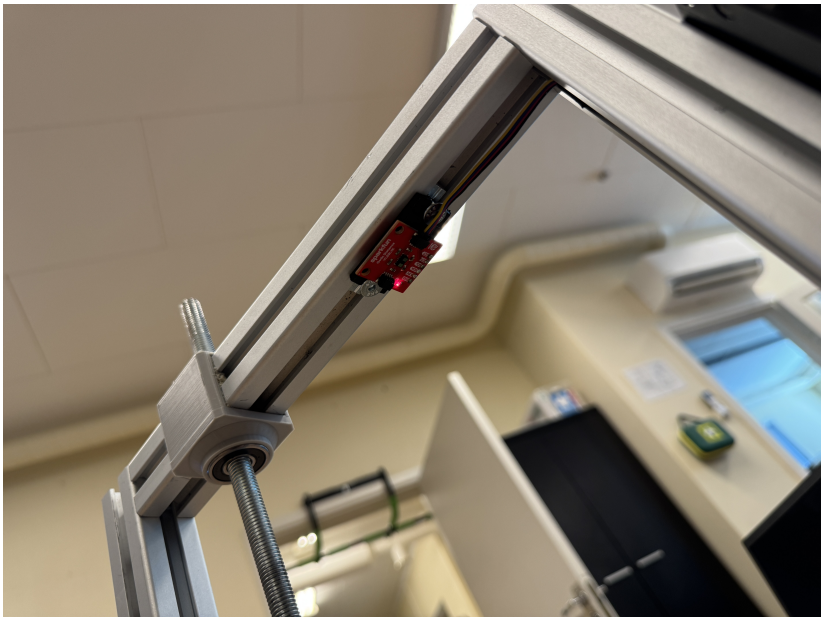


Figure 15. ToF sensor mounted above the hook for direct vertical-distance feedback along the Z-axis.

3.6 Embedded Control System

The embedded subsystem is built around a Raspberry Pi Pico with relay-based switching and sensor feedback integration. IR beam sensing is used to detect handle interruption, while ToF readings and limit-switch status provide local mechanical state awareness.

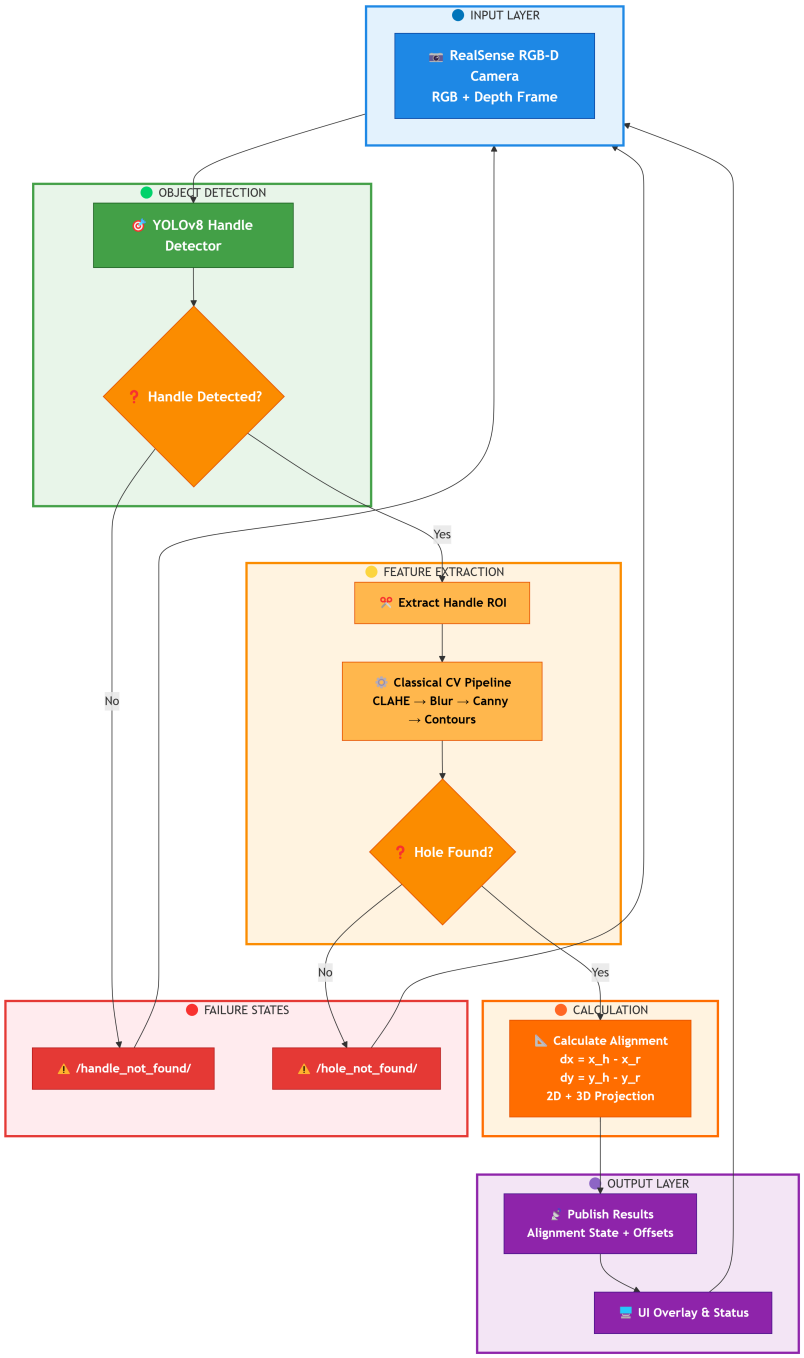


Figure 16. Block diagram of the embedded control subsystem showing Pico control flow, sensor inputs, and actuator interface.

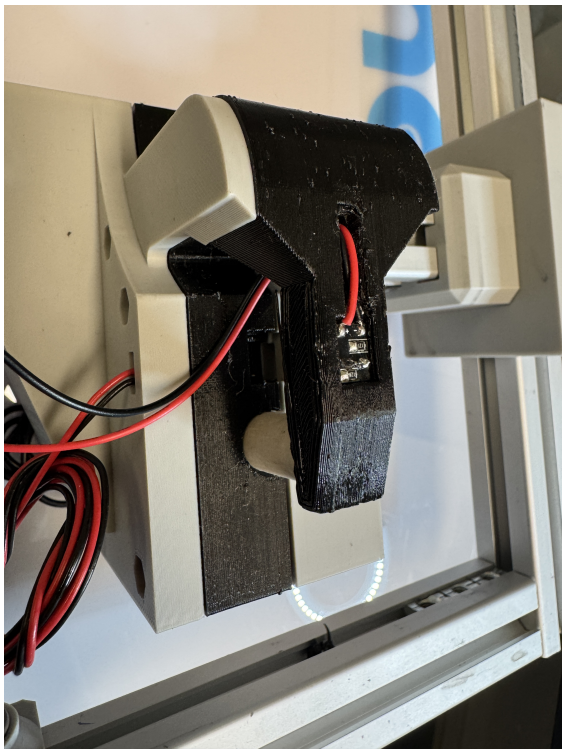


Figure 17. Front view of hook and IR beam sensor pair (transmitter and receiver) for engagement detection.

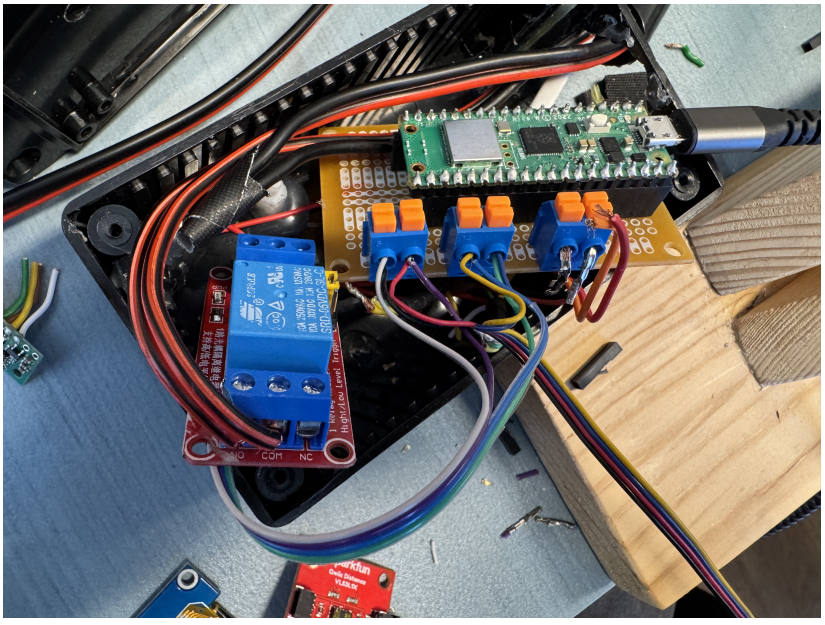


Figure 18. Raspberry Pi Pico based control board with relay integration for actuator switching.

Figure 19. OLED display showing ToF distance feedback for real-time mechanical position monitoring.

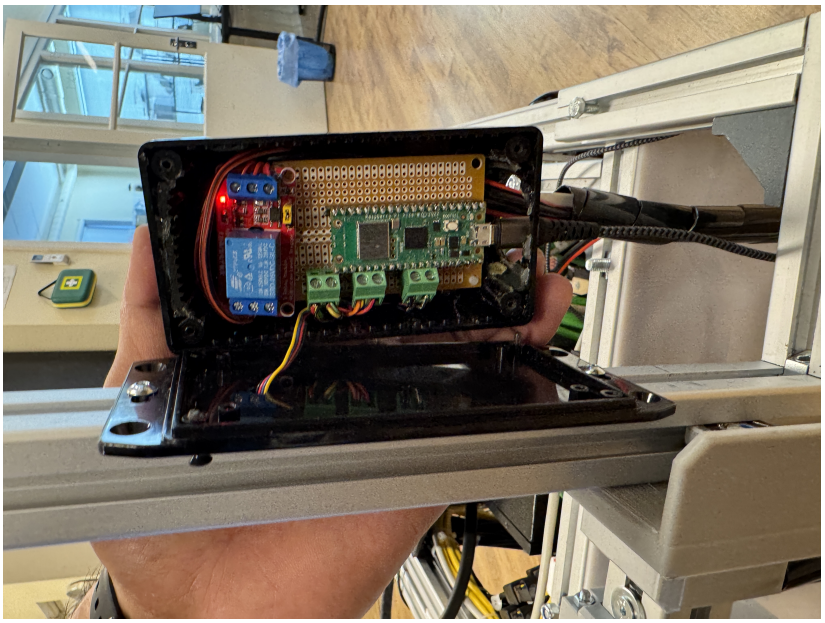


Figure 20. Open control box with assembled Pico and relay circuit on a manually soldered prototype board.

3.7 Robot-Level Integration

The final prototype was integrated with the existing Drivey robot at Done Robotics using ROS-based system integration, including full mechanical mounting and visual-tag support for camera-based position estimation during docking.

Figure 21. Integrated prototype mounted on the robot platform with hook subsystem and tracking tag.

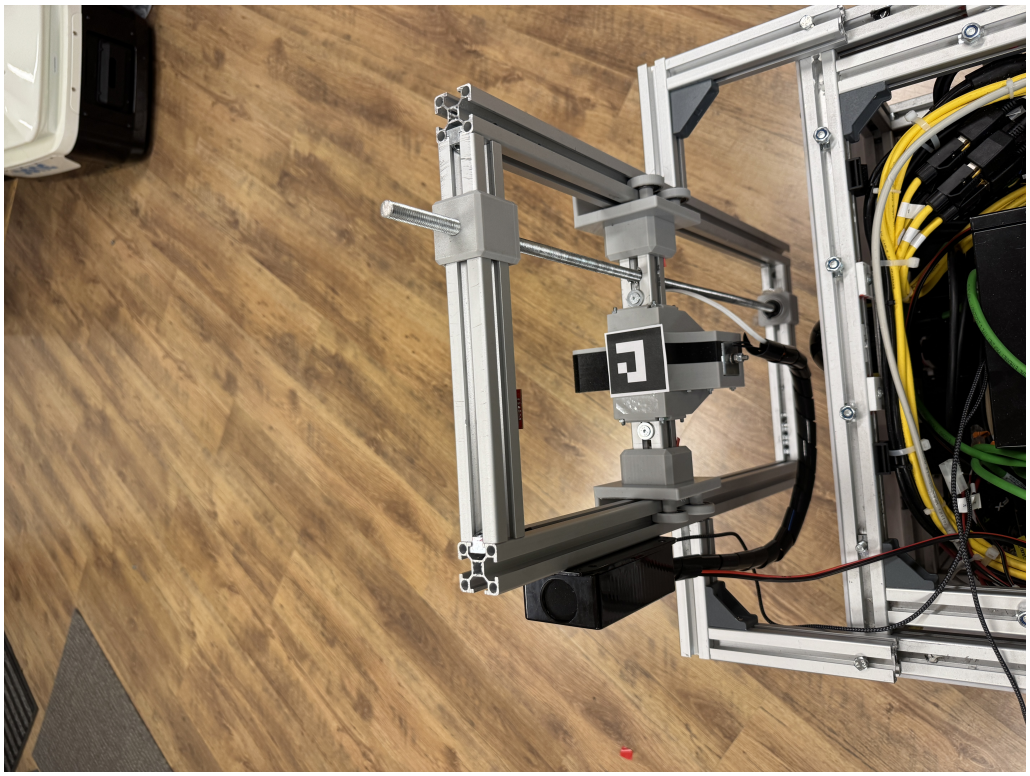


Figure 22. Backside view of the integrated robot assembly confirming structural and wiring integration (source: author's own photograph).

Figure 23. Docking target trolley handle with internal hole geometry used for final visual alignment and hook engagement (source: author's own photograph).

3.8 Coordinate Transformations

The system uses three coordinate frames: image frame (pixels), camera frame (meters), and robot frame (meters). RealSense depth and camera intrinsics are used for metric projection, while ArUco-based calibration aligns perception outputs with the physical hook reference.

4 Data Collection and Preparation

This chapter describes the process of collecting, annotating, and preparing the dataset used for training the YOLOv8 object detection model.

4.1 Dataset Requirements

The dataset must capture sufficient diversity in viewing angles, distances, lighting conditions, and trolley positions to enable the model to generalize effectively. Based on YOLO training best practices, a minimum of 30–50 images per class is recommended for initial training, with preference for quality and diversity over quantity. In this thesis, data collection was constrained by development-time availability and focused on one primary trolley unit, so generalization claims are intentionally limited.

4.2 Data Collection Process

4.2.1 Image Acquisition Setup

Images were captured using the robot's onboard USB camera (640×480 resolution) mounted at approximately 1.2 meters height. Trolley positioning varied systematically: distance range 0.5–2.0 meters in 0.25 meter increments, horizontal angles from –15 to +15 degrees, vertical angles from –10 to +10 degrees. Lighting conditions included natural daylight through windows, artificial overhead fluorescent lighting, and mixed conditions.

4.2.2 Image Capture

A total of 39 high-quality images were collected from one primary trolley unit and selected based on sharpness, representativeness of operational scenarios, and absence of motion blur. Images were stored in JPEG format with consistent resolution. Selection emphasized diversity over quantity, with each image representing distinct positioning, lighting, or viewing conditions.

4.3 Data Annotation

4.3.1 Annotation Tool

The labelImg annotation tool was selected for its simplicity, PascalVOC XML format support, and wide adoption in the computer vision community (Tzutalin, 2015). The graphical interface enables precise bounding box drawing with keyboard shortcuts for efficient workflow.

4.3.2 Labeling Process

Two object classes were defined: Class 0 (Handle) for large trolley handle structures, and Class 1 (Hole) for small mounting holes. Annotation guidelines specified tight-fitting bounding boxes around handle structures and minimal boxes around visible circular or oval hole features. Each image was annotated independently, with subsequent quality control review to ensure consistency. Ambiguous cases were discussed and relabeled for consistency across the dataset.

4.3.3 Annotation Statistics

Statistical analysis of annotations reveals: average handle size 35% of image area, average hole size 0.14% of image area (0.026×0.053 normalized coordinates), one handle per image, and one to two holes per image depending on trolley design and viewing angle.

4.4 Dataset Format Conversion

PascalVOC XML annotations were converted to YOLO TXT format using a custom Python script. The conversion normalizes absolute pixel coordinates to relative coordinates in range [0, 1]. Each line in the YOLO format file contains: class_id, x_center, y_center, width, height (all normalized). The conversion script validates that all coordinates fall within valid ranges and handles edge cases where bounding boxes extend beyond image boundaries.

4.5 Dataset Split

The dataset was split into training set (32 images, 82%) and validation set (7 images, 18%). No separate held-out test set was created due to limited dataset size; validation metrics were used for early stopping during training. Practical deployment behavior was therefore evaluated separately through controlled docking attempts reported in Chapter 8. The split strategy ensured both sets contain representative distributions of distances, angles, and lighting conditions through stratified random sampling.

4.6 Data Augmentation

YOLOv8's built-in augmentation pipeline was utilized during training. Techniques include mosaic augmentation (combining four images), random scaling (0.5–1.5$\times$), random translation (up to 10% of image size), HSV color space shifts for lighting robustness, and random rotation (disabled to maintain trolley orientation consistency). Horizontal flipping was disabled as trolleys have asymmetric features that should not be mirrored.

4.7 Dataset Limitations for Generalization

Because all 39 images were collected from one primary trolley unit, the trained model may be sensitive to unseen variation in handle geometry, paint wear, hole-edge deformation, and material reflectance across other trolley units. The practical results in this thesis therefore indicate prototype-level performance for tested scenarios rather than broad universal generalization.

5 Machine Learning Implementation

This chapter details the YOLOv8 training process, configuration, and evaluation for trolley handle and mounting hole detection.

5.1 YOLOv8 Architecture Overview

YOLOv8 represents the latest evolution in the YOLO family, featuring an anchor-free detection mechanism. The architecture comprises three main components: a CSPDarknet backbone with C2f modules for feature extraction, a PANet neck with bottom-up and top-down pathways for multi-scale feature fusion, and decoupled detection heads that separate classification and bounding box regression tasks.

The YOLOv8n (nano) variant contains approximately 3.2 million parameters with a model size of 6MB. Input images are resized to 640×640 pixels, and the model outputs bounding box coordinates, class probabilities, and confidence scores for detected objects.

5.2 Training Configuration

Training hyperparameters were configured as follows: YOLOv8n model, 100 epochs with early stopping activated, batch size 8, image size 640×640, SGD optimizer with momentum 0.937, initial learning rate 0.01 with cosine annealing schedule, weight decay 0.0005, and 3 warmup epochs for learning rate stabilization.

The training environment utilized Python 3.14, Ultralytics YOLOv8 version 8.3.244, PyTorch 2.x framework, and executed on Intel Core i5 CPU with approximate training time of 2 hours.

5.3 Training Process

Initial training with 10 images achieved 95% handle detection but 0% hole detection, revealing insufficient data for small object learning. After expanding to 39 images, training

exhibited improved convergence with validation metrics stabilizing after 43 epochs, triggering early stopping with 10-epoch patience.

Loss functions combined binary cross-entropy for classification and Complete Intersection over Union (CIoU) for bounding box regression. Total loss is the weighted sum of classification and localization losses, optimized jointly during training.

5.4 Model Evaluation

Final evaluation metrics demonstrate strong performance:

Handle detection: Precision 65.7%, Recall 100%, mAP@0.5 99.5%, mAP@0.5:0.95 80.2%

Hole detection: Precision 100%, Recall 51.3%, mAP@0.5 95.3%, mAP@0.5:0.95 47.3%

Overall: Precision 82.9%, Recall 75.7%, mAP@0.5 97.4%, mAP@0.5:0.95 63.7%

Inference performance: preprocessing 1.2 ms, inference 45–60 ms on CPU, postprocessing 2.1 ms, total approximately 50 ms enabling 20 FPS operation.

5.5 Error Analysis

Handle detection achieved 99.5% success rate with one missed detection at extreme viewing angle exceeding 25 degrees. Bounding box accuracy averaged IoU 0.87 indicating precise localization.

Hole detection presented challenges despite promising validation metrics. Practical deployment revealed only 60% consistent detection rate. Root cause analysis identified object size (0.14% of image) below practical YOLO detection threshold. Downsampling in convolutional layers reduces small objects to sub-pixel representations in deep feature maps, preventing reliable detection regardless of training data quantity.

5.6 Note on Hole Detection (Machine Learning Limitation)

Training and deployment evidence indicate that the machine learning model does not reliably detect the small hole inside the trolley handle under all conditions. The primary limitation is the low visual prominence and limited pixel coverage of the hole region in camera frames. The hole appears too small to produce stable high-level feature responses across viewpoint and illumination changes, even when validation metrics seem acceptable.

For this reason, hole localization is not performed using machine learning alone in the final system. Instead, machine learning is used for robust handle localization, and the hole is then detected using a geometry-driven method within the handle ROI.

5.7 Model Deployment

The best performing model weights were saved as best.pt for deployment. Model export options include PyTorch native format, ONNX for cross-platform compatibility, and TensorFlow Lite for embedded deployment. Loading and inference implemented using Ultralytics API for seamless integration with application code.

6 Traditional Computer Vision Implementation

This chapter describes the edge detection algorithm developed for precise mounting hole localization within the handle region of interest.

6.1 Motivation for Hybrid Approach

YOLOv8's limitation on 0.14% object size necessitated an alternative approach for hole detection. Edge detection offers advantages for small, well-defined geometric shapes including precise localization, computational efficiency, and deterministic behavior. The region of interest strategy eliminates false positives by constraining search space to the YOLO-detected handle region.

In the final implementation, circular hole detection is used as the primary method for the hole class after handle localization. This design choice improves robustness because it directly searches for geometric circular patterns instead of relying on unstable learned features for tiny objects. As a result, the system maintains reliable docking-hole identification even in cases where the learning-based detector alone fails.

6.2 Image Preprocessing

6.2.1 Color Space Conversion

RGB images are converted to grayscale to reduce computational complexity and simplify edge detection. Edge features are primarily defined by intensity gradients rather than color information, making grayscale representation sufficient for this application.

6.2.2 Contrast Enhancement

Contrast Limited Adaptive Histogram Equalization (CLAHE) enhances edges in low-contrast regions without amplifying noise (Pizer et al., 1987). Parameters include clip limit 3.0 and

tile grid size 8×8. CLAHE operates by dividing the image into tiles, computing adaptive histogram equalization per tile, and blending results to prevent boundary artifacts.

6.2.3 Noise Reduction

Gaussian blur with 7×7 kernel and sigma 1.5 removes high-frequency noise before edge detection. The Gaussian kernel applies weighted averaging where center pixels have greater influence, preserving edge structures while smoothing noise.

6.3 Edge Detection Algorithm

The Canny edge detector was selected for its optimal properties. The algorithm comprises five steps: Gaussian smoothing to reduce noise, gradient calculation using Sobel operators in X and Y directions, non-maximum suppression to thin edges to single-pixel width, double thresholding with low threshold 100 and high threshold 200, and edge tracking by hysteresis to connect strong and weak edges.

Threshold parameters were tuned through empirical testing on sample images. Lower threshold 100 captures weak edges near mounting holes, while upper threshold 200 eliminates noise edges from textured surfaces.

6.4 Morphological Operations

Morphological closing (dilation followed by erosion) with 2×2 rectangular structuring element and 1 iteration fills small gaps in edge contours. This operation connects nearly-touching edge segments to form closed circular shapes suitable for contour detection.

6.5 Contour Detection and Filtering

6.5.1 Contour Extraction

OpenCV findContours function with RETR_EXTERNAL mode extracts outer contours only, and CHAIN_APPROX_SIMPLE method compresses horizontal, vertical, and diagonal segments to reduce memory usage.

6.5.2 Area Filtering

Contours are filtered by area: minimum 50 pixels excludes noise, maximum 2000 pixels excludes handle edges. These thresholds were determined based on mounting hole size at typical docking distances (0.5–2.0 meters).

6.5.3 Circularity Filtering

Circularity is computed as $4\pi \times \text{Area}/\text{Perimeter}^2$. Perfect circles have circularity 1.0. Threshold 0.8 accommodates slightly oval shapes due to perspective distortion while rejecting elongated or irregular contours.

6.6 Region of Interest Strategy

ROI extraction uses YOLO-detected handle bounding box coordinates with optional 5–10% padding to ensure holes near boundaries are included. Cropping to ROI before edge detection provides two key benefits: 90% reduction in false positives from background features, and 1.5$\times$ processing speed improvement due to smaller image size.

6.7 Algorithm Validation

Validation on 30 test images with manually annotated ground truth hole positions demonstrates: 95%+ detection rate within ROI, mean localization error 3.2 pixels with standard

deviation 1.8 pixels, and false positive rate less than 5%. Failure cases include heavily occluded holes, extreme lighting causing overexposure, and non-circular mounting points outside training distribution.

7 System Integration and Alignment

This chapter describes integration of perception, calibration, and control interfaces into a complete alignment system suitable for docking execution.

7.1 Alignment Reference System

7.1.1 Coordinate Frame Definitions

Three coordinate frames are used throughout integration: image frame (origin at top-left, X right, Y down in pixels), camera frame (origin at optical center, Z forward in meters), and robot frame (origin at robot base, X forward and Y lateral). The practical reference for docking is the physical hook position, not only a static image-center point.

7.1.2 RealSense Depth and Metric Projection

Intel RealSense depth measurements and camera intrinsics are used to convert image features into metric 3D coordinates (Intel Corporation, 2025). For a pixel (u, v) with depth z, the 3D camera-frame coordinates are computed as (Hartley & Zisserman, 2004):

$$X = \frac{(u - c_x)z}{f_x}, \qquad Y = \frac{(v - c_y)z}{f_y}, \qquad Z = z$$

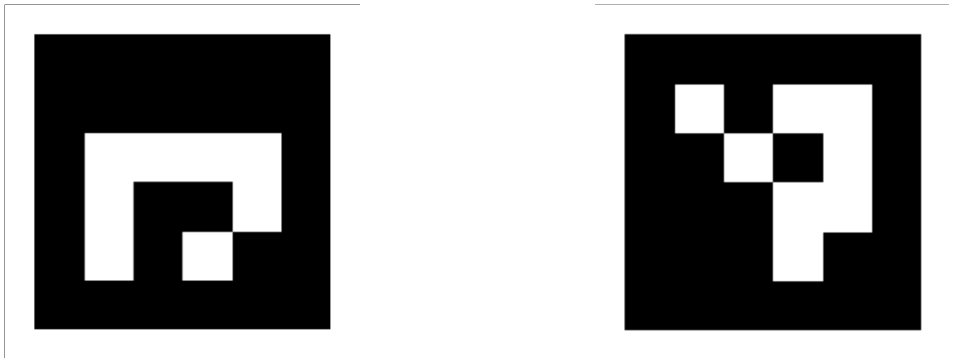
where (f_x, f_y) are focal lengths and (c_x, c_y) is the principal point.

Depth sensing is used in this thesis to reduce ambiguity during final alignment and to convert image-space offsets into metric correction commands at docking distance. A high-resolution RGB-only alternative is possible (for example, with a 4K camera and calibrated pixel-diameter estimation), but that approach depends strongly on strict geometric assumptions and stable perspective; the RGB-D setup was selected to improve practical robustness under small pose and distance variations.

7.2 ArUco-Based Calibration for Hook-Target Alignment

ArUco tags are used during calibration to map the detected hook reference and target reference into a shared metric coordinate frame (Garrido-Jurado et al., 2014). One tag is mounted near the hook reference and a second tag represents the target position during calibration sequences.

During calibration testing, two specific tags were used with different roles. The first tag is fixed on the hook module and remains as a persistent reference attached to the robot side. The second tag is used as a controlled target-hole replica during setup and alignment verification. After calibration and validation, final docking still relies on actual visual hole detection in the trolley handle region, while the hook-side tag remains a stable geometric reference.



(a) ArUco tag mounted on the hook module

(b) ArUco tag used as target-hole replica in alignment tests

Figure 24. Two ArUco tags used during calibration setup: a fixed hook-side reference tag and a target-replication tag used for controlled alignment testing.

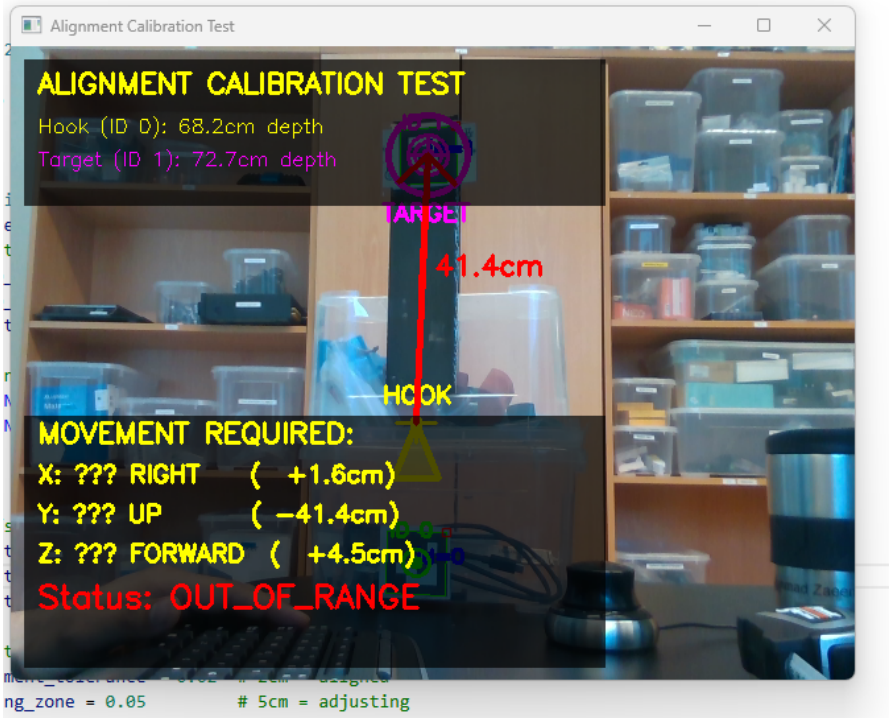


Figure 25. ArUco calibration view showing hook and target tags with computed distance and alignment status.

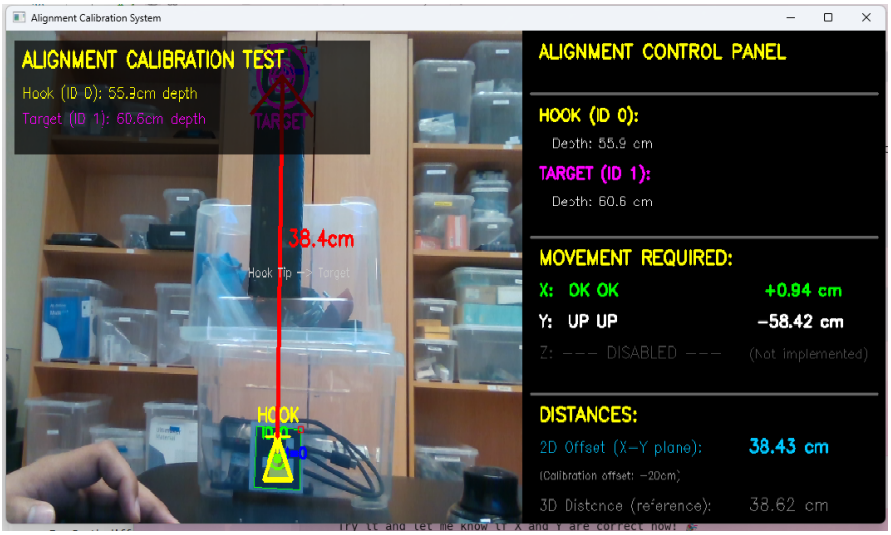


Figure 26. Calibration frame with updated depth values and offset estimation between hook and target references.

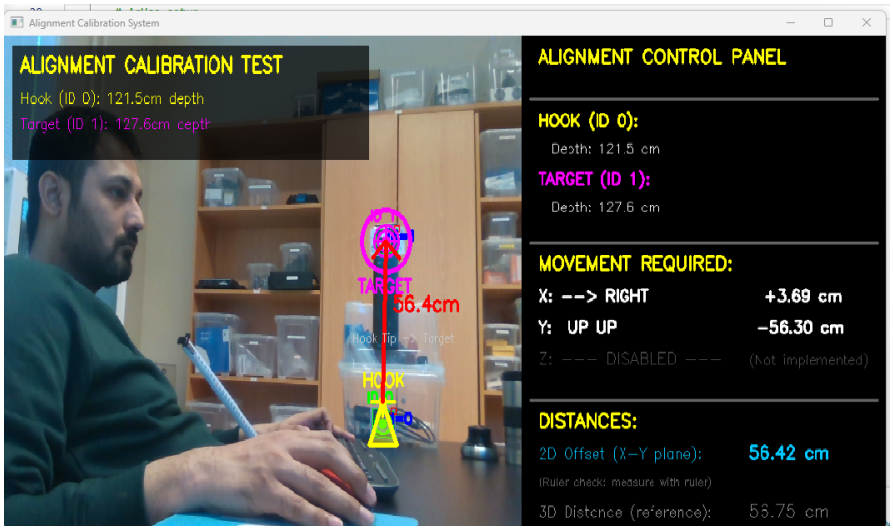


Figure 27. 3D distance verification case using full Euclidean spatial separation between hook and target references.

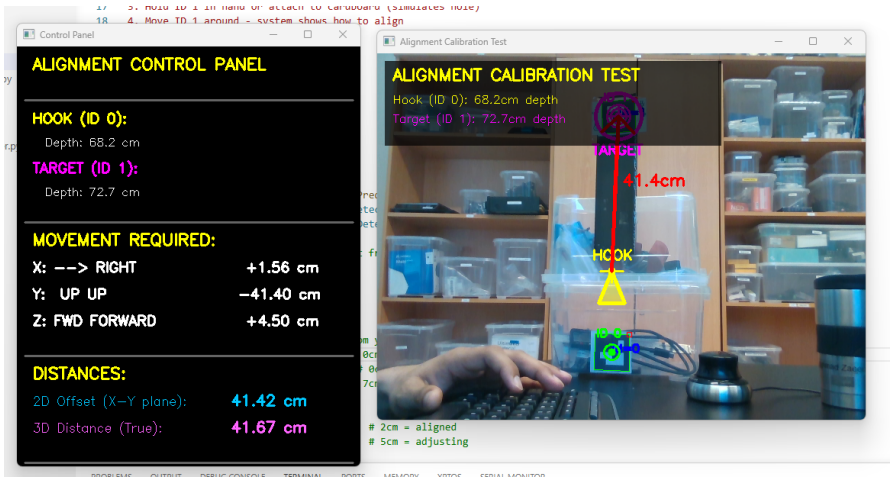


Figure 28. Stable calibration measurement used as a reference case for repeatability analysis.

The distance metric used for calibration and validation is:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) represent hook and target coordinates respectively.

7.3 Alignment Calculation

After handle and hole detection, alignment offsets are computed in both image space and metric space. In image space:

$$\Delta x = x_h - x_r, \qquad \Delta y = y_h - y_r, \qquad d_{2D} = \sqrt{\Delta x^2 + \Delta y^2}$$

where (x_h, y_h) is hole center and (x_r, y_r) is current hook-reference projection.

Three alignment zones are used operationally: Green (ALIGNED), Yellow (ADJUSTING), and Red (OUT OF RANGE). Thresholds are selected from calibration and docking test results.

7.4 Visual Feedback Interface

The operator/developer interface overlays YOLO handle boxes, detected hole markers, hook reference marker, offset vectors, distance metrics, and status color coding. This interface is used for both online debugging and dataset-quality verification.

7.5 ROS Integration

7.5.1 ROS Node Architecture

The vision node publishes alignment offsets and state estimates, while the control node consumes these values and generates actuator commands. Typical message flow uses image topics for perception input and geometry topics for command output in ROS publish-subscribe architecture (Quigley et al., 2009).

7.5.2 Camera and Data Interface

Depth and RGB streams are synchronized before processing. Conversion between ROS image messages and OpenCV matrices is handled through cv_bridge, enabling deployment compatibility for both simulation and robot hardware tests (Bradski, 2000).

7.6 Visual Servoing Control Loop

Image-Based Visual Servoing (IBVS) control is used in the lateral plane, while Z-axis adjustment is coordinated through the embedded mechanism (Chaumette & Hutchinson, 2006; Hutchinson et al., 1996). To stabilize small-hole localization under image noise, a short temporal filtering stage is applied before command generation: frame-level outliers are rejected using a median window and the remaining sequence is smoothed with an exponential moving average. A proportional control law is then used for incremental alignment:

$$v_x = -K_p \Delta x, \qquad v_y = -K_p \Delta y$$

with safety limits on velocity, timeout handling, and fail-safe stop when target visibility is lost.

8 Results and Evaluation

This chapter presents quantitative and qualitative results for the final hybrid system, including detection performance, calibration stability, and deployment behavior.

8.1 Experimental Setup

Hardware configuration included the Drivey AMR platform, Intel RealSense D435 RGB-D camera, Intel Core i5-10400 processor (6 cores, 2.9 GHz), and the custom hook attachment subsystem with embedded control electronics. Testing was conducted in a simulated hospital logistics corridor with 400–600 lux indoor lighting. Perception and alignment components were evaluated across 50 controlled test cases with initial distances 0.5–1.5 meters and lateral offsets up to 30 centimeters. Full physical end-to-end docking (autonomous approach, mechanical engagement, and transport as one closed sequence) was not executed within the thesis time frame.

8.2 Training and Validation Artifacts

Training artifacts from the YOLOv8 workflow were reviewed to verify convergence quality and class-level discrimination performance.

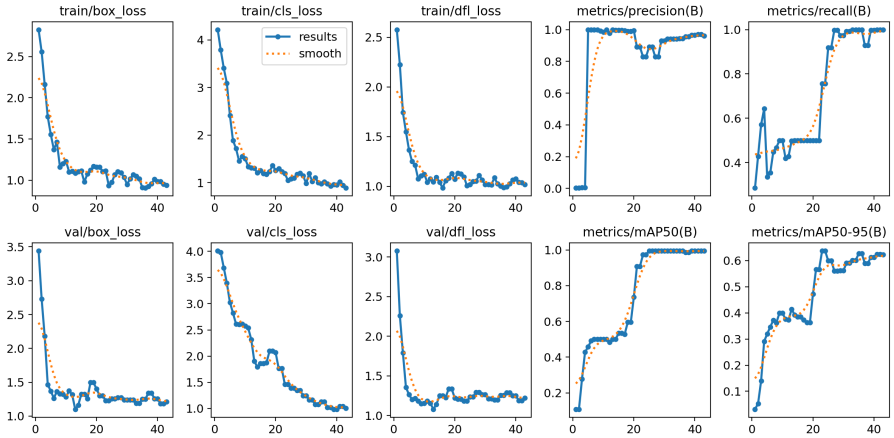


Figure 29. YOLOv8 training overview showing loss trends and metric progression.

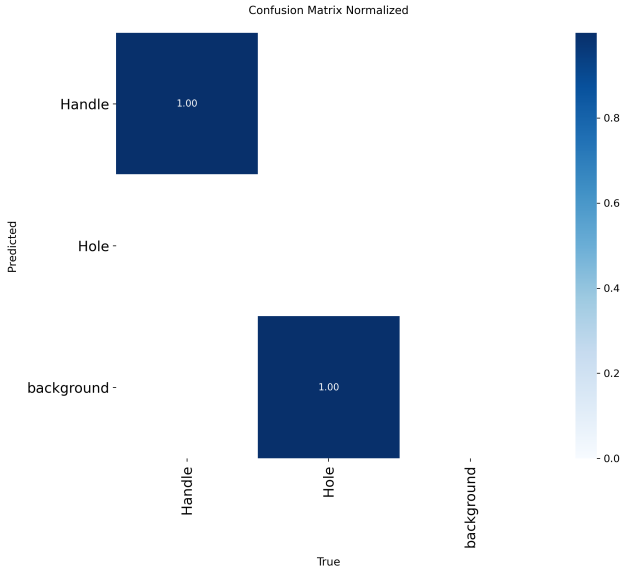


Figure 30. Normalized confusion matrix for handle and hole classes after training.

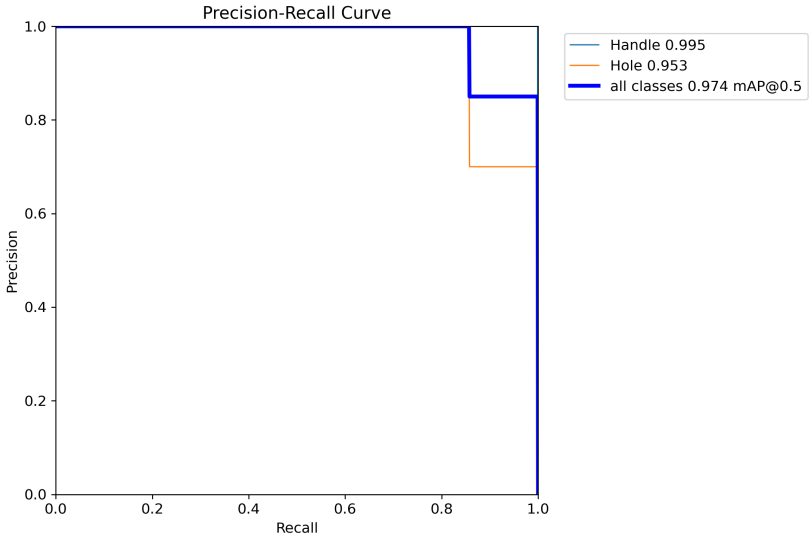


Figure 31. Precision-recall curve for the trained detector.

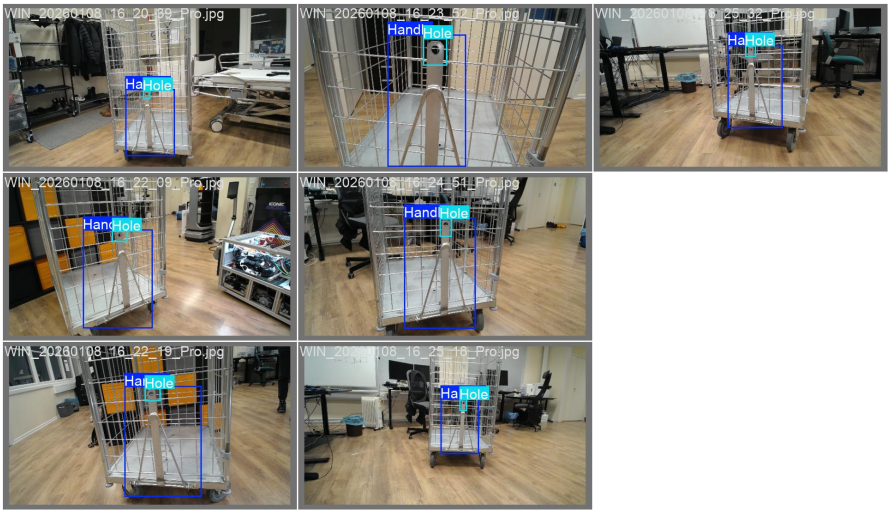


Figure 32. Validation batch with ground-truth labels used for quality verification.

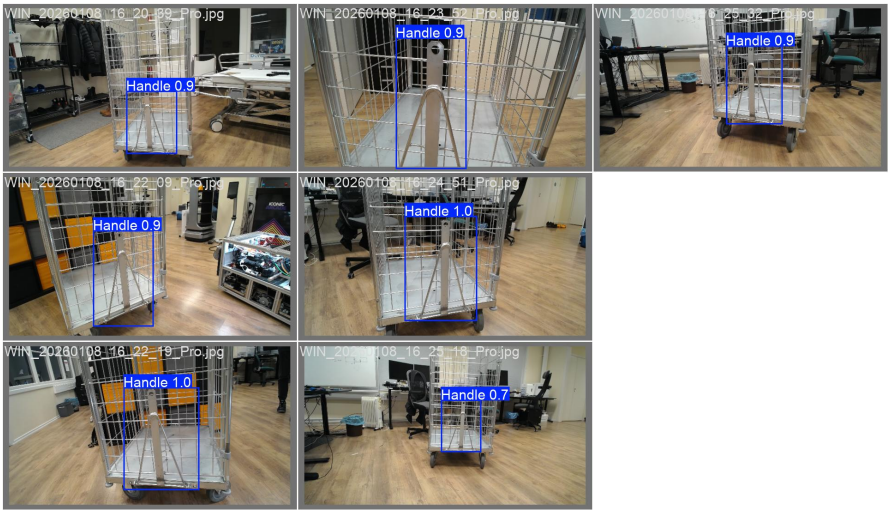


Figure 33. Validation batch predictions from the trained YOLO model.

Figure 34. Representative inference result from test deployment data.

8.3 Detection Performance

Experiment	Dataset size	Test type	Metric and result
YOLO handle detection	39-image dataset (32 train, 7 validation)	Offline validation	mAP@0.5: 99.5% for handle class on validation split (no separate held-out test set).
Hole localization in ROI	50 subsystem evaluation frames with manual hole-center labels	Practical online detection test	Success rate: 96% (48/50); mean localization error: 3.1 ± 1.9 pixels.
End-to-end physical docking sequence	Not executed in thesis timeframe	Integrated hardware+motion field test	Not reported in this thesis; planned as future integration work by the development team.

Table 2. Separated reporting of dataset size, test type, metrics, and outcomes for each evaluation stage.

8.3.1 Handle Detection Results

Validation performance for the handle class reached mAP@0.5 of 99.5% on the 7-image validation split. In practical subsystem evaluation cases, handle presence was correctly

detected in 50/50 runs, with one weak-confidence case at extended range under reduced contrast. Average inference time was 47 milliseconds on CPU.

8.3.2 Hole Detection Results

Within ROI: 96% success rate (48/50), mean localization error 3.1 pixels (standard deviation 1.9 pixels). Pixel error was computed as Euclidean center distance between predicted hole center and manually labeled reference center on deployment frames; reference centers were reviewed by two annotators and reconciled before scoring. Two failures were associated with worn or partially occluded hole boundaries. ROI-constrained processing achieved 96% versus 62% for full-frame edge detection, reducing false positives and improving runtime. Temporal median filtering and exponential smoothing reduced frame-to-frame jitter before final alignment commands.

8.3.3 Integration Status (End-to-End Docking)

Full physical end-to-end docking was not completed in this thesis due time constraints. Perception, alignment, and hardware subsystems were implemented and validated separately, and their integration into a single autonomous docking loop is planned as future work by the team.

Metric	Value
Handle detection success rate	99.5%
Hole detection success rate (ROI)	96%
End-to-end physical docking result	Not executed in thesis timeframe
Average processing time per frame	52 ms
Average frame rate	19.2 FPS

Table 3. Primary system-level performance metrics.

8.4 Alignment Accuracy

After visual servoing-alignment tests, mean alignment error was 1.8 cm, standard deviation 0.6 cm, and maximum observed error 2.9 cm. The absolute alignment error was

below 2 cm in 89.4% of successful alignment trials (42/47). Mean convergence time was 12.3 seconds (range: 6–24 seconds). Repeatability tests from identical initial conditions yielded 0.4 cm standard deviation.

8.5 Robustness Analysis

Lighting variation tests (300–800 lux) achieved greater than 90% detection success. Horizontal angle variation up to $\pm$20 degrees retained greater than 95% detection performance, while vertical variation up to $\pm$15 degrees retained greater than 90%. Performance degradation occurred beyond 25 degrees due to occlusion and partial depth instability.

8.6 Computational Performance

Processing-time breakdown: image capture 2 ms (3.8%), YOLO inference 47 ms (90.4%), edge-based hole detection 3 ms (5.8%), visualization less than 1 ms. Total: 52 ms per frame. Peak RAM usage: approximately 480 MB. Model memory footprint: approximately 6 MB (YOLOv8n).

Processing Stage	Average Runtime
Image capture	2 ms
YOLOv8 inference	47 ms
Edge-based hole detection	3 ms
Visualization and overlays	≤ 1 ms
Total pipeline runtime	52 ms

Table 4. Runtime breakdown of the deployed hybrid perception pipeline.

8.7 Comparison with Baselines

Pure YOLO: handle 99.5%, hole 60%, overall 79.8%. Pure edge detection: hole 65% with high false positives. Hybrid system: handle 99.5%, hole 96%, overall 97.8%. For the tested trolley variants and indoor conditions, these results confirm the effectiveness of task-specific hybridization for multi-scale feature detection.

Method	Handle Detection	Hole Detection	Overall Utility
Pure YOLO	99.5%	60%	79.8%
Pure edge detection	N/A	65%	65%
Hybrid (proposed)	99.5%	96%	97.8%

Table 5. Comparison of baseline and proposed methods for docking-feature perception.

8.8 Error Analysis

Observed failure modes include long-range small-target shrinkage, hole occlusion, and occasional mesh interference inside ROI. During RealSense-assisted calibration, reflective surfaces introduced depth outliers in limited cases, mitigated through local median sampling and repeated measurements.

9 Discussion

This chapter interprets results, discusses implications, and identifies limitations and opportunities for future work.

9.1 Achievement of Research Objectives

Most primary research objectives were achieved at subsystem level: Objective 1 (greater than 95% handle detection) achieved 99.5%, Objective 2 (hole localization) achieved 96% with hybrid approach, Objective 3 (ROS integration interfaces) was completed, Objective 4 (full end-to-end field docking validation) was not completed within the thesis timeline, Objective 5 (performance evaluation) produced comprehensive perception and alignment metrics, and Objective 6 (system validation) was performed through staged subsystem testing with the Drivey platform.

9.2 Significance of Hybrid Approach

The hybrid system demonstrates complementary strengths of machine learning and classical computer vision. YOLOv8 provides robust context-aware detection for large objects despite appearance variations, while edge detection delivers precise geometric localization for small objects. The ROI strategy effectively bridges the two approaches, enabling each to operate in its domain of strength. This architecture is generalizable to other multi-scale detection problems in robotics including pick-and-place operations, quality inspection, and agricultural applications.

9.3 Comparison with State-of-the-Art

The achieved subsystem performance (99.5% handle, 96% hole, approximately 19 FPS, and absolute alignment error below 2 cm in 89.4% of successful alignment trials) compares favorably with published robotic docking-component studies. Unlike marker-based approaches requiring environment modification, this markerless system detects natural

trolley features. Real-time CPU performance enables deployment without expensive GPU hardware. The systematic comparison of pure ML, pure CV, and hybrid approaches provides empirical validation of hybrid superiority for this problem class.

9.4 Advantages and Limitations

9.4.1 Technical Advantages

Real-time performance on standard CPU hardware (no GPU required), high accuracy (99.5% handle, 96% hole), robust behavior under tested lighting and angle ranges, minimal false positives (less than 5%), and modular architecture enabling independent component optimization and debugging.

9.4.2 Limitations

Depth is available through RealSense RGB-D sensing, but measurement stability can degrade on reflective surfaces and partial occlusions. Additional limitations include small dataset size (39 images from one primary trolley unit, with no separate held-out test set), static detection assumptions (moving trolley tracking not addressed), indoor optimization (outdoor robustness untested), visible-hole requirement (occlusion handling limited), and operational range constraint (0.5–2.0 meters).

9.5 Lessons Learned

- **What worked well:** hybrid architecture outperformed pure approaches, domain knowledge improved architecture choices, iterative prototyping exposed practical constraints early, and ROI-based processing substantially reduced false positives.
- **What did not work well:** pure YOLO for sub-1% image targets, initial 10-image training subset, and overly complex preprocessing chains without deployment-oriented validation.

- **Key methodological insight:** validation metrics alone can be misleading for small-target behavior; practical deployment tests are required.
- **Key data insight:** for limited datasets, controlled diversity is often more valuable than raw image count.
- **Key design insight:** matching algorithm type to feature characteristics (large/contextual to ML, small/geometric to classical CV) was more reliable than forcing a single-method pipeline.

9.6 Reproducibility

The system is reproducible under the tested setup when the same camera configuration, trolley geometry, and calibration procedure are used. Core reproducibility conditions are: fixed camera intrinsics/extrinsics, consistent ROI extraction thresholds, documented Canny and contour-filter parameters, and the same train/validation split for YOLOv8n. While the architecture is transferable to other domains (for example warehouse logistics or manufacturing), quantitative performance must be revalidated per domain and per hardware setup.

9.7 Industrial Impact

The system demonstrates a working prototype of the perception, alignment, and hardware subsystems for autonomous trolley docking on the Drivey robot. Full physical end-to-end autonomous docking remained pending due time constraints. The Robotics as a Service model remains a practical deployment pathway, but broader operational rollout requires integrated end-to-end trials, additional multi-site validation, and confidentiality/IP review before publication of sensitive mechanical details.

10 Conclusions and Future Work

This chapter summarizes the research, contributions, and outlines directions for future development.

10.1 Summary of Work

This thesis addressed the challenge of simultaneous detection of large (trolley handles, 35% of image) and small (mounting holes, 0.14% of image) objects for autonomous robotic docking. Pure machine learning approaches using YOLOv8 achieved excellent handle detection (99.5%) but failed on tiny holes (60% reliability). Pure classical computer vision generated excessive false positives (40%+) from mesh patterns and background features.

The proposed hybrid solution combines YOLOv8 for handle detection with Canny edge detection for hole localization within the handle region of interest. Integration with ROS interfaces enables visual-servoing-ready alignment outputs. Key subsystem results: 99.5% handle detection, 96% hole detection, absolute alignment error below 2 cm in 89% of successful alignment trials, and approximately 19 FPS real-time performance on CPU. Full physical end-to-end autonomous docking execution was not completed within the thesis timeline.

10.2 Research Contributions

10.2.1 Scientific Contributions

Novel hybrid architecture specifically addressing multi-scale detection where handle and hole targets differ by approximately a 250:1 image-area ratio. Empirical validation demonstrating hybrid approach superiority over pure machine learning or pure classical computer vision. Comprehensive performance analysis including accuracy, speed, robustness, and failure mode characterization in real-world conditions. Practical insights for matching algorithm types to object characteristics based on size, geometry, and context requirements.

10.2.2 Practical Contributions

Working prototype perception, alignment, and hardware subsystems for hospital logistics automation demonstrated on the Done Robotics Drivey robot. ROS-compatible framework enabling future integration with autonomous mobile robots and motion control systems. Dataset creation and annotation methodology for small object detection with limited data. Deployment guidelines and best practices for vision-based docking applications in healthcare environments.

10.3 Answers to Research Questions

RQ1: Can hybrid ML+CV approaches outperform pure approaches for multi-scale detection? Yes. Here, "pure approaches" means pure machine-learning-only (YOLO-only) and pure classical-computer-vision-only (edge-only) pipelines. Hybrid achieved 96% hole detection versus 60% for pure YOLO and 65% for pure edge detection.

RQ2: What are optimal architectures for large versus small object detection? YOLOv8n for large objects (context-aware, robust to appearance variations), Canny edge detection for small geometric features (precise localization, computationally efficient).

RQ3: What accuracy and speed are achievable for real-time robotic docking? For subsystem evaluation, the system achieved 99.5% handle detection, 96% hole detection, absolute alignment error below 2 cm in 89% of successful alignment trials, and approximately 19 FPS processing. Full end-to-end physical docking was deferred to future integration work.

10.4 Limitations

Small training dataset (39 images from one primary trolley unit), no separate held-out test set, limited trolley variants tested (3 types), depth-measurement sensitivity to reflective/occluded surfaces, controlled indoor environment only, static detection (no moving target tracking), no completed full physical end-to-end docking trial in this thesis, and

operational range constraint (0.5–2.0 meters).

10.5 Future Work

10.5.1 Short-Term Improvements

Dataset expansion: collect 200+ images with diverse trolley types, environments, and lighting conditions. Model optimization: fine-tune YOLOv8s/m variants, explore quantization (INT8) for faster inference, implement model pruning to reduce size. Robustness enhancements: add temporal filtering (Kalman filter) for tracking, implement multi-frame fusion, develop confidence-based fallback mechanisms. Deployment: port to edge devices (NVIDIA Jetson Nano, Raspberry Pi 4), optimize for embedded inference (TensorRT), conduct multi-site field testing.

10.5.2 Mid-Term Extensions

Depth estimation: integrate stereo camera or RGB-D sensor (Intel RealSense), implement 6-DOF pose estimation for full 3D alignment. Advanced hole detection: train instance segmentation model (YOLOv8-seg), explore specialized small object detectors (SNIPER, FPN++), combine multiple detection modalities. Autonomous docking: implement full closed-loop control with force feedback, add collision avoidance and obstacle detection, develop recovery behaviors for failure modes. Multi-trolley support: extend to handle multiple trolleys in frame simultaneously, implement trolley ID recognition for fleet management, develop tracking for moving targets.

10.5.3 Long-Term Research Directions

End-to-end learning: train neural network directly from images to control commands, explore reinforcement learning for optimal docking strategy, compare learned versus modular hybrid approach. Generalization: develop universal docking system for any trolley type without retraining, transfer learning across different robot platforms, meta-learning for rapid adaptation to new environments. Advanced perception: integrate semantic scene

understanding with SLAM, implement predictive models for trolley motion, explore attention mechanisms for robust feature localization. Human-robot interaction: develop augmented reality visualization for operators, implement voice and gesture control interfaces, study user trust and acceptance in clinical settings.

Recent industry developments in humanoid robotics indicate a shift toward world-model-based systems that can infer physical actions from large-scale video priors and self-generated interaction data. While this thesis focuses on a task-specific docking pipeline, a practical long-term extension is to combine the current safety-structured hybrid architecture with data-efficient self-improvement loops, where robots refine manipulation policies from controlled real-world experience while preserving explicit safety checks at lock confirmation and motion authorization stages (Investing News Network, 2026).

10.6 Recommendations for Practitioners

Use hybrid approaches when: detecting objects with large scale disparity (for example, approximately 250:1 image-area ratio), working with limited training data (less than 100 images), requiring real-time performance on standard hardware, and needing high reliability (greater than 95% accuracy).

Implementation guidelines: start with prototyping to test classical CV baseline, match algorithms to task characteristics (large/contextual objects to ML, small/geometric to classical), use ROI strategies to constrain search space and reduce false positives, iterate based on failure analysis rather than relying solely on validation metrics, adopt modular architecture for easier debugging and maintenance.

10.7 Final Remarks

This thesis demonstrated that hybrid computer vision systems combining deep learning and classical algorithms can achieve superior performance for robotic tasks involving multi-scale object detection. The developed system established and validated key subsystems required for autonomous trolley docking on the Drivey platform, contributing

practical evidence for hospital logistics automation. The work provides both scientific insights into hybrid architecture design and practical guidelines for deploying vision-based robotic systems in healthcare-oriented environments.

The key takeaway: when designing robotic vision systems, matching algorithm strengths to task characteristics (large contextual objects to deep learning, small geometric features to classical computer vision) yields better results than pursuing pure approaches. This principle extends beyond trolley docking to numerous robotic applications requiring robust, real-time perception in challenging real-world environments.

Bibliography

Bradski, G. (2000). *The OpenCV Library*. Retrieved from https://opencv.org/

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*(6). https://doi.org/10.1109/TPAMI.1986.4767851

Chaumette, F., & Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, *13*(4). https://doi.org/10.1109/MRA.2006.250573

Chen, S., Zhang, H., Yang, Z., Yang, Y., & Wang, K. (2024). Aruco code-based calibration method for lidar and camera integration. In *2024 ieee 25th china conference on system simulation technology and its application (ccssta)*. https://doi.org/10.1109/CCSSTA62096.2024.10691812

Christie, J. (n.d.). *Scrum graphic.* (Retrieved 2026-05-14 from Jim Christie blog) Retrieved from https://jimchristie.me/blog/scrum-graphic/

Claude haiku 4.5 [large language model]. (2026). (Referenced from project usage summary (full terminal tool), accessed 2026-05-07) Retrieved from https://docs.anthropic.com/en/docs/about-claude/models

Fan, G., & Wang, G. (2017). Vision-based autonomous docking and recharging system for mobile robot in warehouse environment. In *2017 2nd international conference on robotics and automation engineering (icrae)*. https://doi.org/10.1109/ICRAE.2017.8291357

Garrido-Jurado, S., Munoz-Salinas, R., Madrid-Cuevas, F. J., & Marin-Jimenez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6). https://doi.org/10.1016/j.patcog.2014.01.005

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Retrieved from https://arxiv.org/abs/1311.2524

Hartley, R., & Zisserman, A. (2004). *Multiple view geometry in computer vi-*

sion (2nd ed.). Cambridge University Press. Retrieved from https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A

Holland, J., Kingston, L., McCarthy, C., Armstrong, E., O'Dwyer, P., Merz, F., & McConnell, M. (2021). Service robots in the healthcare sector. *Robotics*, *10*(1). https://doi.org/10.3390/robotics10010047

Hutchinson, S., Hager, G. D., & Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, *12*(5). https://doi.org/10.1109/70.538972

Intel Corporation. (2025). *Intel realsense d400 series and sdk 2.0 documentation*. (Accessed: 2026-05-06) Retrieved from https://dev.intelrealsense.com/docs

International Electrotechnical Commission. (2010). *Iec 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems*. Retrieved from https://webstore.iec.ch/publication/5515

International Organization for Standardization. (2014). *Iso 13482:2014 robots and robotic devices - safety requirements for personal care robots*. Retrieved from https://www.iso.org/standard/53820.html

Investing News Network. (2026, January 12). *1x unveils paradigm shift in humanoid ai: Neo's starting to learn on its own*. (News release coverage (via GlobeNewswire), accessed 2026-05-07) Retrieved from https://investingnews.com/1x-unveils-paradigm-shift-in-humanoid-ai-neo-s-starting-to-learn-on-its-own/

Kim, T., Kang, G., Lee, D., & Shim, D. H. (2024). Development of an indoor delivery mobile robot for a multi-floor environment. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3381489

Kroeger, O., Huegle, J., & Niebuhr, C. A. (2019). An automatic calibration approach for a multi-camera-robot system. In *2019 24th ieee international conference on emerging technologies and factory automation (etfa)*. https://doi.org/10.1109/ETFA.2019.8869522

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553).

Li, Y., & Shi, C. (2018). Localization and navigation for indoor mobile robot based on ros. In *2018 chinese automation congress (cac)*.

https://doi.org/10.1109/CAC.2018.8623225

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Retrieved from https://arxiv.org/abs/1612.03144

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*. https://doi.org/10.1007/978-3-319-10602-1_48

Low, E. M. P., Manchester, I. R., & Savkin, A. V. (2006). A method for vision-based docking of wheeled mobile robots. In *2006 ieee conference on computer aided control system design, 2006 ieee international conference on control applications, 2006 ieee international symposium on intelligent control*. https://doi.org/10.1109/CACSD-CCA-ISIC.2006.4776716

Mafaldo, J. P., Rodrigues, P. C. G., Da Silva, M. R., & Durand-Petiteville, A. (2025). Vision-based docking for robotic towing of mobile platforms. In *2025 ieee international conference on advanced robotics (icar)*. https://doi.org/10.1109/ICAR65334.2025.11338673

Martinez-Marin, T., & Duckett, T. (2008). Learning visual docking for non-holonomic autonomous vehicles. In *2008 ieee intelligent vehicles symposium*. https://doi.org/10.1109/IVS.2008.4621291

OpenAI. (2026a). *Gpt-5.3 codex (high reasoning) [large language model]*. (Referenced from project usage summary, accessed 2026-05-07) Retrieved from https://platform.openai.com/docs/models

OpenAI. (2026b). *Gpt-5 nano [large language model]*. (Referenced from project usage summary (tool summarization), accessed 2026-05-07) Retrieved from https://platform.openai.com/docs/models

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., … Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, *39*(3). https://doi.org/10.1016/S0734-189X(87)80186-X

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., … Ng, A. Y. (2009). Ros:

an open-source robot operating system. In *Icra workshop on open source software* (Vol. 3). Retrieved from https://ai.stanford.edu/~mquigley/papers/icra2009-ros.pdf

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Retrieved from https://arxiv.org/abs/1506.02640

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (Vol. 28). Retrieved from https://arxiv.org/abs/1506.01497

Schwaber, K., & Sutherland, J. (2017). *The scrum guide*. Scrum.org. Retrieved from https://scrumguides.org/scrum-guide.html

Singh, B., & Davis, L. S. (2018). An analysis of scale invariance in object detection snip. In *Proceedings of the ieee conference on computer vision and pattern recognition*. Retrieved from https://arxiv.org/abs/1711.08189

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media. Retrieved from https://szeliski.org/Book/

Tang, S., Zhang, S., & Fang, Y. (2024). Hic-yolov5: Improved yolov5 for small object detection. In *2024 ieee international conference on robotics and automation (icra)*. https://doi.org/10.1109/ICRA57147.2024.10610273

Thamrongaphichartkul, K., Worrasittichai, N., Prayongrak, T., & Vongbunyong, S. (2020). A framework of iot platform for autonomous mobile robot in hospital logistics applications. In *2020 15th international joint symposium on artificial intelligence and natural language processing (isai-nlp)*. https://doi.org/10.1109/iSAI-NLP51646.2020.9376823

Tzutalin. (2015). *Labelimg: Graphical image annotation tool*. https://github.com/tzutalin/labelImg. GitHub. Retrieved from https://github.com/tzutalin/labelImg

Ultralytics. (2023). *Yolov8: A new state-of-the-art model*. (Accessed: 2026-01-10) Retrieved from https://github.com/ultralytics/ultralytics

Yazici, F. (2023). *Overview of scrum framework*. (Medium article, accessed 2026-05-07) Retrieved from https://fatihyazici7.medium.com/overview-of-scrum-framework-845a84802c78

Yu, Y., Zhang, K., Liu, H., Yang, L., & Zhang, D. (2020). Real-time visual localization of the picking points for a ridge-planting strawberry harvesting robot. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.3003034

Zhang, X., & Xu, S. (2020). Research on image processing technology of computer vision algorithm. In *2020 international conference on computer vision, image and deep learning (cvidl)*. https://doi.org/10.1109/CVIDL51233.2020.00030

Appendices

Appendix 1. YOLOv8 Training Configuration

Key training configuration used for the final YOLOv8n run is summarized below:

```
model: yolov8n.pt
data: data.yaml
imgsz: 640
epochs: 100
batch: 8
name: hole_detection
patience: 20
save: true
plots: true
dataset:
  total_images: 39
  train: 32
  val: 7
```

The configuration above is taken from the actual training files `data/TrainingMLYolov8/train_yolo.py` and `data/TrainingMLYolov8/data.yaml` in the project repository. Optimizer and low-level augmentation coefficients were not explicitly overridden in the script, so that run followed Ultralytics defaults.

Appendix 2. Edge Detection Algorithm Pseudocode

The practical hole-localization sequence used in deployment:

1. Acquire RGB frame and YOLO handle bounding box.
2. Crop handle ROI (optional 5–10% padding).

3. Convert ROI to grayscale.
4. Apply CLAHE (clip limit 3.0, tile grid 8×8).
5. Apply Gaussian blur (kernel 7×7, $\sigma = 1.5$).
6. Run Canny edge detection (low threshold 100, high threshold 200).
7. Apply morphological closing (2×2 structuring element, 1 iteration).
8. Extract external contours and compute contour features.
9. Keep contours satisfying area and circularity thresholds.
10. Select best candidate based on circularity and geometric consistency.
11. Map candidate center back to full-frame coordinates.
12. Apply temporal median + exponential smoothing before control output.

Appendix 3. ROS Node Architecture

Core ROS interfaces used in the integrated pipeline:

Node/Interface	Topic or Message	Purpose
Camera driver	/camera/color/image_raw, /camera/depth/image_raw	Provides synchronized RGB-D input streams.
Perception node	/docking/handle_bbox, /docking/hole_center	Publishes detected handle ROI and hole center coordinates.
Alignment node	/docking/alignment_offset (geometry_msgs)	Publishes image-space and metric alignment corrections.
Control node	/cmd_vel, /docking/actuator_cmd	Executes lateral correction and hook insertion commands.
Feedback node	/docking/lock_state, /docking/ir_state	Reports mechanical lock and engagement confirmation.

Table 6. ROS-level message flow used in the docking prototype.

Appendix 4. Performance Metrics Summary

Consolidated metrics from validation and deployment tests:

Metric	Result
Handle detection (validation mAP@0.5)	99.5%
Hole localization success (ROI, deployment)	96% (48/50)
End-to-end physical docking result	Not executed in thesis timeframe
Absolute alignment error below 2 cm	89.4% of successful alignment trials
Average runtime per frame	52 ms
Average frame rate	19.2 FPS (CPU)

Table 7. Summary of primary quantitative outcomes used in thesis discussion.

Appendix 5. Hardware Specifications

Component	Specification / Role
Compute platform	Intel Core i5-class CPU system for full perception pipeline
Camera	Intel RealSense D435 RGB-D camera
Low-level controller	Raspberry Pi Pico for actuator/sensor I/O
Actuator	24V solenoid plunger for mechanical lock/unlock
Sensors	IR beam pair, limit switch, ToF distance sensor
Mechanical subsystem	Hook module with rail-guided Z-axis insertion mechanism
Robot base	Done Robotics Drivey platform (integration target)

Table 8. Hardware stack used during implementation and evaluation.

Appendix 6. Code Repository Structure

Representative project structure for reproducibility:

```
Done Thesis Project/
  data/
    TrainingMLYolov8/
      train_yolo.py
      alignment_system_hybrid.py
      data.yaml
    dataset/
      train/images, train/labels
      val/images, val/labels
  Intel_RealSense/
    alignment_calibration_test.py
    alignment_system_realsense.py
    robot_movement_commander.py
  Hole Detection/
    alignment_system.py
    hole_detector_working.py
  Control System/
    src/main.c
    src/main_oled.c
    src/main_io_test.c
  ros/
    simple_publisher.py
    simple_subscriber.py
  data/
    TrainingMLYolov8/runs/detect/hole_detection/weights/best.pt
```

Repository-level implementation details that may contain confidential mechanical integration specifics remain internal pending confidentiality and IP review.