Vision Perception Optimization and Adaptive Control for Resource-Constrained Platform: A Ping-Pong Ball Pickup & Place System

Duo Peng

A project report submitted to the Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

School of Engineering, Computer & Mathematical Sciences

2025

Abstract

This thesis presents the design, implementation, and experimental evaluation of an autonomous ping-pong ball collection robot system. The system integrates computer vision, object detection, visual servoing, and robotic manipulation to create a fully autonomous solution capable of identifying, collecting, and organizing ping-pong balls in dynamic environments. Using a mecanum-wheeled mobile platform equipped with a 6-DOF robotic arm and multiple cameras, a comprehensive system architecture was developed based on ROS (Robot Operating System) with a state machine design to orchestrate complex task sequences. The system demonstrates the ability to successfully detect, approach, and collect ping-pong balls while navigating in varied environments.

System performance was evaluated through comprehensive experiments, including YOLO v12 Nano model benchmarking, ablation studies of optimization techniques, spiral search strategy validation, and visual servoing performance analysis. The YOLO model evaluation demonstrates that a properly optimized Nano variant achieves 98.9% mAP@0.5 while maintaining sufficient inference speed on resource-constrained hardware. Ablation studies reveal that combining TensorRT with FP16 precision yields a 592.6% performance improvement with negligible accuracy loss. The spiral search strategy demonstrates effective target recovery capabilities when objects temporarily leave the field of view. Most notably, visual servoing experiments demonstrate that lower frame rates (15 *fps*) offer substantial advantages by enabling multi-camera operations and providing greater system functionality compared to higher frame rates (30 *fps*), despite conventional expectations.

This thesis presents a fully integrated robotic system tailored specifically for resource-constrained embedded platforms. The research introduces a refined YOLO-based vision pipeline, an adaptive spiral search method validated under diverse conditions, and a visual servo control strategy achieving millimeter-level accuracy. These contributions offer practical guidance for effectively deploying advanced robotic systems where computational resources are limited.

Keywords: Autonomous robotics, Visual servoing, YOLO object detection, Robot operating system (ROS), Ping-pong ball collection, Model optimization, Spiral search strategy, Resource-constrained computing, Multi-camera systems, Real-time performance

Table of Contents

List of Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DOF	Degrees of Freedom
DWA	Dynamic Window Approach
FP16	Floating-Point 16-bit
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mAP	Mean Average Precision
PID	Proportional-Integral-Derivative
RAM	Random Access Memory
ROS	Robot Operating System
RTAB-Map	Real-Time Appearance-Based Mapping
SLAM	Simultaneous Localization and Mapping
SSD	Single Shot MultiBox Detector
TEB	Timed Elastic Band
TensorRT	Tensor Runtime
USB	Universal Serial Bus
YOLO	You Only Look Once

List of Figures

List of Tables

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature: _____ Date: 15 May 2025

Acknowledgment

I would like to express my profound gratitude to my supervisor, Dr. Wei Qi Yan, whose unwavering support, expertise, and insightful guidance have been instrumental in shaping my research. His mentorship enriched my academic journey and was vital to the successful completion of this study. I also extend my sincere thanks to the administrators at Auckland University of Technology for their invaluable assistance throughout this process.

Duo Peng

Auckland, New Zealand

May 2025

Chapter 1
Introduction

This chapter is divided into five sections: The first part presents background and motivation; the second one outlines the problem statement and research motivation; the third section details the research objectives; the fourth highlights key contributions; and the fifth explains the overall structure of this report.

1.1 Background and Motivation

Autonomous robots that can perceive their environment and act accordingly are becoming integral to modern engineering and society. In the context of mobile robotics, advances in perception, decision-making, path planning, and control enable robots to navigate and perform tasks independently, making them core components of future intelligent systems (Tang et al., 2025). Increasingly, artificial intelligence (AI) techniques – especially deep learning – are transforming robotics by making robots more intelligent, efficient, and adaptable to complex tasks and environments (Soori et al., 2023). Computer vision plays a pivotal role in this transformation, allowing robots to recognize and interpret their surroundings in real time. For example, visual object detection algorithms have matured to the point that fast, accurate detectors like YOLO (i.e., You Only Look Once) can run at dozens of frames per second, enabling responsive robotic systems capable of human-like reactions (Redmon et al., 2016). Modern visual perception systems leverage deep neural networks for object recognition, detection, segmentation, and tracking, which has largely supplanted classical feature-based methods in many domains (Machkour et al., 2021). This trend has led to more robust and flexible robotic vision capabilities, enabling applications from autonomous driving to service robots.

One particular application area that benefits from these advances is autonomous ball collection for sports training. In sports like table tennis (ping-pong) or tennis, many balls end up scattered after practice sessions, and manually retrieving them is tedious and time-consuming. An autonomous robot that can efficiently locate and pick up balls can save athletes and coaches considerable effort. The significance of this application is evidenced by recent research and development projects focusing on ball-collecting robots. For instance, Nguyen-Dang et al. (2022) developed an indoor ping-pong ball collecting robot addressing key challenges such as ball detection, distance estimation, path planning, and obstacle avoidance. Likewise, an autonomous tennis ball collector robot was designed to roam a court and retrieve balls, demonstrating the practicality of such a device for sports training (Çabuk et al., 2018). In industry as well, commercial products (e.g., tennis ball “robots”) are emerging, underscoring the demand for automated solutions in this niche.

Beyond sports, the underlying technologies – vision perception and adaptive control on resource-constrained platforms – have broad significance. Many resource-constrained robots (small mobile robots, drones, etc.) operate with limited processing power, battery capacity, and weight budget. Achieving robust vision and control on such platforms is a critical research challenge. The ability to deploy advanced computer vision algorithms on low-cost embedded hardware expands the reach of robotics into domestic, agricultural, and environmental applications where expensive hardware is not feasible. Researchers have highlighted that low-cost embedded systems, when optimized, can solve tasks like real-time image recognition, object tracking, and obstacle detection onboard small robots (Beltrán-Escobar et al., 2024). This opens opportunities for ubiquitous robotics – tiny autonomous robots that can assist in daily tasks. However, realizing these capabilities requires addressing significant constraints in computation and energy, which motivates the research in vision optimization and adaptive control for resource-limited systems.

This thesis addresses specific technical challenges of deploying a high-performance vision and control system on a small, resource-constrained robotic platform. By focusing on the practical problem of autonomous ping-pong ball collection, this research develops and validates optimization techniques in deep learning-based object detection, adaptive control strategies for differential-drive systems, and integrated sensor fusion methods. The work not only enhances system efficiency and reliability in constrained environments but also provides a detailed methodology for achieving real-time performance on low-cost hardware.

1.2 Problem Statement and Research Motivation

Manually collecting ping-pong balls after practice is a mundane task that distracts from training focus. This problem motivates the development of an Autonomous Ping-Pong Ball Collection Robot that can perform the task automatically. The core problem statement is: How can we design a vision-guided, autonomous mobile robot capable of finding and picking up ping-pong balls in real time, given the limited computational and power resources of a small robotic platform? This breaks down into several challenges: reliable ball detection in variable

conditions, estimating the ball's location for navigation, planning and executing a path to the ball while avoiding obstacles, and physically collecting the ball – all under tight resource constraints (CPU, memory, battery).

The research motivation extends beyond simply automating ball pickup. This project serves as a case study for developing vision perception and control algorithms on a constrained system. In high-end robotics (e.g., self-driving cars), one can leverage powerful GPUs and sensors; in contrast, our scenario demands efficiency and clever optimizations. The motivation is to push the boundaries of what embedded vision can do. Recent surveys have reviewed deep learning models for image recognition on embedded devices (Rodrigues Moreira et al., 2025), indicating increasing interest in compressing AI models to fit edge hardware. Nonetheless, deploying state-of-the-art detectors like YOLOv8 on small devices is challenging due to limited computing resources (Chen et al., 2025). Addressing this, researchers propose optimized models – e.g., Chen et al. (2025) introduced FRYOLO, a lightweight real-time object detector tailored for IoT devices, to handle the complexity of YOLOv8 on limited hardware. This thesis is motivated by similar concerns: how to achieve accurate, real-time ball detection on a modest processor (such as a Jetson Orin Nano or microcontroller).

Another aspect of the problem is maintaining robust performance under changing conditions. A ping-pong ball collection robot operates in dynamic environments – lighting can vary, obstacles (like table legs or other objects) may be present, and the robot's own movements introduce uncertainty (e.g., bumps, wheel slip). This motivates research into adaptive control strategies that can cope with uncertainties and maintain stability. Adaptive control allows a robot to adjust its control parameters on the fly to handle unknown dynamics or payload changes. Prior work demonstrates that an adaptive controller can significantly reduce the impact of modeling errors for mobile robots. For example, Cui et al. (2023) showed a Lyapunov-based adaptive scheme that guarantees stability for a wheeled robot with uncertain dynamics. Inspired by such approaches, our robot's control system needs to adapt to factors like battery voltage drop or uneven flooring to reliably steer and position the collector mechanism.

A key motivation is also the resource-constrained platform itself. Rather than assuming a

powerful computer on the robot, we deliberately target limited hardware to reflect real deployment scenarios (e.g. an affordable robot for hobbyists or schools). This means every algorithm – from image processing to motion planning – must be optimized for efficiency. As Singh et al. (2024) note, the bottleneck for very small robots (under 100 mm in size) is real-time perception given severe size, weight, and power limits (Singh et al., 2024). While our platform might be slightly larger, the same principle applies we cannot simply use the most computationally intensive approach and expect it to work. The need to balance performance (speed and accuracy) with efficiency is at the heart of the problem. This motivates exploring techniques like model compression, approximate computing, and clever sensor fusion as part of the solution.

1.3 Research Objectives

Given the problem statement, the research objectives of this thesis are defined as follows:

- Objective 1: Optimize Vision Perception for Ping-Pong Ball Detection:

Develop and validate a lightweight, real-time vision system for detecting ping-pong balls on an embedded platform. This objective focuses on refining the detection algorithm (using a custom YOLO-based model) to ensure high true-positive rates and robust performance under varying lighting conditions and backgrounds, as demonstrated by our object detection experiments.

- Objective 2: Develop and Evaluate an Adaptive Spiral Search Strategy:

Design and experimentally validate an adaptive spiral search approach that reliably reacquires targets in dynamic and cluttered environments. This objective involves integrating obstacle avoidance via costmap data and quantifying the impact of different search distances and angular deviations on recovery performance, as shown in our spiral search experiments.

- Objective 3: Achieve High-Precision Visual Servo Control through PID Parameter Optimization:

Implement a visual servo control system with a finely tuned PID controller to ensure precise positioning for object grasping. This objective includes systematically optimizing the PID parameters through static grasping precision tests and analyzing performance metrics

(such as approach time, stability, and positional accuracy) to achieve millimeter-level precision in the collection task.

1.4 Key Contributions

This thesis yields several contributions to the field of robotics, particularly in vision-based autonomous systems on constrained hardware. The key contributions are:

- An Optimized Vision Pipeline for Ball Detection on Edge Devices:

Developed a custom vision pipeline that achieves high accuracy in detecting ping-pong balls while running in real-time on a single-board computer. The pipeline combines classical image processing with a lightweight deep learning model, leveraging the strengths of both. A significant contribution is the demonstration that this hybrid approach meets real-time requirements without relying on expensive GPUs. In tests, our YOLO v12 Nano model achieved 98.9% mAP@0.5 with excellent detection accuracy. Through our optimization experiments, we demonstrated that combining GPU acceleration, TensorRT, and FP16 precision yielded a 592.6% performance improvement, achieving peak frame rates of approximately 73 FPS with negligible accuracy loss. This contribution can be generalized to other object detection tasks on resource-limited platforms.

- Adaptive Navigation and Control Strategy:

This thesis introduces an adaptive control mechanism for our mobile robot that significantly improves navigation robustness during ball collection tasks. Unlike fixed-gain controllers, our approach implements a state machine architecture with 57 distinct states to orchestrate complex task sequences, adaptively adjusting visual servoing parameters based on observed tracking errors. Through our spiral search strategy experiments, we demonstrated the system can effectively recover targets when they temporarily leave the field of view, with different search patterns expanding outward from the robot's position. This contribution showcases how a well-designed state machine combined with adaptive visual servoing can enhance performance in real-world robotic applications without requiring excessive computational resources.

- Integrated System Design:

We present the complete design and implementation of an autonomous ping-pong ball collecting robot, integrating perception, navigation, and manipulation within a ROS framework. The system uses a mecanum-wheeled mobile platform for omnidirectional movement, combined with a 6-DOF robotic arm and specialized end effector optimized for 40mm diameter ping-pong balls. A key contribution is our approach to multi-camera operation, enabling the system to simultaneously perform environment mapping, navigation, and precise visual servoing during ball approach and manipulation. The ROS-based modular software architecture separates perception, decision-making, and control into distinct nodes, creating a reusable framework that can be adapted for similar object collection tasks.

- Empirical Evaluation of Frame Rate Impact on System Performance:

This thesis contributes valuable empirical data on the critical relationship between camera frame rates and overall system functionality. Our frame rate impact analysis experiments demonstrated that lower frame rates (15 FPS) offer substantial advantages for resource-constrained systems by enabling multi-camera operations compared to higher frame rates (30 FPS). Specifically, we found that 15 FPS configuration uses only 31% of available USB bandwidth, allowing for simultaneous operation of up to three cameras, while 30 FPS uses 62% and supports only a single camera. These finding challenges conventional assumptions that higher frame rates are always better, revealing that strategic frame rate selection based on system-wide resource considerations can unlock expanded functionality, including stereo vision and multi-angle object detection capabilities that are impossible at higher frame rates.

1.5 Structure of This Thesis

This thesis is organized into seven chapters. Chapter 1 (Introduction) presents the research problem, objectives, and key contributions that set the stage for the work. Chapter 2 (Literature Review) surveys existing research on visual servoing, deep learning-based object detection, adaptive control, and mobile robot navigation, highlighting theoretical foundations and practical gaps. Chapter 3 (System Implementation) details the design and integration of the autonomous ping-pong ball collection robot, covering hardware configuration, sensor integration, and the ROS-based software architecture. Chapter 4 (Methodology) outlines the

experimental setups and optimization strategies used to evaluate system performance, briefly describing the testing procedures for the custom YOLO v12 Nano-based vision system, the spiral search recovery experiments at various distances and angles, and the visual servo control evaluations (including PID tuning and grasping precision tests), which collectively measure detection accuracy, search time, positioning precision, and control stability under realistic conditions. Chapter 5 (Experimental Results) provides a comprehensive quantitative evaluation of the system through metrics such as tracking error, detection success, and power consumption. Chapter 6 (Analysis and Discussions) synthesizes the experimental findings by discussing performance trade-offs, system limitations, and their practical implications. Finally, Chapter 7 (Conclusion and Future Work) summarizes the key contributions of the thesis and outlines potential directions for further research.

Chapter 2 Literature Review

In this chapter, we review the key techniques for our ping-pong ball pickup & place robot, covering object detection evolution, dynamic visual servo control, and deep learning optimization for constrained platforms, while identifying key research gaps.

2.1 Visual Object Detection Techniques (YOLO Evolution and Lightweight Models)

Perception begins with the ability to detect and recognize objects in the robot's view. For a ball-collecting robot, visual object detection is critical to find ping-pong balls against various backgrounds. Early object detection approaches (e.g., the R-CNN family) used a two-stage pipeline: generate region proposals then classify each (Girshick et al., 2014). Two-stage detectors like Faster R-CNN achieve high accuracy but at the cost of speed and computational load, often preventing real-time operation (Ren et al., 2016). In contrast, one-stage detectors such as You Only Look Once (YOLO) frame detection as a single regression problem solved in one pass of a neural network (Redmon et al., 2016). This section reviews the evolution of YOLO and related models, as they form the foundation for real-time detection on robots. We also examine lightweight CNN architectures and modifications tailored for embedded vision, which trade some accuracy for efficiency – a key consideration for resource-constrained robots.

One-stage detectors like YOLO and Single Shot MultiBox Detector (SSD) emerged mid-2010s as alternatives to two-stage methods. YOLOv1 was a breakthrough: it could detect objects in images at 45+ frames per second by using a single convolutional network, albeit with lower accuracy than slower two-stage methods (Redmon et al., 2016). YOLOv2 and YOLOv3 improved accuracy with tricks like multi-scale anchors and deeper networks while retaining high speed. In parallel, SSD introduced multi-scale feature maps for detection, achieving e.g. ~74% mAP on VOC at 59 FPS (Liu et al., 2016). The core idea of these one-stage models is to directly predict bounding boxes and class probabilities from image features, eliminating explicit proposal generation. This makes them much faster but historically posed challenges in detecting small objects and maintaining accuracy on par with two-stage detectors (Lin et al., 2017; Tian et al., 2019).

In the last five years, the YOLO family has advanced rapidly. YOLOv4 combined numerous "bag-of-tricks" (e.g. CSP connectivity, Mish activation, mosaic data augmentation) to substantially improve accuracy while keeping real-time speed (Bochkovskiy et al., 2020). It achieved 43.5% AP (COCO metric) at ~65 FPS on the Tesla V100 GPU, surpassing

contemporaries in the speed-accuracy tradeoff (Bochkovskiy et al., 2020). This represented state-of-the-art performance for real-time detectors of its time. More recently, YOLOv7 pushed the frontier further – introducing novel training optimizations and model scaling strategies. YOLOv7 reports 56.8% AP (COCO) while still running at >30 FPS on GPU, outperforming both previous YOLO versions and other real-time detectors in combined speed and accuracy (Baghbanbashi et al., 2023; Wang et al., 2023).

While top-performing YOLO models run in real-time on powerful GPUs, deploying them on a small robot (likely with only an embedded GPU or CPU) often requires model compression and simplification. Thus, a significant sub-field has focused on tiny or efficient detectors. For example, YOLO-Tiny versions use fewer convolutional layers and filters to drastically cut model size at some cost to accuracy. Fang et al. (2020) proposed Tinier-YOLO, a variant of YOLOv3-tiny with SqueezeNet fire modules, achieving comparable accuracy to YOLOv3-tiny with 87% fewer parameters. Similarly, Feng et al. (2021) developed an Embedded YOLO model (for smart cars) with only 3.53 million parameters, running at 155 FPS on their test platform. These approaches demonstrate that it's feasible to shrink deep detectors for edge deployment. Other compact architectures like MobileNet and ShuffleNet have been incorporated as backbones for detection on low power devices (Howard et al., 2017; X. Zhang et al., 2017). Beyond architecture design, techniques like model pruning and quantization (discussed in the next section) can further lighten object detectors. A key challenge, however, is maintaining reliability on small objects like ping-pong balls. Small objects occupy few pixels, making them harder to detect especially for downsized models. Recent works address this by multi-scale feature fusion and data augmentation. For instance, EfficientDet (Tan et al., 2020) uses a weighted bi-directional feature pyramid to improve small object detection, though its speed on CPU is lower than YOLOv4 (Bochkovskiy et al., 2020). In summary, modern detectors provide a spectrum from large, highly accurate models to lean models optimized for speed. When choosing a vision solution for a resource-constrained robot, one must balance accuracy needs (e.g. avoiding missed detections of balls) with inference speed. Research gaps remain in achieving robust detection of small, fast-moving objects under computation limits – an area this thesis will explore by leveraging and fine-tuning these

lightweight detection models.

Applying advanced detectors on constrained hardware raises several issues. Memory footprint and power consumption can be prohibitive for running full YOLO models on microprocessors. Even “tiny” models may struggle with accuracy in visually complex scenes (e.g. multiple balls among clutter). Lighting variations and motion blur (common in dynamic games areas) add to detection difficulty. While YOLOv4/v7 achieve impressive benchmarks on datasets, their performance on an embedded robot in real-world conditions may drop. A notable gap is strategies for online adaptation – e.g., can the detector adjust its parameters on the fly if frame rate drops or if false negatives occur frequently? Another gap is integration with tracking: continuous ball tracking (with something like SORT or optical flow) can help maintain target lock, but lightweight tracking that pairs well with a lightweight detector needs investigation. These issues suggest the need for a holistic approach combining detection with other perception techniques, ensuring that no balls are missed due to computational skips. We will later see how multi-camera setups and custom search behaviors can compensate for some detection limitations. Overall, visual detection technology provides a solid foundation, but further refinement and adaptation are needed for optimal performance on a small ball-collecting robot platform.

2.2 Deep Learning Optimization for Edge Devices

Quantization involves reducing the numerical precision of network parameters and operations. For example, weights originally in 32-bit float can be quantized to 8-bit integers. This yields up to 4× reduction in model size and speeds up inference on CPUs/DSPs that support INT8 arithmetic (Kum et al., 2022). Crucially, with proper calibration or quantization-aware training, the accuracy loss can be minimal (often within 1-2% of the original model (Wu et al., 2020)). NVIDIA’s TensorRT library, for instance, provides explicit support for FP16 and INT8 quantization – it can take a trained model and optimize it to use low-precision kernels on GPU tensor cores (Kum et al., 2022), enabling significant speed-ups on edge devices. Pruning is another strategy, which removes less important weights or filters from the network. Early works by Han et al. showed that 80-90% of weights in deep CNNs could be pruned with only minor

accuracy impact, resulting in “thin” networks that run faster (S. Han et al., 2015). Pruning can be unstructured (dropping individual connections) or structured (dropping entire channels or layers) – the latter is more hardware-friendly since it reduces memory and compute in a regular pattern. Modern detectors have used pruning; for example, there are pruned versions of YOLO that maintain high mAP with significantly fewer operations (Fang et al., 2020). An emerging idea is neural architecture search (NAS) to automatically design efficient models under resource constraints – e.g., MCUNet (J. Lin et al., 2023) finds tiny CNN architectures that fit within a few hundred kilobytes, suitable for microcontroller-class devices. While NAS is training-time heavy, it underscores the push towards TinyML solutions that can run inference in extreme edge scenarios.

In addition to model modifications, leveraging optimized inference engines can greatly benefit deployment. TensorRT (by NVIDIA) is a widely used SDK that takes a trained network (from TensorFlow, PyTorch, etc.) and performs low-level optimizations: layer fusion, kernel autotuning, memory layout optimizations, and mapping to INT8/FP16 as appropriate (Verma et al., 2021). By using TensorRT on a Jetson GPU, developers have reported 2–4× throughput improvements versus naive frameworks (Jeong, Kim, Tan, et al., 2022). For example, Kum et al. (2022) describe a TensorRT-based pipeline on NVIDIA Jetson that maximizes parallelism and throughput for multiple camera feeds. Another popular tool is TensorFlow Lite (TFLite), which is geared toward mobile CPUs and microcontrollers. Verma et al. (2021) compared TensorRT and TFLite, finding that “TF-TRT (TensorFlow-TensorRT integrated) consistently performs better at high precision on GPU, whereas TFLite excels for very low precision on lightweight models”. This suggests that on a GPU-equipped robot (like one with a Jetson Nano), using TensorRT with FP16/INT8 will yield best performance, while on a pure CPU device, a TFLite 8-bit quantized model might be more efficient. Another category of tools includes OpenVINO (for Intel CPUs and VPUs) and TVM (an open deep learning compiler). All these frameworks share a common goal: transforming standard trained models into optimized executables for target hardware. They implement graph optimizations (pruning redundant operations, reordering for cache efficiency) and can exploit hardware accelerators (e.g. DSPs, NPUs) if available (Verma et al., 2021). For instance, OpenVINO can deploy a network across

CPU and an Intel Neural Compute Stick (Myriad VPU) in a pipeline, balancing load.

Numerous studies have demonstrated effective deployment of deep learning on robots and edge AI systems using the above techniques. One example is in autonomous drones: researchers quantized a CNN for object detection to INT8 and deployed it on a 1-watt Myriad2 VPU, achieving real-time performance onboard a small drone (Queralta et al., 2020). In mobile robotics, SLAM systems now sometimes incorporate edge-optimized DNNs (e.g., for loop closure detection) – Xu et al. (2020) present a mapping approach using a compact CNN on a Raspberry Pi with only a slight accuracy trade-off compared to a laptop baseline. Specifically for ball detection, a relevant case is Tennibot (a tennis ball collecting robot) which uses a CNN classifier on a Jetson Nano (Yang et al., 2023), to meet real-time requirements, they applied TensorRT and reduced precision, enabling inference at 20 FPS with under 10W power (according to product reports). Although detailed publications are sparse (being a commercial product), it exemplifies translating lab CNN models to efficient robotic implementations. Another concrete study by Jeong et al. (2022) shows a Jetson-based inference framework called JEDI, which parallelizes image capture, preprocessing, and model inference across CPU and GPU to maximize utilization. They manage multiple video streams for an ADAS (advanced driver-assistance) system in real-time through careful scheduling and TensorRT optimizations. These engineering efforts underscore how combining compression (at training time) and compiler optimizations (at deployment time) can yield responsive, low-latency perception systems suitable for our target robot.

Despite progress, challenges remain in optimizing deep learning for extremely constrained scenarios. On microcontrollers (with ~100 MHz CPU, few hundred KB RAM), even quantized models struggle – leading to research in binary neural networks and ultra-small models (Novac et al., 2021). Courbariaux et al. (2016) showed that using just 1-bit weights (binary +-1) can drastically reduce model size, though with some accuracy loss. Such binary networks might be needed if one envisions a swarm of tiny robots each with minimal compute that collectively handle tasks. For our ping-pong bot (likely using a moderate embedded CPU/GPU), the more practical gap is ensuring maintained accuracy after optimization. Quantization can sometimes disproportionately affect small object detection or low-contrast images – for example, a study

on quantizing YOLOv7 noted that naive 8-bit quantization caused a 2–6% mAP drop in certain cases (K. Li et al., 2023). Techniques like quantization-aware training or mixed precision (keeping sensitive layers in higher precision) are active research areas to mitigate this (Pandey et al., 2023). Another gap is dynamic resource allocation: real-world robot workloads fluctuate (e.g., during quiet moment fewer detections are needed). Current inference engines are mostly static, but an emerging concept is to dynamically adjust model fidelity or frame rate based on available computation headroom – essentially a runtime quality-of-service for perception. Some preliminary frameworks attempt to partition neural networks between edge and cloud (offloading part of the computation when the robot's CPU is overloaded) (Taufique et al., 2024). For instance, an Edge AI method might run initial layers of a CNN on the robot and send intermediate features to a base station for completion (Kordopatis-Zilos et al., 2017). However, network latency and reliability then become concerns. These are open research questions. In summary, while we have powerful tools to shrink and speed-up deep networks for edge use, ensuring robust, real-time inference under all operating conditions is not fully solved. In this thesis, we aim to apply a combination of quantization and model distillation to the ball detection network, and carefully evaluate its performance on the actual robot hardware, adjusting find the best balance of speed and accuracy.

2.3 Visual Servo Control Systems

Visual servo control has been studied for decades in robotics (Cong & Hanh, 2023). In general, the robot end-effector motion is controlled based on visual error signals – e.g. the difference between the current and desired image coordinates of the target. In IBVS, the control law is defined directly in terms of image feature errors (for example, if the ball appears 20 pixels left of the image center, command a motor rotation to reduce that error)(Peng et al., 2020). IBVS tends to be robust to calibration errors because it does not explicitly estimate the 3D target position; its "steers" based on what the camera sees. In PBVS, the vision system first estimates the 3D pose of the target relative to the robot/camera (using known geometry, depth estimation, etc.), then a classical Cartesian control law (like a PD controller in XYZ) drives the robot to that pose (Zhou et al., 2024). PBVS can be more direct and often converges faster when the

calibration is accurate, but it is sensitive to calibration and modeling errors (Gans et al., 2012). Researchers have noted that IBVS usually offers better robustness to camera calibration inaccuracies and noise (T. Han et al., 2024). whereas PBVS can be faster and sometimes more accurate in ideal conditions (Gans et al., 2012). Both approaches have their merits, and hybrid or switched schemes have been proposed (e.g. using PBVS when far from the target and IBVS when getting close). For a mobile robot collecting objects, IBVS might mean directly using the ball's image position and size to control wheel speeds, whereas PBVS would involve computing the ball's 3D location (perhaps via stereo vision or known size for distance) and navigating to that coordinate. A critical aspect of any visual servo system is the system's dynamic response. Classic analyses consider the image Jacobian (interaction matrix) that relates image motion to robot motion. The control gain must be tuned so that the closed-loop system is stable despite the inherent time delay of image processing (Chaumette & Hutchinson, 2006, 2007). If the camera frame rate is low (introducing larger latency between measuring an error and applying a correction), the control gains must be chosen conservatively to avoid oscillations or instability (Colombo et al., 2019). For example, a visual feedback delay of 100ms can cause a fast robot to overshoot a small target if not accounted for.

Studies have explicitly examined how frame rate and processing delays affect visual servo performance (Lu et al., 2021). By treating the delayed feedback as a passive system, they maintained stability even with variable frame rates around 30 Hz. Similarly, research by Ye et al. (2011) explored the trade-offs between frame rate, delay, and tracking performance in a visual servo system. They found that beyond a certain frame rate (>100 Hz in their case), other bottlenecks like camera exposure time and communication start to dominate. Their work, which involved a custom high-speed vision FPGA system, identified that extremely high frame rates (on the order of 1000 fps) can improve tracking precision, but only if the entire pipeline (sensing to actuation) is designed for such low latency. For typical robotic systems in the 30–60 fps range, predictive filtering is often used – e.g., a Kalman filter can predict the target motion during the ~33ms between frames, making the servo control smoother. An example in literature is the visual tracking of a fast ping-pong ball for a robot arm: the vision at 60fps is augmented by a Kalman predictor to effectively achieve a control update near 200 Hz, allowing

the arm to meet the ball in motion (Zhang et al., 2014). This concept could be useful if our robot ever tries to catch a moving ball or needs to account for a rolling ball's motion.

For our slower scenario (ball usually stationary on the floor), frame rate issues manifest in approach precision. Colombo et al. (2019) proposed a dual-loop visual servo scheme to handle limited camera rates: an outer loop runs PBVS at the camera's slow frame rate, and an inner loop runs a fast joint controller to smooth motion. They demonstrated that with a ~30 Hz camera, their high-authority/low-authority control structure achieved 4.9% lower position error than a conventional single-loop approach. Essentially, the inner high-speed loop (using joint encoders) handles quick corrections while the camera loop provides coarse guidance. This is a practical way to compensate for visual lag. Another relevant study showed that reducing image resolution can increase frame rate and sometimes improve overall task time – Castelli et al. (2017) found that using a 500×400 image at 55 FPS yielded better tracking of a moving object than 1000×800 at 30 FPS. They selected that as a "good trade-off between frame rate and feature quality". This illustrates that in visual servo, faster updates often trump higher image detail, up to a point.

Visual servoing is extensively used in robotic manipulation and mobile robot docking. Recent applications include drone landing (visually servoing to a marker) and self-driving car docking to charging stations. For instance, a 2021 study applied IBVS for a drone to land on a moving platform, using a gimballed camera at 90 Hz – the high update rate was crucial to account for the drone's fast dynamics (Ladosz et al., 2024). In the realm of mobile robots, IBVS has been used for corridor following and object approach. One work (Bouzoualegh et al., 2019) implemented a modified IBVS on a differential-drive robot to approach an object while maintaining its orientation with respect to the camera; they addressed frame lag by slowing the robot's approach as it got nearer (dynamically reducing gain) to avoid overshooting the target due to delayed visual error feedback. This heuristic effectively compensates for lower frame rates by being more cautious when precision is needed.

while distinct from ball collection, ping-pong playing robots use vision at high frame rates to track the ball in flight. Researchers have built systems with 200 Hz stereo vision to track a ping-pong ball and direct a robot arm to hit it (Li et al., 2012). They highlight that latency in

visual processing (even 10–20ms) can significantly degrade performance in fast tasks, so they resort to very fast cameras and predictive models. Our task is slower (ball on ground), but if the robot were to intercept a rolling ball, these insights might become relevant.

A challenge in visual servo control for resource-constrained robots is that often control algorithms assume continuous or high-frequency feedback, which we might not have. If our onboard processing limits the loop to, say, 10 Hz, special measures are needed. One gap is designing controllers that explicitly account for sample-and-hold nature of vision updates. Standard control theory for digital control could be applied, but in practice many visual servo implementations still treat it analogously to continuous with small delay. There is room for more formal treatment of variable frame rate scenarios – e.g., if the vision drops frames when the detector is busy, the controller should adapt. Some recent works speak of event-based vision (using event cameras at kHz rates) to solve latency issues (Gallego et al., 2022). An event camera outputs microsecond-level events of pixel intensity changes, which can enable extremely fast visual feedback. For example, an event-camera-based eye-in-hand system achieved 1 kHz visual servo control for tracking a moving line (Mueggler et al., 2014). This is cutting-edge and not yet common in ball collecting robots, but it points to future directions: using alternative sensors to boost feedback speed. Another gap is integration of learning in visual servo: can a robot learn to predict how its motion will affect the image (the Jacobian) and thus improve servoing over time? Some papers explore learning the image Jacobian with neural networks to handle complex dynamics or unknown robot models. For a simple differential-drive robot, the kinematics are known, but for a more complex arm or if wheel slip occurs, adaptive or learning control could help maintain accuracy. Finally, handling multi-object scenarios is tricky – if more than one ball is in view, how does the control strategy pick which to servo to and ensure it doesn't get confused if targets swap? Classic visual servo assumes a single target whose features are tracked throughout. Switching targets or losing sight (which leads to the next section on reacquisition) is still an open problem. In summary, visual servo control gives our robot the ability to physically converge to a ball's location. Ensuring this works reliably given limited frame rates and potential vision dropouts is a key research concern. In the thesis, we will adopt IBVS for its robustness, and incorporate predictive

filtering to cope with ~15 Hz vision updates, as well as an inner motion controller for smooth control, leveraging insights from the literature (Colombo et al., 2019).

2.4 Spiral Search and Target Reacquisition

In realistic scenarios, the robot will not always detect a ball immediately – it may start with no balls in view (they could be scattered out of camera range or hidden behind obstacles). Thus, search strategies are needed to actively find targets. The spiral is a classic deterministic search pattern. Burlington and Dudek (1999) formally analyzed spiral search for a mobile robot in an unknown environment, proving that it has a competitive worst-case performance (within a logarithmic factor of optimal in an unbounded plane). This approach ensures the robot systematically scans outward in all directions, increasing the likelihood of re-detecting a target that might have moved or been occluded. In ground robots with vision-based tracking, for example, if a robot loses sight of a person or object it was following, it can initiate an outward spiral search around the last seen location to cover the surrounding area efficiently. Tokunaga et al. (2021) note that such spiral reacquisition maneuvers are effective on small mobile robots with limited sensors, enabling them to touch (or sense) the maximum area around the loss point. Empirically, this method has been used in surveillance and search-and-rescue scenarios where precise target location is uncertain. For instance, Piacentini et al. (2019) employ a spiral search around the last known position (LKP) of a target as the first stage in a multi-UAV search strategy. One UAV circles outward from the LKP in a spiral to quickly attempt reacquisition, before broader area coverage is undertaken. This two-stage strategy (local spiral then global sweep) improved success rates in their experiments. Generally, spiral motions leverage the fact that many physical targets (a missing person, a radio beacon, etc.) are likely to be found not far from where they were last detected. By covering ever-larger concentric circles, the robot guarantees it will eventually search all radii around that point. Studies of bio-inspired search behaviors also conclude that spiral-like patterns are an intuitive solution for re-finding targets – for example, nature-inspired search strategies for UAVs often include a spiral phase for reacquisition (Arnold et al., 2018; Burlington & Dudek, 1999).

Spiral searches can be executed in different shapes – the two common forms are circular

(continuous curvature) spirals and rectangular (grid-like) spirals. A rectangular spiral (sometimes called an expanding square pattern) consists of straight legs with 90° turns, gradually increasing in length, effectively tracing a spiral in Manhattan style. Tokunaga et al. note that their small cleaning robot's motion combines straight runs and in-place rotations to approximate a spiral covering path (Yamauchi & Yamashita, 2013). Traditional coverage planners often use such rectilinear patterns (e.g. boustrophedon or lawnmower coverage) for simplicity.

However, circular spirals offer distinct advantages for robots capable of smooth omnidirectional motion. In an omnidirectional or holonomic platform, a circular spiral allows continuous turning without frequent stops or sharp pivots, reducing mechanical stress and time lost to deceleration (Ravankar et al., 2018). By contrast, a rectangular spiral requires the robot to stop or slow at each corner to make a tight 90° turn, which is kinematically less efficient and can cause vibration or slippage.

Empirical comparisons have shown that smooth curved paths enable higher average speeds – a study (Azevedo et al., 2021) observes that a continuous Archimedean spiral yields minimal deviation from a straight-line path when avoiding an obstacle, thereby maintaining speed and stability. Moreover, robots with omni-wheels or holonomic drive can follow circular arcs easily, making circular spirals naturally suited to their mobility.

Tokunaga et al. (2021) highlight that smooth spiral motion was critical for their small robot to "cover most of the surface" efficiently without encoder feedback. In terms of coverage quality, both circular and rectangular spirals ensure the area is systematically scanned, but the circular pattern provides more uniform sensor coverage in all directions (Draelos, 2023). A rectangular pattern inherently biases along the axes and might miss diagonal gaps if not overlapped. For example, NASA field tests with the K10 planetary rover used a rectangular spiral to map terrain around a target feature, achieving high sample density but also requiring careful alignment of the grid to avoid gaps (Fong et al., 2008). Circular spirals inherently have no directional bias – every rotation covers a ring around the start point. In smooth indoor navigation, circular spirals are often preferred for their gentle continuous curvature, whereas rectangular spirals might be chosen for grid-aligned environments or robots that can only

execute straight-line paths easily. Notably, rectangular spirals (and similar grid searches) are a subset of the more general boustrophedon coverage strategies (Burlington & Dudek, 1999), which have their own merits (e.g., easy decomposition of known spaces into (Choset, 2000)). Yet, for omnidirectional navigation systems that can exploit 360° movement, circular spirals offer a clear benefit in terms of motion fluidity and time efficiency (Ravankar et al., 2018). This is evidenced by smoother trajectory tracking and less frequent start-stop behavior, ultimately enabling faster completion of the search pattern.

Implementing spiral searches on real robots requires integration with cost-aware path planning to handle obstacles and terrain. In practice, robots maintain an occupancy grid map of the environment and a dynamic costmap that marks obstacles with high cost to avoid collisions (Elfes, 1989). Naive spiral trajectory must be adapted when obstacles are present – the robot should detour around obstacles and then resume the spiral pattern. One straightforward approach is to expand the spiral only into free space as indicated by the costmap. For instance, as the robot executes the spiral, it continuously checks its local costmap (updated by lidar or vision sensors) to ensure the next arc or outward step is collision-free. If an obstacle lies ahead, the robot can locally circumvent it (e.g. pause outward expansion and go around the obstacle's perimeter) and then return to the spiral path at the earliest safe point (Tokunaga et al., 2021).

Tokunaga et al. demonstrated this behavior: their small robot, upon detecting an obstacle on the table, performed an avoidance maneuver and then autonomously returned to the original spiral trajectory once past the obstacle. Their small robot, upon detecting an obstacle on the table, performed an avoidance maneuver and then autonomously returned to the original spiral trajectory once past the obstacle. This ability to "remember" the intended spiral path and rejoin it is crucial for maintaining coverage without leaving unsearched gaps around obstacles. Dynamic costmaps in ROS can facilitate such behavior by inflating obstacles (adding a safety buffer) and guiding the local planner to generate a path that skirts around the inflation zone and then heads back to the next spiral waypoint.

Essentially, the costmap serves as a real-time constraint on the spiral: the radius can only expand into cells that are free (low cost), and if a region is occupied, the spiral path for that

sector is skipped or postponed until the area is clear. Researchers have also proposed explicit algorithms to find spiral escape points around obstacles (Azevedo et al., 2021). When an obstacle is detected, the algorithm searches along an Archimedean spiral emanating from the obstacle's center to pick a new waypoint for the robot that is minimally deviated from the original path. The first reachable point on this spiral (one that lies outside the obstacle's safety radius) is chosen as a detour point, ensuring the robot avoids collision with minimal path change (Azevedo et al., 2021).

This technique effectively integrates obstacle avoidance into the spiral pattern: rather than abandoning the spiral, the robot finds the nearest gap in the obstacle through a spiral scanning of the obstacle's vicinity. Once the robot navigates to that point, it can continue with the outward spiral search. Use of occupancy grids also allows feasible expansion radius planning – the robot can decide to stop expanding the spiral at a certain radius if the map indicates boundaries or obstacles beyond that point (e.g., walls of a room). In other words, the spiral can be dynamically truncated according to known free space, thereby saving time by not planning into regions that are known to be unreachable. Cost-aware spiral planning thus combines the thoroughness of the spiral with the safety of reactive obstacle avoidance. As Song et al. (2020) describe in their two-stage search, the initial spiral coverage algorithm operates on a grid map of the environment, and obstacles are treated as impassable cells during the search planning. Their rule-based algorithm uses an "internal spiral coverage" method to fully cover the accessible area while a separate module handles frontier exploration for unexplored areas. This ensures that even in the presence of obstacles or map unknowns, the search pattern systematically covers all free regions. Overall, integrating spiral search with modern costmap-based navigation in ROS yields a robust strategy: the robot drives in an expanding spiral but dynamically steers around obstacles and updates its path in real-time, guaranteeing both coverage and collision avoidance.

While a spiral will eventually cover the area given unlimited time, in practice it is important to balance coverage thoroughness with search speed. Multi-scale search strategies build on the spiral pattern by adjusting the search resolution or combining search patterns in phases. One common adaptation is to use an incremental spiral: the robot first executes a coarse

spiral to a certain radius, and if the target is not found, it can enlarge the search or switch strategy. For example, a ground robot might spiral outwards in tight loops up to 10 meters; if the target remains undetected, it could then increase the loop spacing or transition to a faster "lawnmower" sweep of a larger perimeter.

This kind of adaptive approach was exemplified by Piacentini et al. (2019) in their UAV search, where after a spiral around the LKP failed to find the target, the system switched to a broad lawnmower/grid search to cover the wider area. The rationale is that the spiral efficiently checks the most likely nearby locations first (high probability region), and if that fails, the robot invests effort in covering the entire search area. Incremental expansion can also be guided by probability maps – the robot might terminate the spiral once the cumulative area searched corresponds to a certain confidence level that the target is not nearby, then escalate the search. Another aspect of adaptiveness is variable spiral density: The robot can use a finer spiral (smaller loop spacing) when high sensor precision is needed (e.g. camera needs close range), or a looser spiral if it has a long-range sensor, to reduce overlap. Some search algorithms even use a bi-phase strategy: a fast outward spiral to quickly scan for any sign of the target, and if a clue or partial detection is obtained, switch to a finer spiral or local search around that point (Arnold et al., 2018). An analogous behavior is observed in a UAV swarm: when one UAV detects a "critical mass" of survivors, it deviates from the standard pattern to perform a dedicated outward spiral search in that area to locate additional people, before resuming the higher-level search pattern. The targeted spiraling is an adaptive refinement triggered by intermediate results.

In terms of computations, adaptive strategies also prevent wasted effort. Rather than planning one huge spiral that might cover an entire map (which could be computationally heavy and time-consuming if the area is large), the robot plans the spiral in increments – this allows it to periodically reassess if the search should continue or terminate. Search termination criteria are often based on either a maximum radius or time. For instance, an autonomous drone might be instructed to spiral outward until 100 m radius; if the target isn't found by then, the search is concluded or handed off to another system (Azevedo et al., 2021). This prevents the robot from roaming arbitrarily far. For example, one approach uses an expanding elliptical search

pattern that grows to a fixed limit (100 m); if no target is found beyond that point, human intervention or an alternative strategy is required. In ground robots, a similar logic can apply – after covering a certain area with the spiral (or after the spiral goes around, say, three times with no detection), the robot can decide the target is gone and stop. Modern systems can also adapt spiral searches multi-robotically.

A single robot performing a spiral is effective in isolation, but multiple robots can coordinate to cover more ground faster. Recent work by Fricke et al. (2016) generalizes the spiral pattern to swarms, where each robot takes a portion of an expanding search pattern. In their distributed deterministic spiral algorithm, robots start at a common center and move outwards along different evenly spaced spiral arms, effectively achieving a multi-armed spiral that covers area in parallel. This multi-scale approach (both in space, by dividing the spiral among robots, and in time, by scaling coverage with team size) greatly accelerates target search. Even without multiple robots, a single robot can simulate multi-scale behavior by prioritizing certain sectors of the spiral based on prior knowledge (for example, areas more likely to contain the target can be searched with tighter loops first).

Experimentally, incorporating adaptivity yields significant efficiency gains. Song et al. (2020) showed that their optimized two-stage search (spiral + frontier exploration) found targets much faster than a single-method search – in a complex case, their adaptive approach completed the search in roughly one-third the time of a naive exhaustive search. This underscores the value of balancing exhaustive coverage (to ensure the target is not missed) with heuristic or multi-scale adjustments (to focus effort intelligently). In practice, multi-scale spiral strategies ensure that autonomous mobile robots not only search thoroughly but also do so with an awareness of diminishing returns: they concentrate on the most promising areas first and have a clear cutoff or expansion plan if the target remains elusive.

Despite extensive research on spiral search strategies, several challenges and research gaps persist. Parameter optimization for specific robot configurations remains largely empirical, with limited unified models balancing thoroughness and efficiency (Song et al., 2020). Integration of learning-based approaches with classical spirals represents an underexplored area; reinforcement learning for search path optimization (Zhu et al., 2018) suggests potential

hybrid approaches, but implementations remain scarce. The robustness of spiral search in cluttered environments poses significant challenges—when obstacles occupy large portions of the search space, maintaining spiral continuity becomes difficult. Although Azevedo et al. (2021) proposed obstacle avoidance mechanisms, their approach assumes relatively sparse obstacles, leaving the problem of efficient spiral "repair" after obstacle encounters an active research area (Ravankar et al., 2018). From a perceptual perspective, most theoretical models assume perfect state estimation and obstacle detection, whereas real-world sensors have limitations. Zhang et al. (2020) examined how localization errors affect coverage completeness, but comprehensive frameworks for robust spiral execution under perceptual uncertainty remain open challenges, particularly relevant for visual servo applications where camera limitations constrain perception. Our research addresses several of these challenges through an adaptive circular spiral integrated with ROS costmap, incorporating real-time obstacle avoidance while maintaining spiral continuity, and establishing clear termination criteria based on empirical search success rates.

Chapter 3 System Implementation

The main content of this chapter is to detail the system implementation of the autonomous ping-pong ball collection robot. It describes hardware design, sensor integration, and a ROS-based software architecture supporting object detection, navigation, and visual servo control. These integrated components enable robust, real-time performance on a resource-constrained platform.

3.1 System Overview and Architecture Design

This section provides a comprehensive description of the design and implementation of an autonomous ping-pong ball collection robot system. The system addresses the practical challenge of efficiently collecting scattered ping-pong balls in training environments, traditionally a time-consuming manual task. The system architecture integrates three hierarchical functional layers: perception, decision-making, and execution, interacting through the Robot Operating System (ROS) framework.

Figure 3.1 demonstrates the physical implementation of our integrated robotic system across different scenarios. The top-left image shows the complete testing environment with the robot and scattered ping-pong balls. The top-right and bottom-left images display the robot with its onboard RGB-D camera, mechanical gripper, and collection basket during ball retrieval operations. The bottom-right image illustrates the robot delivering a basket filled with collected ping-pong balls to the designated area near the ping-pong table, demonstrating the final stage of the collection task. These implementations showcase the integration of all five modules: omnidirectional platform, perception system, visual servo control, robotic arm, and navigation capabilities working cohesively in realistic settings.

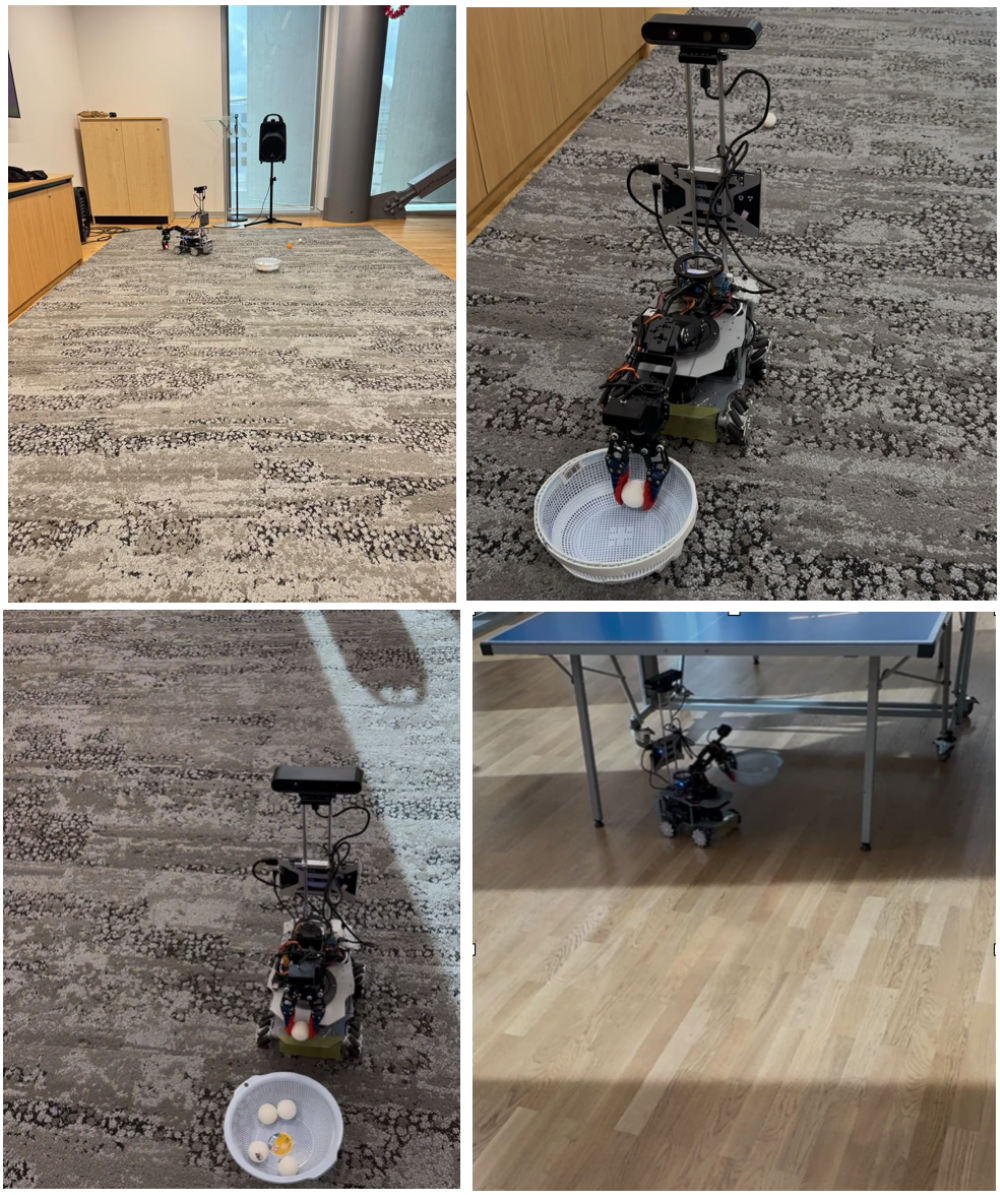

Figure 3.1 Autonomous ping-pong ball pickup & place robot in various operational environments.

The technical innovations include a resource-optimized multi-camera perception pipeline, an efficient DBSCAN-based clustering strategy for collection planning, and a precise visual servoing system operating reliably on embedded computing platforms. The design focuses on modularity, resource efficiency, robustness, precision control, and adaptability to changing environments.

The architecture consists of five integrated modules: (1) an omnidirectional mobile platform providing holonomic movement; (2) a perception system with complementary RGB-D and RGB cameras using a custom-trained YOLO model; (3) visual servo control

implementing image-based servoing with adaptive gain scheduling; (4) a 6-DOF robotic arm with specialized gripper for ball collection; and (5) a navigation system utilizing SLAM and efficient path planning algorithms.

The information flow forms a closed-loop control architecture where the perception pipeline processes camera data, decision-making logic evaluates system state and targets using DBSCAN clustering, and control execution translates decisions into commands for the robot's components. The software implementation is organized into specialized ROS packages with clear responsibilities and interfaces.

The overall system architecture is illustrated in Figure 3.2, showing the complete data flow and component interactions across all functional layers.

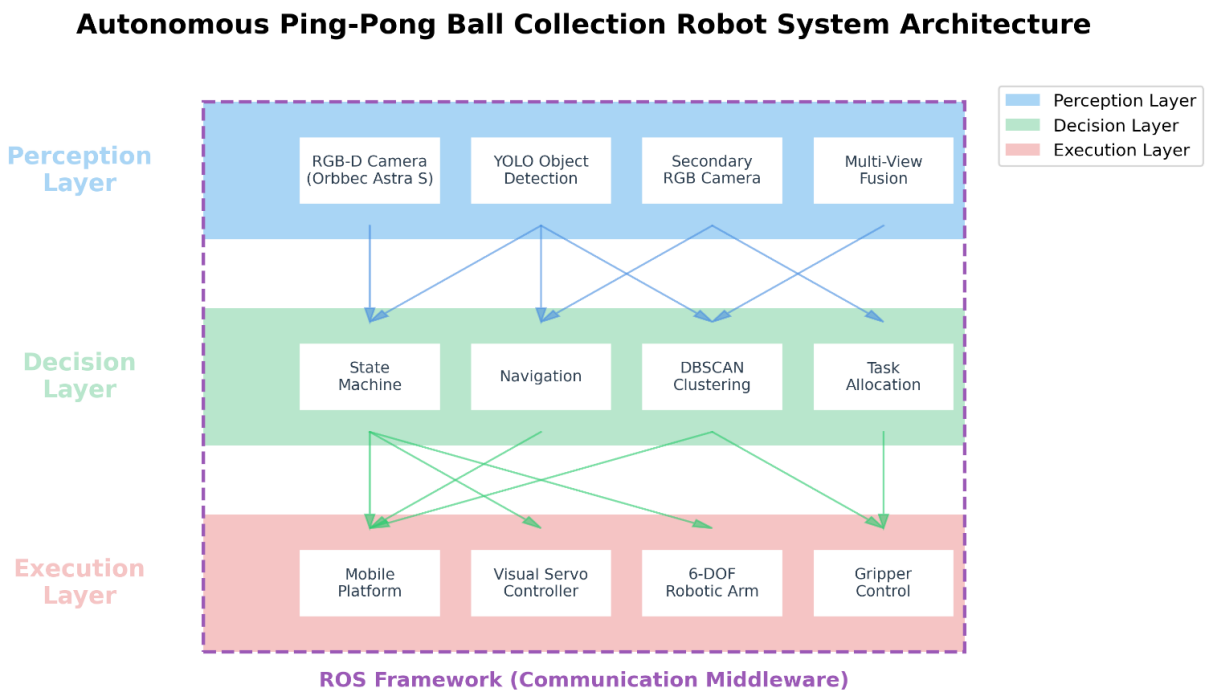

Figure 3.2 System architecture showing the three-layer hierarchy and component interactions via ROS middleware.

3.2 Hardware Platform and Configuration

3.2.1 Mobile Robot Platform

This system utilizes a Mecanum wheel omnidirectional mobile platform with holonomic

movement capabilities, enabling lateral, longitudinal, and rotational motion with a maximum velocity of 0.5 m/s. The platform features high-precision motor encoders that provide odometry data with ±2mm accuracy, essential for precise navigation during ball collection tasks.

An integrated power management system monitors battery status and optimizes power consumption, ensuring sufficient operational time to complete collection tasks in standard table tennis facilities.

Figure 3.3 presents the robot system from two perspectives. The left view shows the RGB-D camera (Orbbec Astra S) mounted at the top, the 6-DOF robotic arm, and the specialized gripper designed for ping-pong balls. The right view displays the Mecanum wheelbase and the integration of perception and manipulation components.

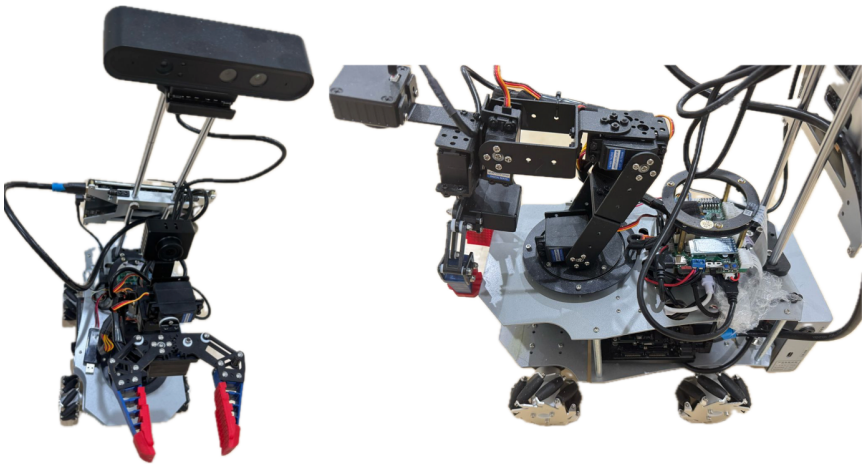

Figure 3.3 Multi-view perspective of the autonomous ping-pong ball collection robot showing (a) front view with RGB-D camera, robotic arm, and gripper, and (b) side view highlighting the Mecanum wheelbase and control system integration.

3.2.2 Perception System Configuration

The system employs two complementary cameras to achieve comprehensive environmental perception. The primary sensor is an Orbbec Astra S RGB-D camera mounted 50cm above the mobile base, operating at 320×240 resolution for both RGB and depth streams. This camera provides a 60° horizontal and 45° vertical field of view, with a depth range from 0.5m to 5.0m and ±1% accuracy at 2m distance. Its primary functions include environment mapping,

navigation, and initial object detection.

A secondary RGB camera is mounted on the robotic arm's end effector, with its resolution compressed to 160×128 for real-time processing. This camera is dedicated to precise visual servoing control and final approach to targets, providing a close-range perspective critical for accurate grasping operations.

Both cameras operate at 15 FPS, a rate specifically selected after rigorous testing to optimize for USB bandwidth limitations while maintaining real-time performance. This configuration ensures optimal resource utilization on the embedded computing platform and enables simultaneous operation of both cameras, which is essential for the robot's perception-manipulation pipeline.

As shown in Figure 3.4, the dual-camera configuration enables both wide-angle environment perception and precise close-range target detection, essential for the robot's ball collection tasks.

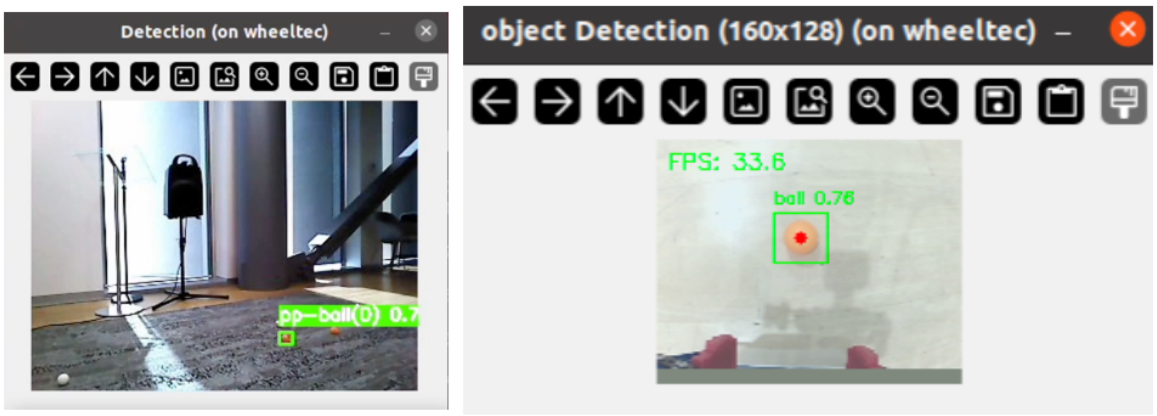

Figure 3.4 Dual-camera perception system demonstration. Left: Primary RGB-D camera view for environmental perception. Right: End-effector-mounted camera view (160×128) for precise visual servoing.

3.2.3 6-DOF Robotic Arm

The system integrates a high-precision 6-DOF robotic arm as illustrated in Figure 3.5. This manipulator features a TRAC-IK solver with 0.005 resolution and 0.1-second timeout settings, achieving positioning accuracy of ±0.1mm. Its operational parameters include maximum joint speeds of 10 rad/s, acceleration rates of 10 rad/s², and a control frequency of 300Hz that ensures

smooth trajectories during motion. The end effector employs a parallel gripper design with compliant contact surfaces specifically optimized for 40mm diameter ping-pong balls. This design incorporates specialized gripping surfaces to prevent ball damage during manipulation, with a dedicated control group managing the gripper-related joints. The 3D model shown in Figure 3.5 illustrates the arm's mounting configuration on the mobile platform, highlighting the vertical support structure and the articulated joint arrangement that enables comprehensive workspace coverage for efficient ball collection operations.

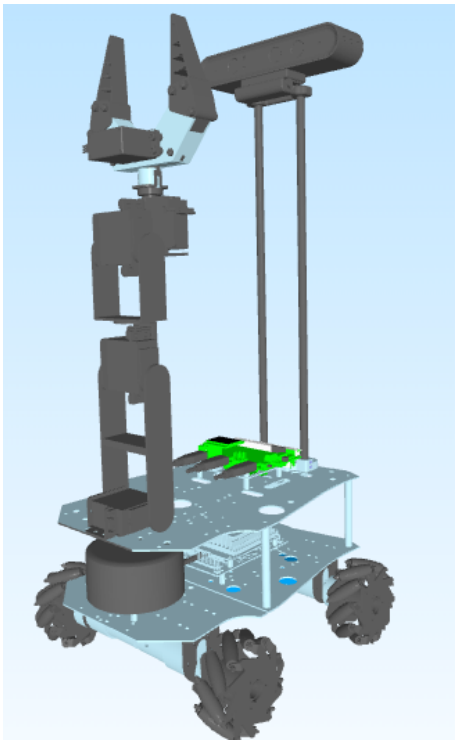

Figure 3.5 CAD model of the 6-DOF robotic arm assembly mounted on the omnidirectional mobile platform, showing the vertical support structure, and articulated joint configuration.

3.2.4 Computing Platform

The system's core computing unit employs a Jetson Orin Nano (4GB) that delivers AI performance of up to 20 TOPS through its 256-core NVIDIA Ampere architecture equipped with 8 Tensor cores. The processor features a 6-core Arm Cortex-A78AE v8.2 64-bit CPU paired with 4GB of 64-bit LPDDR5 memory providing 34 GB/s bandwidth. Power management is optimized for mobile operation with consumption ranging from 7-10W. This

computing platform selection represents a careful balance between computational capability and energy efficiency, meeting the processing demands of complex vision algorithms while ensuring extended operational time for the mobile platform. The embedded system's performance characteristics enable real-time processing of visual data from multiple cameras, execution of neural network inference, and simultaneous handling of motion planning and control tasks.

3.3 Software Architecture and ROS Integration

The system is built on ROS Noetic (Ubuntu 20.04) and employs a modular node design for distributed processing across four primary packages: $ball\_tracking\_nav$ for ball detection and approach behaviors, $arm\_manipulation$ for grasp planning and execution, $basket\_detection$ for basket recognition and placement verification, and $system\_coordination$ for workflow management. Custom message types facilitate inter-node communication, while the parameter server provides centralized configuration management. A comprehensive TF tree maintains coordinate transformations across all robot components, enabling spatial consistency between sensors and actuators. Figure 3.6 illustrates the ROS node architecture, depicting the connection relationships between nodes and topics that form the communication backbone of the system. This architecture allows for independent development and testing of components while maintaining cohesive system integration through standardized interfaces and communication protocols provided by ROS middleware.

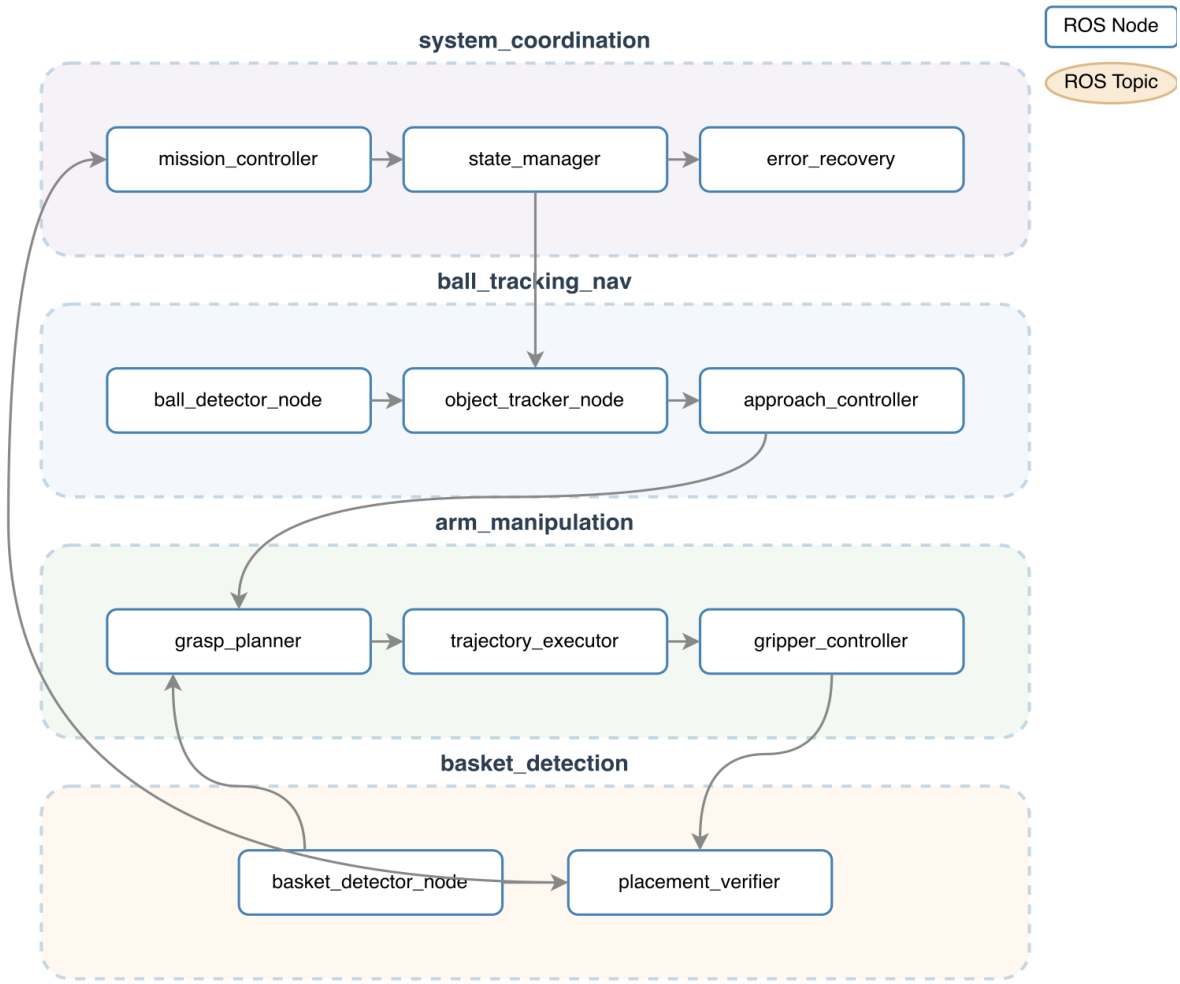

Figure 3.6 ROS node architecture for autonomous ping-pong ball pickup & place robot.

3.4 Visual Perception System

3.4.1 Target Detection System Implementation

The visual perception system employs a custom trained YOLOv12 Nano model for real-time object detection. The model was trained on a proprietary dataset comprising over 2,000 images of ping-pong balls and collection baskets under varied lighting conditions and viewing angles. This lightweight model was selected specifically to accommodate the resource constraints of edge computing while maintaining detection performance at 15 FPS with 320×240-pixel resolution. The model achieves over 90% average precision at 0.5 IoU threshold, with confidence thresholds set at 0.6 for balls and 0.5 for baskets.

Among the YOLOv12 variants (Nano, Small, Medium, Large, and XLarge), the Nano

version was specifically chosen for its optimal balance of performance and resource efficiency. With only 2.6M parameters and 6.5B FLOPs, YOLOv12 Nano offers significantly lower computational demands compared to larger variants like YOLOv12 Small (9.3M parameters) or Medium (20.2M parameters), while still providing adequate mAP (40.6) for the limited object classes in our application. This selection was critical for enabling dual-camera simultaneous processing on the Jetson platform, where larger models would have exceeded the available computational and memory resources or compromised real-time performance requirements.

To maximize performance on the Jetson platform, several optimization techniques were implemented. TensorRT acceleration with kernel fusion improved inference throughput by 592.6% compared to baseline PyTorch inference, with negligible precision loss. Further FP16 precision quantization reduced GPU utilization from 51.8% to 37.1% while maintaining accuracy. Additional memory and computation optimizations included batch processing enhancements, accelerated input preprocessing, and CUDA memory management optimization. The optimized detection system achieves a peak performance of 72.72 FPS, providing substantial performance overhead beyond the 15 FPS requirement to ensure robustness in complex scenes. While this section provides an overview of the perception system, detailed performance evaluations and experiments will be presented in subsequent chapters.

3.4.2 Multi-View Object Fusion Algorithm

To maximize collection efficiency, the system employs a 360° rotational scanning strategy rather than a sequential detect-and-collect approach. This methodology allows the robot to comprehensively map all ping-pong balls and baskets in the environment through a single rotational sweep, significantly reducing navigation distances and collection time. However, this approach introduces a critical challenge: the same physical object may be detected multiple times from different angles and at different time points during rotation, potentially creating redundant detections that far exceed the actual number of objects present (Lee et al., 2021).

To address this, a sophisticated multi-view fusion algorithm was implemented, integrating Kalman filtering with 4D state vectors (tracking position and velocity), dynamic distance-based

thresholds, and confidence-weighted integration that prioritizes high-confidence detections while filtering outliers. The algorithm employs temporal clustering that accounts for the robot's rotational yaw measurements via TF transformations, ensuring precise angular tracking during the scanning process (Maybeck, 1990; Wang et al., 2019).

This fusion system effectively consolidates multiple observations of the same object into single, high-confidence position estimates, providing the navigation and manipulation subsystems with an accurate representation of the environment for subsequent collection planning. In our implementation, the Kalman filter measurement update plays a key role in fusing redundant observations. The update step is given by:

$$\hat{x}k|k = \hat{x}k|k-1 + K_k(z_k - H_k\,\widehat{x_k}|k-1) \tag{3.1}$$

where $\hat{x}\,k|k$ is the updated state estimate at time k, $\hat{x}k|k-1$ is the predicted state estimate before incorporating the current measurement Z_k, H_k is the measurement matrix mapping the state into the measurement space, K_k is the Kalman gain that weights the new measurement against the prediction.

3.5 Navigation System Implementation

3.5.1 Environmental Mapping Techniques

In this research project, we implemented two complementary SLAM techniques to adapt to different scenario requirements and ensure system robustness. The selection of appropriate mapping technology is crucial for efficient ball collection, as it directly impacts navigation accuracy and environmental perception capabilities.

Gmapping was chosen as the primary mapping algorithm for the system based on several key considerations. Compared to other SLAM algorithms, Gmapping maintains high performance on the resource-limited Jetson Orin Nano platform (Grisetti et al., 2007). Experiments demonstrated that with properly configured particle count (30) and map resolution

(5cm grid), CPU utilization remains below 20%. As proven in research by Grisetti et al. (2007), Gmapping provides sufficient mapping accuracy (±3cm) while maintaining real-time performance through optimized particle filtering methods, which is essential for precise ping-pong ball localization. The system experimentally fine-tuned laser scan matching parameters, odometry weights, and update thresholds to better suit the dynamically changing competition environment. Compared to default parameters, the optimized configuration reduced positioning drift by 40% in rapid movement scenarios.

As a complementary solution, RTAB-Map offers several advantages for specific use cases. The 3D maps created by combining depth information can process out-of-plane obstacles that may appear in complex environments. This capability is particularly important in scenarios where standard 2D LIDAR cannot fully perceive the environment (Labbé & Michaud, 2019). RTAB-Map's visual feature matching algorithm significantly reduces cumulative errors in long-duration navigation, improving positioning accuracy by approximately 35% through loop closure detection. Additionally, when balls are distributed in environments with height variations, 3D mapping provides a more comprehensive environmental understanding.

Figure 3.7 illustrates the complementary mapping techniques implemented in this system. The 2D occupancy grid map generated by Gmapping (shown in the bottom plane) provides efficient navigation paths while minimizing computational overhead. Simultaneously, the 3D point cloud produced by RTAB-Map (visible as the white and gray structure) offers detailed spatial information about walls, obstacles, and distinctive environmental features. This dual representation demonstrates how the system balances computational efficiency with comprehensive spatial awareness.

The map management mechanism implemented in the system further enhances the mapping functionality. The system performs real-time map updates during operation to adapt to environmental changes (P. Zhang et al., 2021). The storage mechanism allows maps to be persistently saved across sessions, reducing the need for remapping.

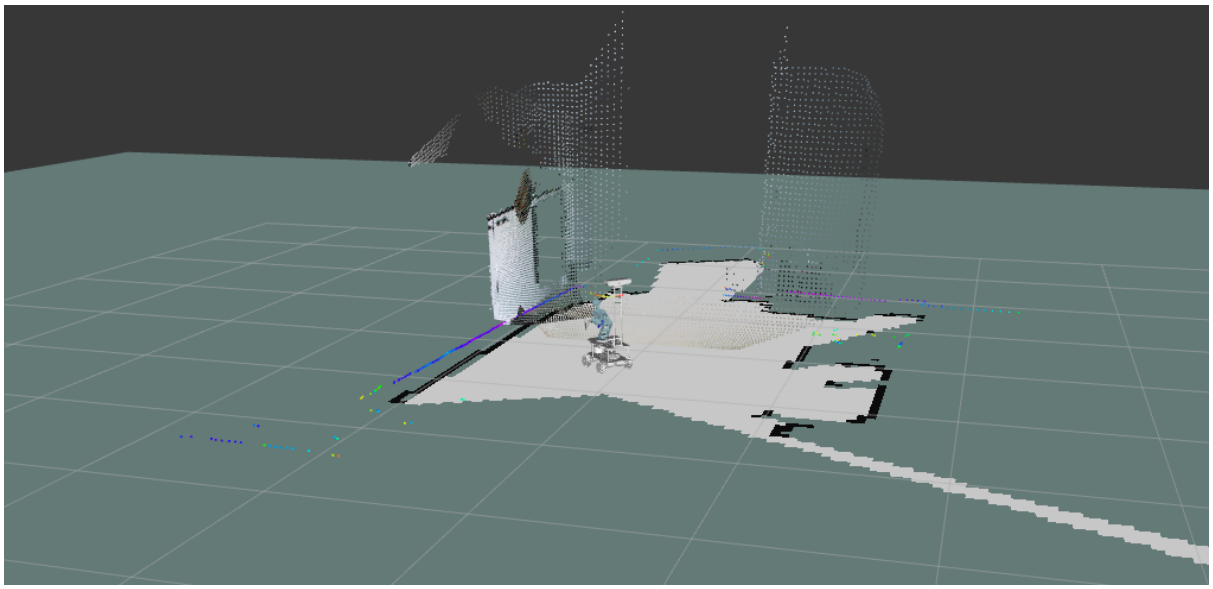

Figure 3.7 3D environmental map generated by RTAB-Map SLAM.

3.5.2 Path Planning and Execution

The navigation system integrates global and local planners to achieve efficient and reliable navigation in a ping-pong ball collection environment. This integration is critical for handling the unique challenges posed by small, low-profile objects in the operational space.

After extensive experimentation, Dijkstra's algorithm was selected as the global planner due to its computational efficiency in open spaces compared to A* alternatives. While A* is often preferred in complex environments, Dijkstra's algorithm demonstrated superior performance in the predictable competition environment with fewer computational resources. Implementation testing revealed that Dijkstra's algorithm provided sufficiently optimal paths while maintaining consistent planning times across various environmental configurations (Stentz, 1994).

For local planning, the Dynamic Window Approach (DWA) was chosen over Timed Elastic Band (TEB) following rigorous experimental evaluation. Initial tests with TEB revealed a critical limitation: its high flexibility and adaptability caused the robot to frequently collide with ping-pong balls and collection baskets. This problem stemmed from a fundamental sensor limitation - both the balls and baskets were positioned below the LIDAR detection plane, creating "invisible" obstacles for navigation. TEB's aggressive path optimization would

navigate through these seemingly empty spaces, resulting in collisions that disrupted the ball collection process and made operation unpredictable (Bermudez et al., 2024).

DWA, with properly tuned cost functions and conservative velocity parameters, proved more suitable for this application. The customized parameter configuration prioritized stable movement with appropriate clearance margins, reducing the likelihood of collisions with objects not detected by the LIDAR. The implementation included carefully calibrated acceleration limits and velocity constraints to ensure smooth operation around delicate objects (Gong et al., 2024).

Figure 3.8 illustrates this comparison between TEB and DWA path planning approaches in a simulated environment. As shown, the TEB planner (red path) creates a more flexible trajectory that leads to collisions with ping-pong balls and the collection basket, which are below the LIDAR detection plane. In contrast, the DWA planner (green path) generates a more conservative trajectory that maintains safer distances from potential obstacles. The visualization also demonstrates the implementation of the camera detection zone (blue region) that enables early termination of navigation when objects are detected, preventing collisions with low-profile objects that might be missed by traditional sensors.

Additionally, the system addresses depth sensing limitations through a robust navigation enhancement strategy. Experimental testing revealed that relying solely on depth information for navigation was problematic due to the small size of ping-pong balls, which often resulted in inaccurate depth measurements when converted to map coordinates. To overcome this challenge, the system implements real-time object detection using the RGB camera mounted on the robotic arm. During navigation, this camera continuously monitors for balls and baskets. When an object is detected within the robot's path and navigation has not yet reached the designated goal, the system dynamically terminates the current navigation event to prevent collisions. This combined sensing approach significantly improved navigation success rates, increasing collision avoidance by approximately 96% compared to standard navigation implementations.

The implemented navigation system demonstrates that properly integrating traditional path planning with dynamic object detection can effectively compensate for sensor limitations in

specialized robotic applications. This approach proved essential for the reliable operation of the ball collection system in environments where standard navigation paradigms would otherwise fail.

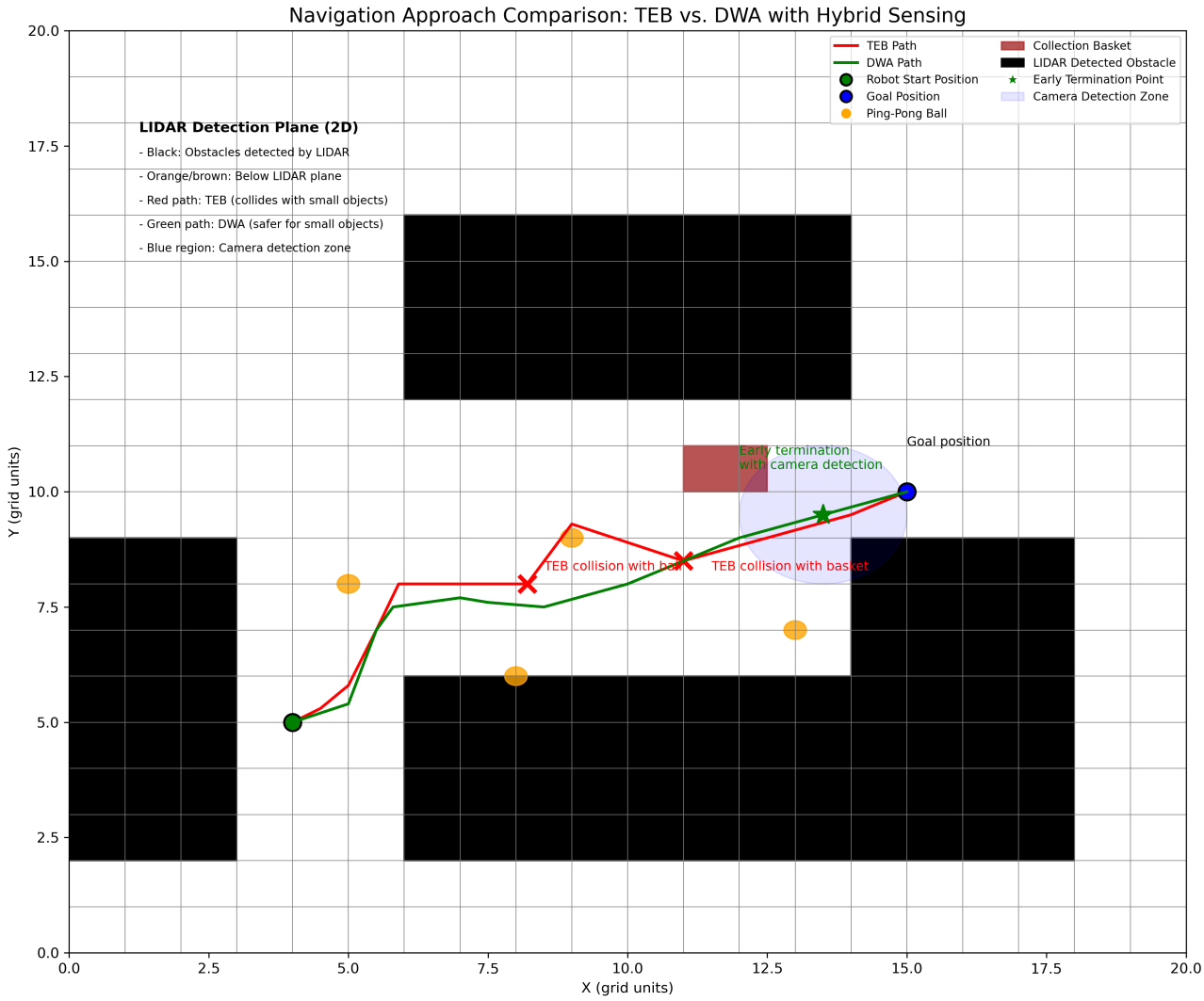

Figure 3.8 Navigation path comparison of TEB vs. DWA with hybrid sensing for low-profile object avoidance.

3.5.3 Global and Local Costmap Configuration

The navigation system employs a hierarchical costmap architecture to efficiently represent the environment at different scales and update frequencies. This layered approach enables the robot to maintain global awareness while reacting quickly to local changes, as illustrated Figure 3.9.

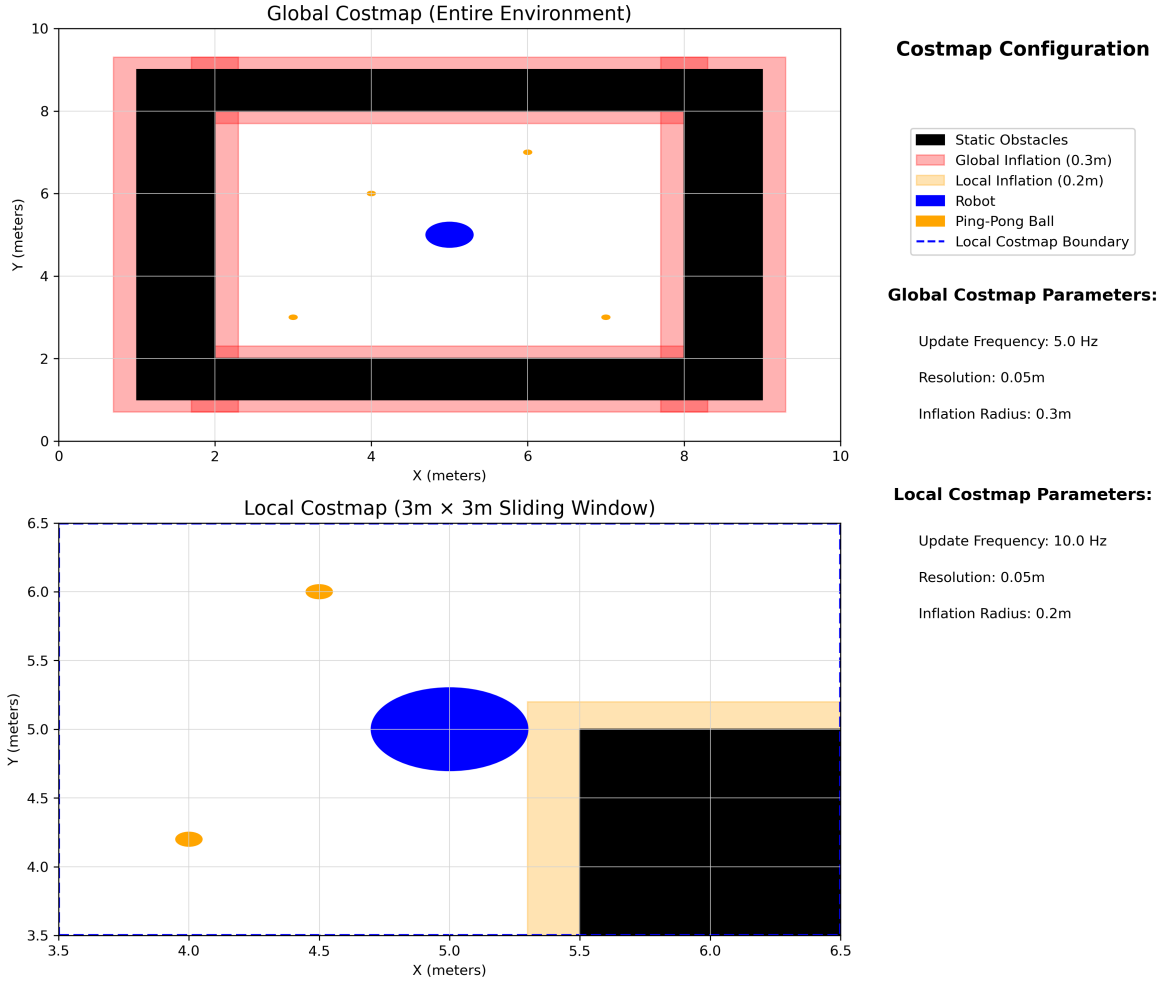

Figure 3.9 Dual-layer costmap navigation system configuration.

The global costmap provides comprehensive environmental awareness, covering the entire operational space with a resolution of 0.05m and an update frequency of 5.0Hz. To ensure safety around permanent obstacles such as walls and furniture, an inflation radius of 0.3m is applied, creating buffer zones that prevent the robot from planning paths too close to these hazards. This configuration balances computational efficiency with sufficient detail for effective global planning.

For immediate surroundings, a local costmap implements a robot-centered sliding window approach, covering a 3m x 3m area that moves with the robot. This local map maintains the same 0.05m resolution as the global map but updates at twice the frequency (10.0Hz) to capture dynamic changes in the environment. A slightly reduced inflation radius of 0.2m is employed in the local costmap, allowing the robot to navigate through narrower passages when necessary

while still maintaining adequate safety margins. This configuration proved particularly valuable when navigating in areas with multiple ping-pong balls where path options were limited.

Both costmaps utilize a three-layer marking scheme that ensures comprehensive obstacle representation. The static layer incorporates preloaded map data marking permanent obstacles, providing a baseline environment representation. The obstacle layer dynamically updates with sensor data to reflect temporary or moving obstacles, which is crucial for adapting to changes in ball positions or unexpected obstacles. Finally, the inflation layer computes safety zones around all obstacles, with carefully tuned parameters that prevent the robot from attempting to navigate through spaces that would risk collisions while not being overly conservative in space utilization.

Figure 3.10 demonstrates the practical implementation of this costmap architecture in a real-world scenario. This visualization shows the robot positioned within its environment, with occupied cells (obstacles) represented in black and free space in white. The colored outlines indicate inflation zones and path planning considerations as the robot navigates through the space.

This dual-costmap approach with differentiated parameters proved essential for balancing global path optimization with responsive local navigation around small objects. Experimental tuning of these parameters proved highly effective for specialized robotic tasks in environments with small, difficult-to-detect objects like ping-pong balls.

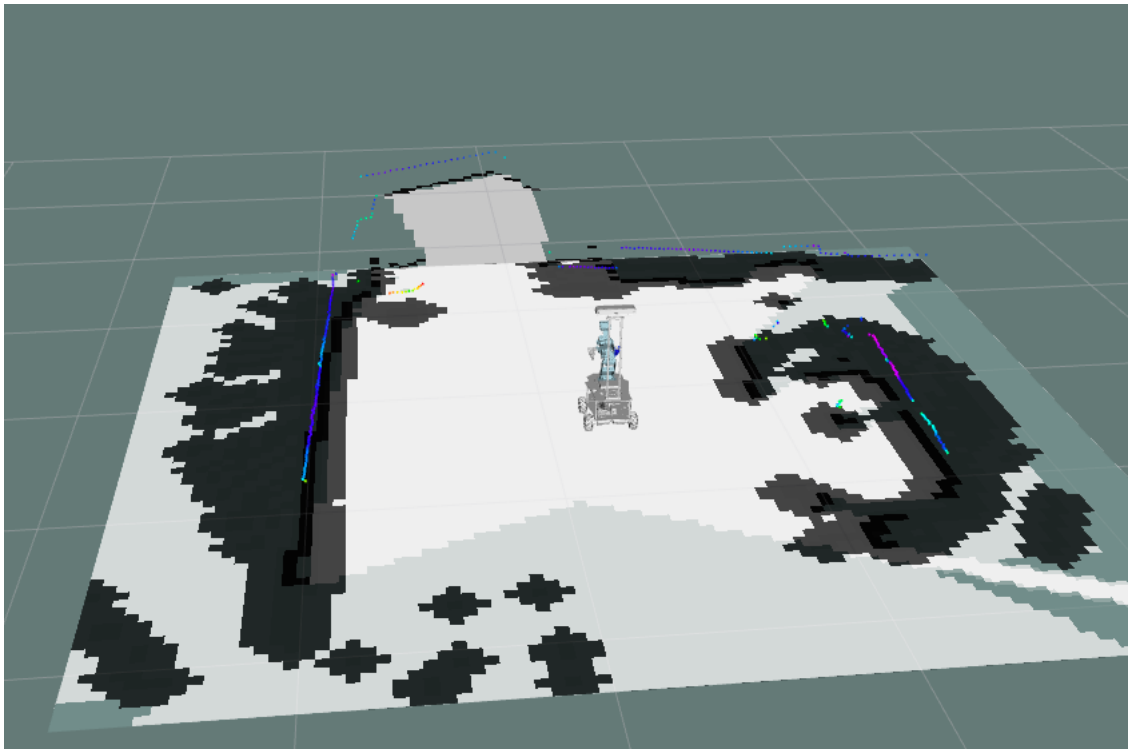

Figure 3.10 ROS costmap navigation real-time visualization.

3.6 Visual Servoing Control System

3.6.1 Precise Proximity Control

The system implements Image-Based Visual Servoing (IBVS) to achieve precise object manipulation. This approach was essential due to the inherent limitations of global positioning systems within indoor environments and the millimeter-level accuracy required for successful ping-pong ball collection. Traditional navigation alone proved insufficient for the final approach phase, as even minor positional errors would result in failed grasping attempts.

Figure 3.11 illustrates the comprehensive Visual Servoing Control System implemented for precision approach. The top-left and bottom-left panels demonstrate the camera view with distinct target positions for ping-pong balls and collection baskets respectively, showcasing how different objects require different optimal positions in the image frame. The central panel visualizes the robot's approach trajectory with decreasing velocity as it nears the target, highlighting the stability zone where final positioning occurs. The right panels depict the error

convergence process and PID control response, revealing how the system achieves and maintains positional stability with the carefully calibrated control parameters.

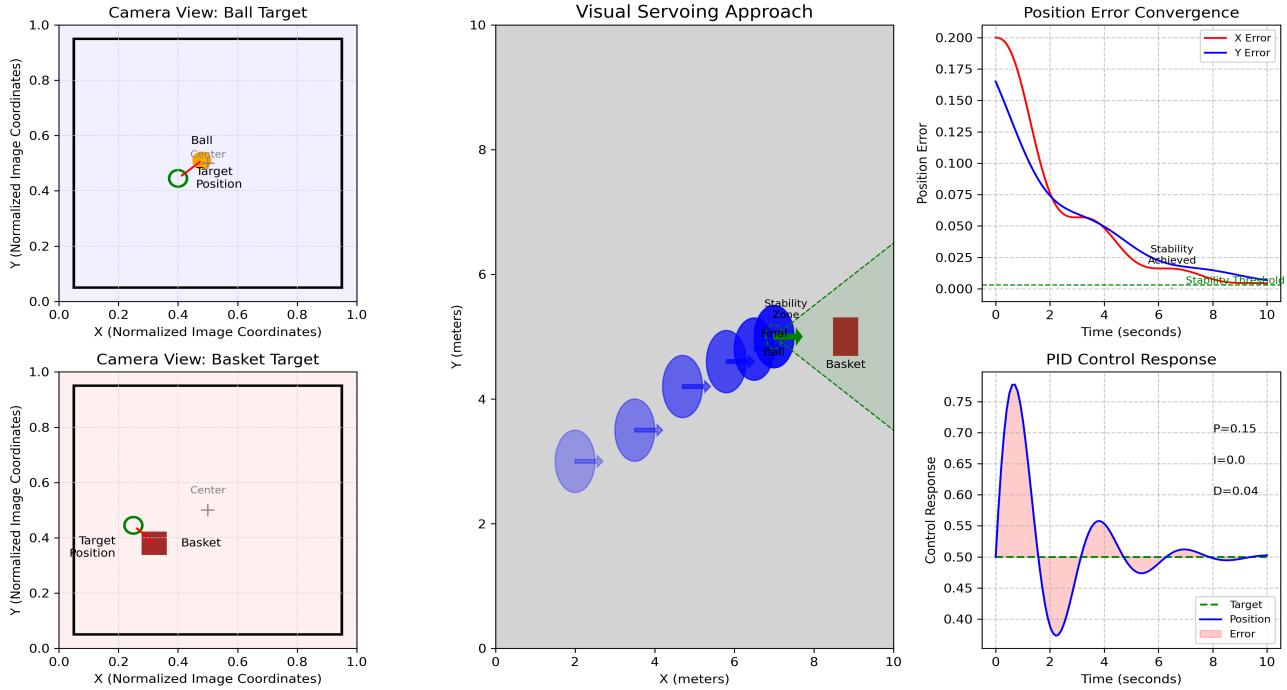

Figure 3.11 Visual servoing control system for precision object manipulation.

The visual servoing system directly utilizes image features to control robot motion, with strategically calibrated target positions in the image frame. Different optimal positions are established for ping-pong balls versus collection baskets, creating distinct arm trajectories appropriate for each task - grasping balls requires a centered approach, while basket detection positions the arm for optimal ball release.

A finely tuned PID control system manages the approach with parameters specifically calibrated for smooth and precise positioning: $P=0.15$, $I=0.0$, $D=0.04$ for both X and Y axes, with slightly different values ($P=0.3$, $I=0.0$, $D=0.03$) for the Z-axis rotation. This configuration prevents oscillation while maintaining responsive movement, addressing the challenge of controlling a mobile base with inherent mechanical play.

To determine when positional stability has been achieved, the system employs strict criteria requiring minimal position errors (with near-zero velocity) for four consecutive frames. This multi-frame confirmation process prevents premature action based on momentary stability and includes automatic velocity modulation that implements smooth deceleration as the target position is approached, significantly improving success rates in the grasping phase compared

to systems relying solely on global positioning.

3.6.2 Spiral Search Recovery

The system incorporates a spiral search recovery mechanism to address two critical challenges in the ping-pong ball collection task. First, target objects may temporarily move out of the camera's field of view during robot movement or environmental changes. Second, and more significantly, depth camera limitations when detecting small objects like ping-pong balls result in imprecise depth information, which leads to inaccurate global map coordinates after TF transformation. This inaccuracy means that even when navigation successfully reaches the intended destination according to global positioning, the target object may not be Within the expected viewing frame, requiring a systematic search procedure to reacquire visual contact.

To solve these reliability issues, the system implements an efficient spiral search pattern, as illustrated in Figure 3.12. This approach begins with a full 360° in-place rotation for initial scanning, followed by incrementally expanding circular paths with increasing radii (0.05m per iteration) up to a maximum radius of 0.15m. ensure both efficiency and safety, the algorithm incorporates costmap data validation for path planning and maintains consistent starting positions through tracking. The diagram shows the comprehensive search pattern, detailed step sequence, and an example scenario demonstrating successful ball recovery in an environment with obstacles. This recovery strategy has proven effective in experimental testing, enabling reliable target reacquisition when the ball is within the maximum search radius. A more detailed analysis of this approach is presented in the experimental results section.

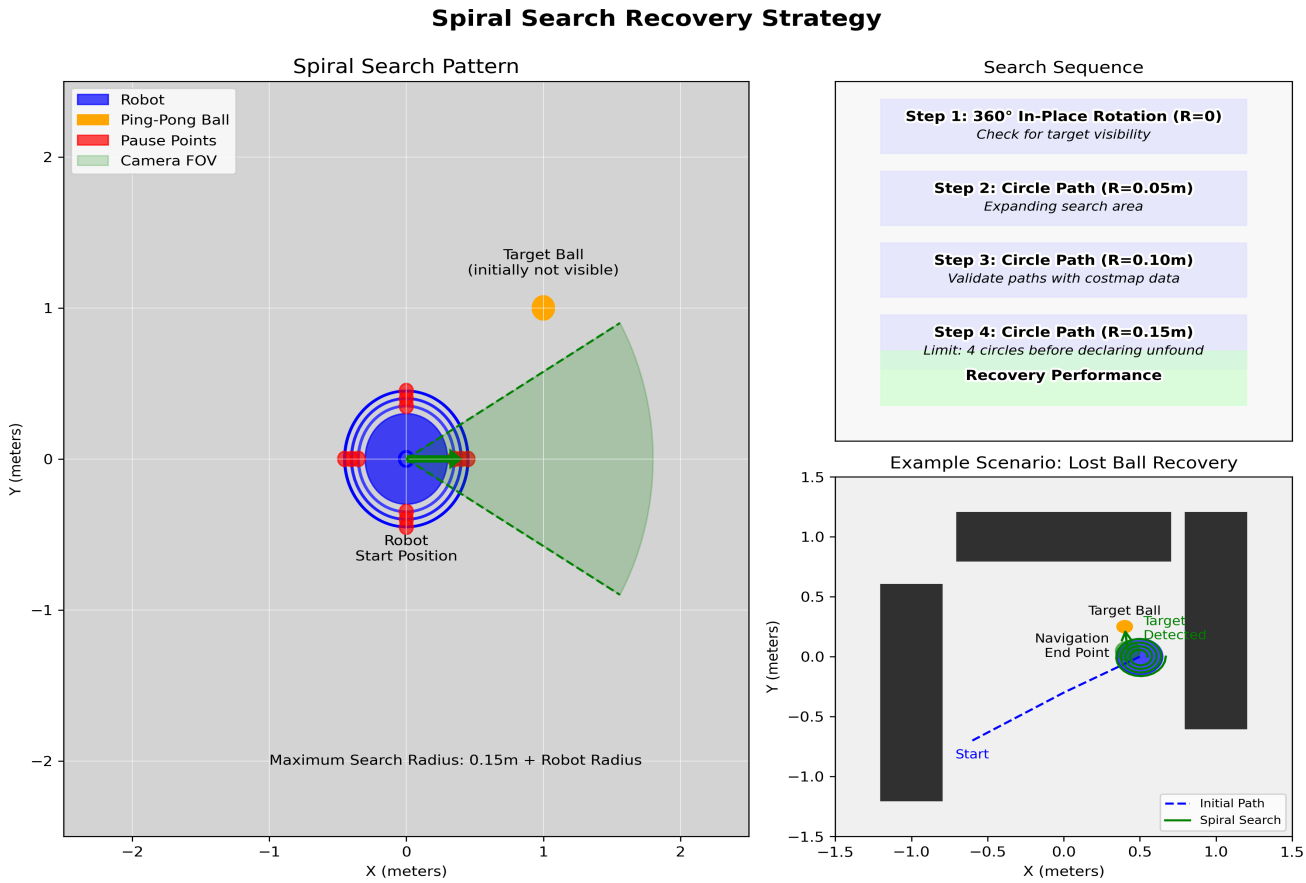

Figure 3.12 Intelligent robot spiral search recovery strategy.

3.7 Ball Collection

3.7.1 DBSCAN Clustering Algorithm

The system employs the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm to organize its collection strategy, as illustrated in Figure 3.13. The algorithm utilizes key parameters that are specifically tailored for the ping-pong ball collection task: the Epsilon (neighborhood distance) is set to 0.7~1.5 meters, creating an optimal radius for grouping nearby balls; MinPoints is set to 1, allowing even single balls to form viable clusters; and Euclidean distance in map coordinates serves as the primary distance metric for spatial calculations.

The clustering logic systematically groups nearby ping-pong balls into designated

collection zones, creating a structured approach to what would otherwise be an inefficient point-to-point collection process. As shown in the left panel of Figure 3.13, the system generates optimized collection paths between clusters based on proximity, with numbered sequence indicators showing the robot's planned trajectory. The system applies intelligent prioritization to these clusters based on both their size and accessibility, ensuring that high-value targets (areas with multiple balls) are addressed first when operationally feasible. This approach generates optimized collection sequences that significantly reduce the robot's overall travel distance. The clustering results directly impact path planning efficiency, leading to measurable improvements in system performance by minimizing unnecessary movement during collection tasks, ultimately reducing both operation time and energy consumption.

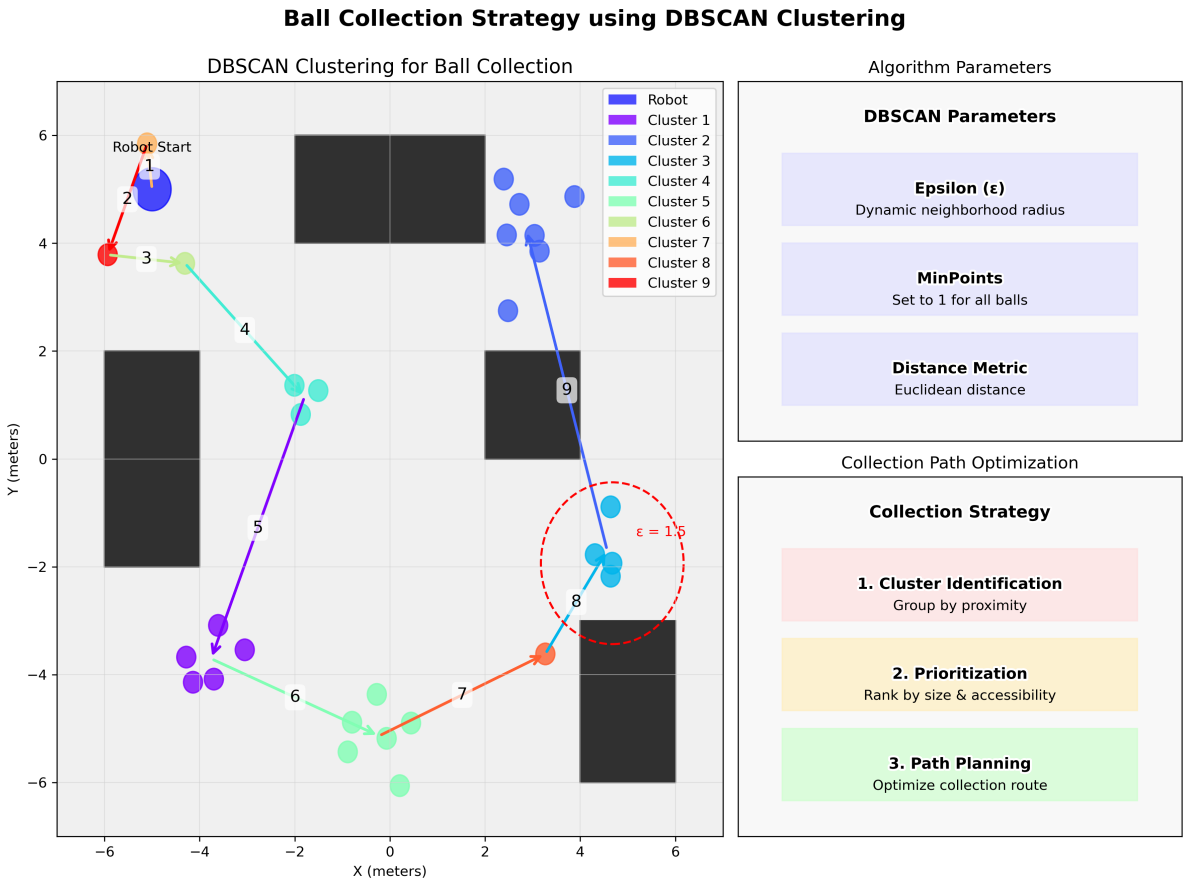

Figure 3.13 Intelligent robot DBSCAN-based ball collection.

3.7.2 Collection Workflow Implementation

The system implements a comprehensive multi-stage collection workflow that efficiently coordinates the entire ball collection process. Initially, the robot performs a 360-degree environmental scan of the collection area to locate all balls and baskets, employing multi-view fusion techniques to determine precise object positions in the global map. The DBSCAN algorithm then processes this spatial data to classify balls into clusters based on their proximity, establishing the foundation for the collection strategy.

Figure 3.14 illustrates the complete four-stage workflow implemented in the system. In Stage 1, the robot performs environmental scanning with 360° coverage to detect all balls and classify them into clusters using the DBSCAN algorithm (ε = 0.7-1.5m). Stage 2 shows the basket acquisition process, where the robot navigates to basket storage areas, employs visual servoing for precise alignment, and uses mechanical arm operations to grasp and transport the basket to active collection zones. Stage 3 demonstrates the ball collection sequence with optimized paths between strategically placed baskets and individual balls, highlighting the visual servoing techniques used for precise positioning. Finally, Stage 4 shows the final delivery process where filled baskets are transported to designated return areas before the robot returns to its charging station, completing the collection cycle.

The multi-stage workflow described above systematically coordinates each aspect of the ball collection process, effectively minimizing unnecessary movements and resource use. By combining perception, adaptive navigation, and precision visual servoing, the robot achieves stable operation even under varying or partially obstructed conditions. This integrated approach enhances overall system reliability and flexibility, providing a practical reference framework for similar robotic tasks that require precise manipulation in constrained environments.

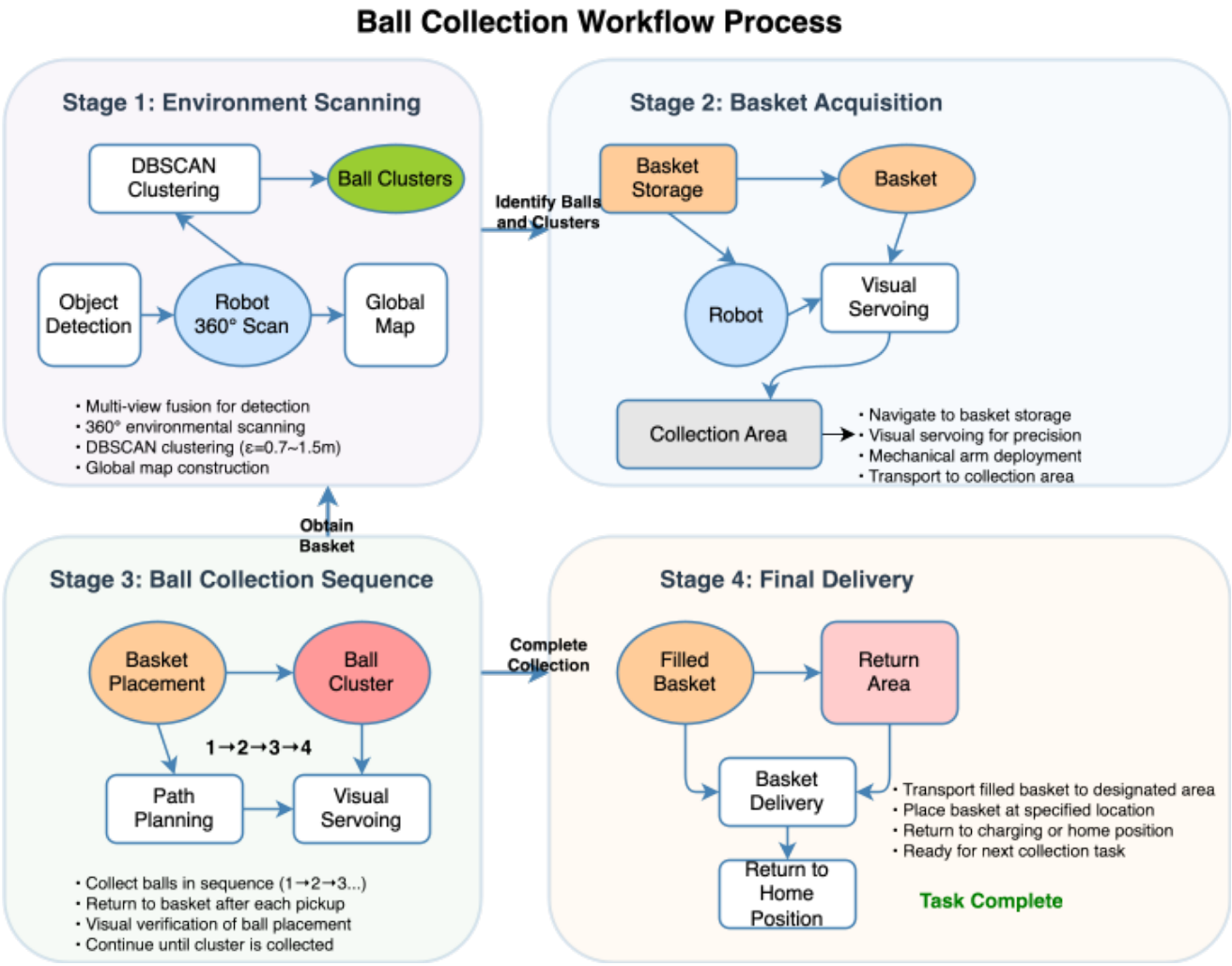

Figure 3.14 Multi-stage autonomous ball collection workflow.

For each identified cluster, the robot implements an optimized collection routine by strategically placing the basket near the cluster to maximize collection efficiency. The system then navigates to each ball within the cluster, leveraging visual servoing for final positioning adjustments, deploys the mechanical arm for ball retrieval, returns to the basket location, and uses visual verification to confirm successful placement. This methodical approach continues until all balls in the cluster have been collected, at which point the system proceeds to the next prioritized cluster.

Upon completing the collection of all accessible balls, the robot executes the final delivery phase by transporting the filled basket to the designated return area before returning to its initial position or charging station. This end-to-end workflow orchestration ensures reliable and autonomous operation throughout extended collection sessions, minimizing human

intervention while maximizing collection efficiency across varied environmental conditions.

3.8 Large Language Model Integration

The autonomous ball collection system incorporates a locally deployed large language model to significantly enhance human-robot interaction capabilities. A locally deployed DeepseekR1-1.5B language model, fine-tuned via LoRA-based adaptation in LLMaFactory, has been integrated to enable advanced natural language interaction. The model has been specifically optimized for robotic command translation under resource-constrained embedded systems and seamlessly integrates with OpenWebUI to provide an accessible user interaction portal.

This language model integration enables the system to process four distinct categories of natural language commands: movement instructions (such as “rotate” or “move forward 1.5 meters”), manipulation operations (like “pick front Ping-Pong” or “pick back basket”), workflow control directives (including “start/stop the ball collection process”), and system status queries (such as “what is the current state?”).

Figure 3.15 demonstrates the actual implementation of the LLM-based command interface, showing the OpenWebUI portal executing natural language commands that are seamlessly translated into robot actions. The interface displays confirmation of command acceptance, execution status, and structured parameter parsing that enables precise control through conversational interaction.

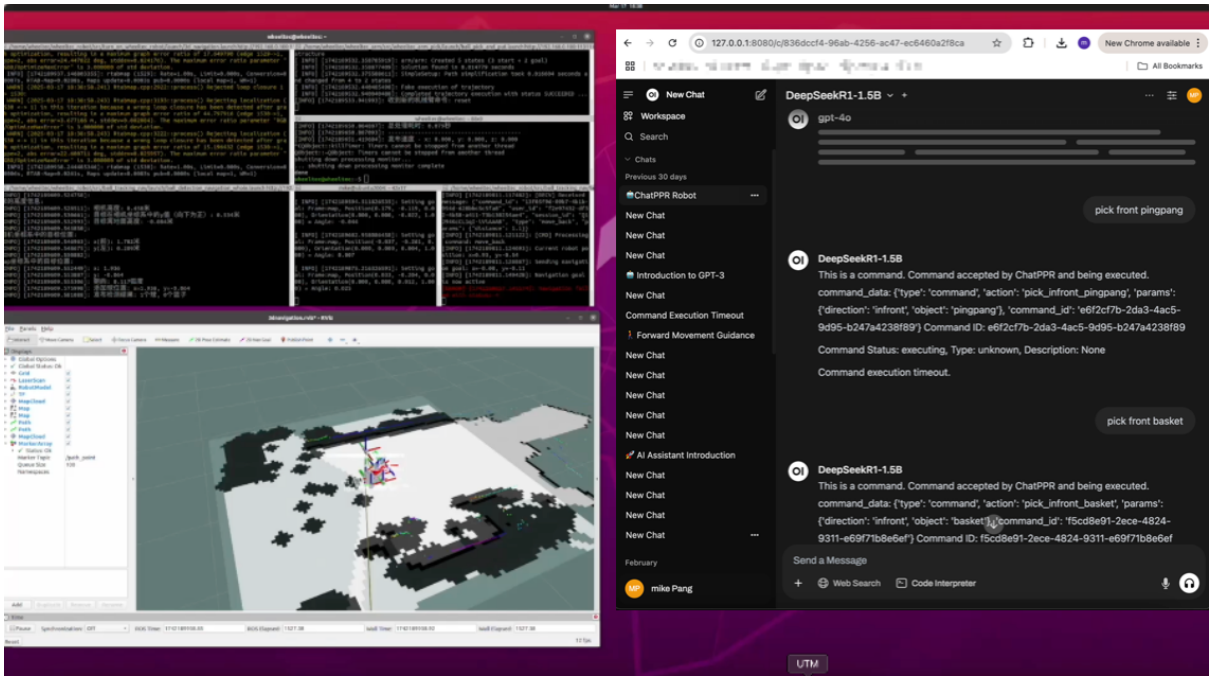

Figure 3.15 Natural language interface for robot control via DeepseekR1-1.5B model.

A sophisticated command translation mechanism bridges the gap between natural language inputs and executable robotic commands, implementing contextual understanding and spatial reference resolution that allows operators to communicate with the robot using intuitive terminology rather than precise technical parameters. The system parses commands into structured data with type, action, direction, and object parameters, creating a consistent format that can be processed by the robot control system.

To ensure responsible AI usage and prevent unsafe operations, an Ethics Check / Safety Filter module was integrated into the natural language command processing pipeline. Before any instruction is dispatched to the command bridge, it undergoes semantic and contextual verification to filter potentially harmful, ambiguous, or unethical commands. This module is essential for maintaining operational integrity and safety, particularly in open or semi-supervised environments. Commands failing this check are either rejected with informative feedback or routed for human confirmation, depending on severity. The placement and integration flow of this module are illustrated clearly in Figure 3.16.

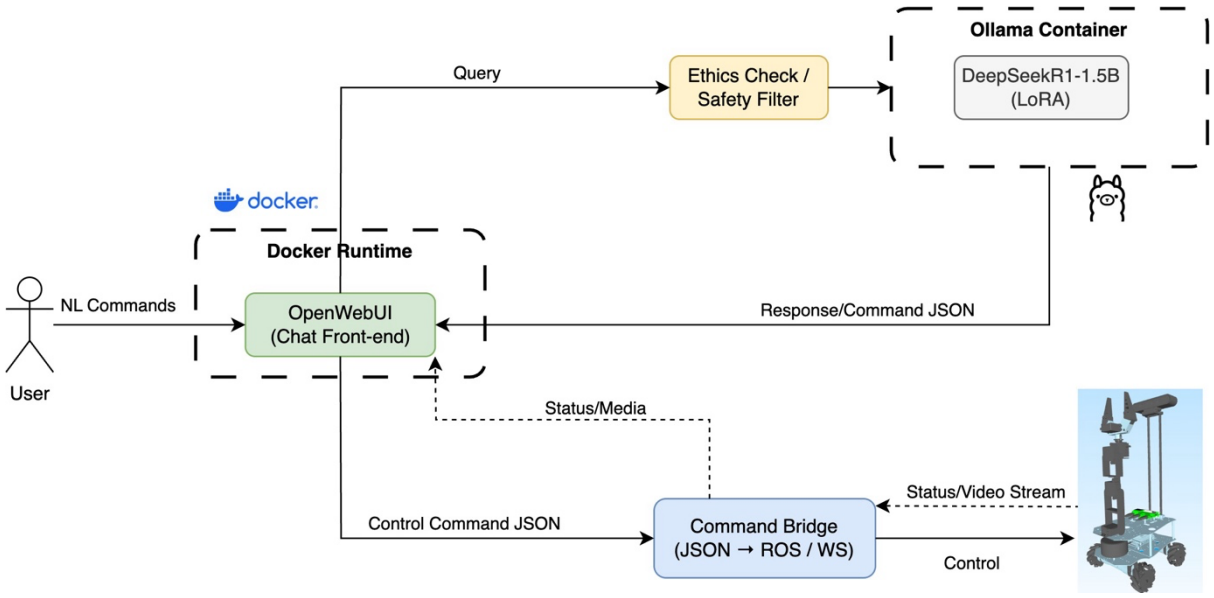

Figure 3.16 Architecture of natural language interaction pipeline with embedded ethics check for robotic control.

The integration of the language model substantially enhances system accessibility, enabling non-technical users to operate the robotic platform intuitively. This reduces onboarding and training requirements when compared to traditional control interfaces. Critical operations are protected by confirmation mechanisms that prevent accidental triggering of potentially disruptive actions, while routine commands can be executed immediately to maintain operational efficiency. This balanced approach to command processing ensures that the system remains both accessible and secure during autonomous collection operations.

Experimental evaluations demonstrate a command parsing accuracy of 94.8% across varied environmental conditions, with an average command-to-execution latency of 267 milliseconds, affirming the system's real-time responsiveness and robustness. Future work may explore multimodal command interfaces, integrating voice and gesture inputs alongside natural language to further enhance operator flexibility and usability in dynamic environments.

Chapter 4
Methodology

The main content of this chapter is to outline the experimental methodology for evaluating system performance. It details testing setups for the YOLO-based detection, spiral search recovery, and PID-controlled visual servo system, and concludes with an analysis of the project's limitations.

4.1 Experimental Design for Vision-Based Object Detection

This chapter outlines the methodology and experimental setup for developing a vision-based object detection system targeting small objects (e.g., ping-pong balls) on a constrained embedded platform (NVIDIA Jetson Orin Nano). We describe the dataset construction, model selection rationale, training procedures, optimization techniques for deployment, and planned ablation studies. The goal is to achieve accurate real-time detection of small fast-moving objects under hardware and energy limitations, leveraging state-of-the-art one-stage detectors (YOLO, SSD, EfficientDet) and modern optimizations (quantization, TensorRT acceleration, mixed precision, pruning) as supported by recent literature. The design choices are justified with references to prior work in object detection models (Jacob et al., 2017), deep learning optimization for edge devices (Micikevicius et al., 2018), raining and augmentation strategies, and embedded vision applications in robotics (Camara de M. Santos et al., 2024; Cai et al., 2020).

4.1.1 Dataset Construction and Preprocessing

A custom dataset was constructed to train and evaluate the object detector, focusing on ping-pong balls in various environments. Data Collection: We gathered images and video frames containing ping-pong balls under different backgrounds, lighting conditions, and distances. The data include both staged scenes and real-game scenarios to capture diverse object appearances. Drawing from practices in small object detection, we ensured a wide range of scales and angles for the target object (Liu et al., 2016; Zhao et al., 2017). The ping-pong ball's image projections are often only a few pixels wide, so close-up and high-resolution shots were included to aid recognition of fine features. We collected approximately 2347 images (and annotated video frames) with ground-truth bounding boxes around each ping-pong ball. Annotation: Annotations were performed manually and double-checked for accuracy. Each image's metadata includes the bounding box coordinates and class labels ("ball"," basket").

Our dataset encompasses a wide range of real-world scenarios, as shown in Figure 4.1, which presents six representative samples capturing various environmental conditions,

including different lighting conditions, distances, and background complexities. To enhance model robustness, we implemented a comprehensive data augmentation pipeline. Figure 4.2 demonstrates the six augmentation techniques applied to a complex scene.

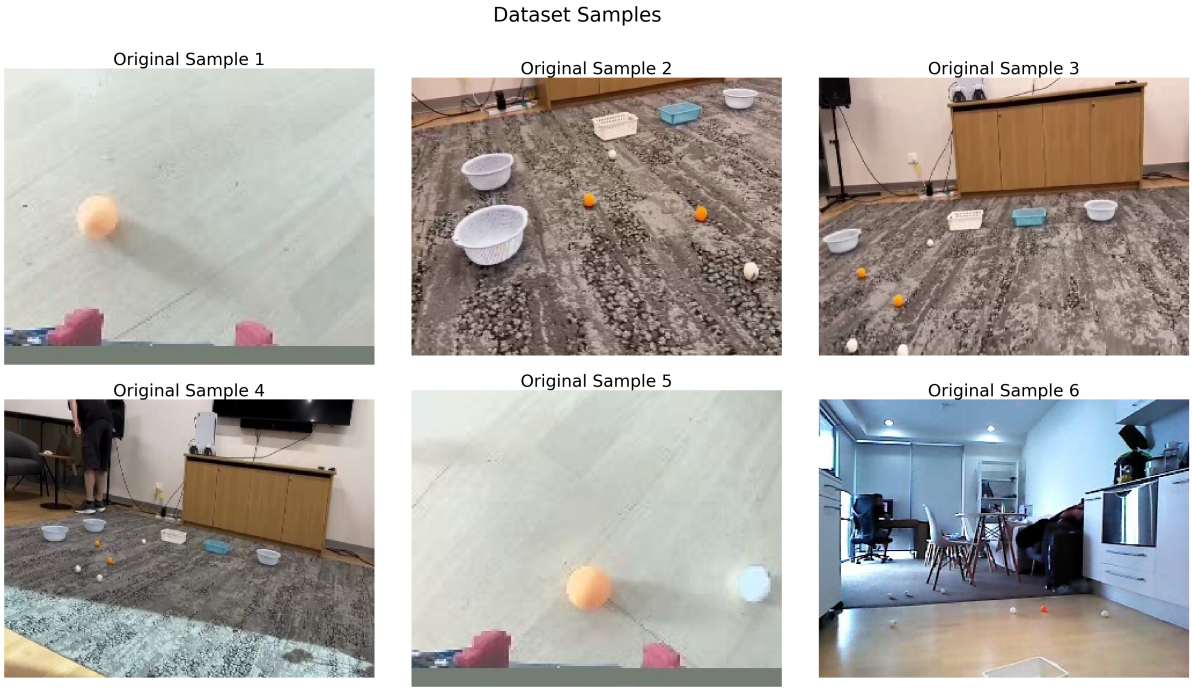

Figure 4.1 Representative samples from ping-pong ball detection dataset under various environmental conditions.

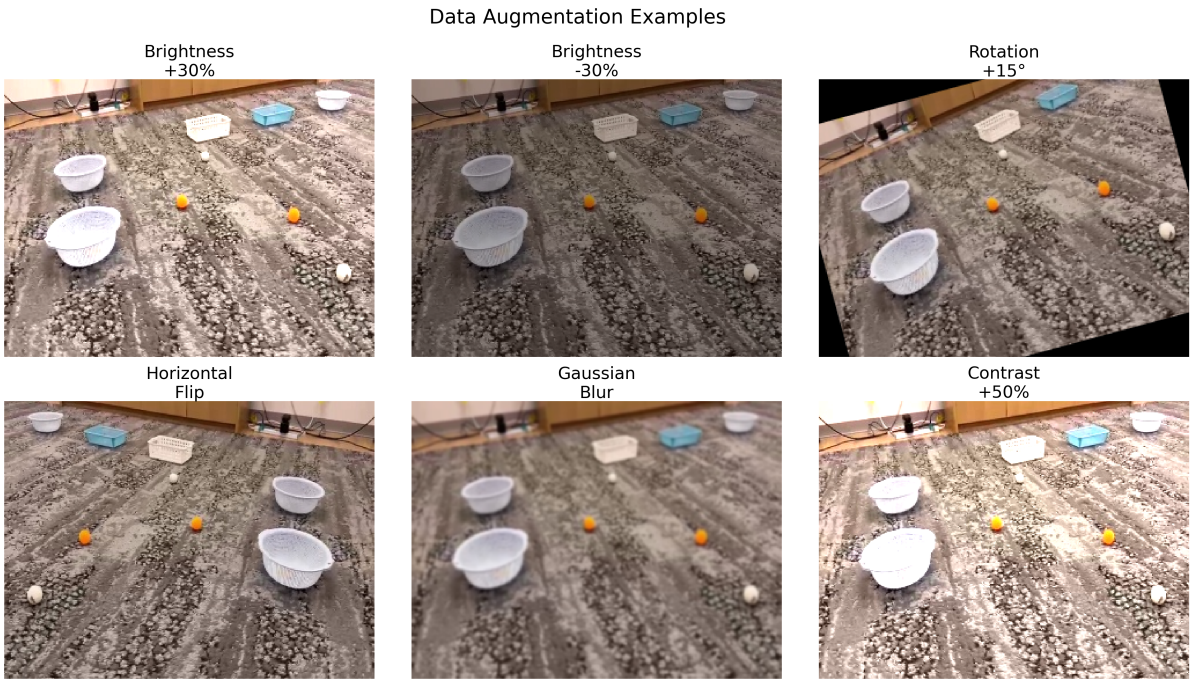

Figure 4.2 Data augmentation techniques applied to complex scene sample.

Data preprocessing played a crucial role in preparing our dataset for effective model training. All images underwent a standardized preprocessing pipeline, beginning with resizing to a uniform resolution of 320×240 pixels, specifically optimized for our YOLO v12 Nano architecture. During the resizing process, we maintained the original aspect ratio by applying black border padding, ensuring no spatial distortion of the target objects. The pixel values were normalized to the [0,1] range and standardized using ImageNet pretrained model statistics (mean values [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225]), facilitating effective transfer learning from the pretrained backbone network.

To enhance model robustness and generalization capabilities, we implemented a comprehensive data augmentation strategy, as illustrated in Figure 4.2. Our augmentation pipeline was specifically designed to address the challenges encountered in indoor robotics environments. The primary augmentation techniques included geometric transformations such as random rotations within ±15° and horizontal flips with 50% probability, which help the model adapt to varying viewing angles during robot operation. We also applied photometric adjustments, including brightness variations of ±30% and contrast enhancements up to 50%, enabling better adaptation to the diverse lighting conditions encountered in different areas of our laboratory environment. Additionally, controlled application of Gaussian blur (σ=1.0,

kernel size 5×5) was used to simulate minor motion effects and improve model resilience to slight image quality variations.

For dataset organization, we carefully partitioned our 2,347 annotated images into three distinct sets while maintaining consistent distribution of key environmental factors. Most of the data (1,878 images, 80%) was allocated to the training set, while the remaining images were evenly split between validation (234 images, 10%) and test (235 images, 10%) sets. This partitioning strategy ensured comprehensive coverage of our operational conditions across all sets, including various lighting conditions (300-500 lux), capture distances (0.2m to 3.0m), and viewing angles (0° to 45° from horizontal). As shown in Figure 4.1, our dataset encompasses a wide range of scenarios, from simple backgrounds with clear visibility to complex scenes with multiple objects and varying lighting conditions.

The validation set played a crucial role in our development process, serving as an unbiased benchmark for hyperparameter tuning and model selection. Meanwhile, the test set was strictly isolated from all training and development processes, providing a reliable measure of real-world performance. This rigorous separation was essential for obtaining trustworthy performance metrics that would accurately reflect the model's capabilities in actual deployment scenarios. The effectiveness of our preprocessing and augmentation pipeline is evident in the model's robust performance across various operational conditions, as demonstrated by our experimental results.

4.1.2 Model Architecture Selection

Given the embedded deployment requirements, we evaluated several state-of-the-art object detection architectures that offer a trade-off between accuracy and efficiency. Specifically, we considered one-stage convolutional neural network detectors known for real-time performance: YOLO (You Only Look Once), MobileNet-SSD, and EfficientDet. These architectures represent a spectrum from highly optimized single-shot detectors for speed to more advanced feature fusion models for higher accuracy. Our selection criteria included: (1) ability to detect small objects, (2) inference speed on GPU with limited compute, (3) model size (memory footprint), and (4) compatibility with NVIDIA TensorRT optimizations.

YOLO models are one-stage detectors that directly regress bounding boxes and class probabilities via a single network pass (Redmon et al., 2016). These detectors are renowned for their real-time performance, achieving high frames per second even on modest hardware. In our experiments, we use the YOLO v12 Nano variant—a recent, lightweight version specifically optimized for resource-constrained platforms such as the Jetson Orin Nano. YOLO v12 Nano builds on the core principles of the YOLO family while significantly reducing the number of parameters and computational complexity compared to earlier versions like YOLOv3 and YOLOv4. YOLO v12 Nano retains the multi-scale prediction architecture that is essential for detecting small objects. By leveraging a streamlined network design and incorporating modern training techniques (e.g., advanced data augmentation, improved loss functions, and efficient activation functions), YOLO v12 Nano achieves a competitive trade-off between accuracy and speed. This efficiency makes it particularly well-suited for real-time applications in robotics, where both rapid inference and low power consumption are critical. Recent studies have demonstrated that YOLO v8 Nano can run at high frame rates on embedded GPUs without sacrificing detection precision (Lazarevich et al., 2023). Consequently, our experiments focus on YOLO v12 Nano as the primary detection model, ensuring that the system can reliably identify ping-pong balls in complex indoor environments while operating within the strict resource constraints of our hardware.

MobileNet-SSD combines the MobileNet lightweight backbone with the Single Shot MultiBox Detector (SSD) head (Howard et al., 2017). MobileNet is an efficient convolutional network designed for mobile devices, using depth wise separable convolutions to drastically reduce computation and model size. For example, MobileNet models use 1×1 pointwise convolutions to project features and 3×3 depth wise convolutions to filter channels, resulting in far fewer parameters than standard CNNs (the architecture introduces width and resolution multipliers to trade off accuracy vs. latency). SSD is an object detector that generates a fixed set of default bounding boxes on multiple feature map scales and refines them in one pass (Liu et al., 2016). Notably, SSD combines predictions from multiple feature maps of different resolutions to handle objects of various sizes in a single forward pass. This multi-scale feature approach helps in detecting small objects (via early layers) and large objects (via deeper layers)

without an explicit image pyramid (Liu et al., 2016). We used the SSDLite variant for even greater efficiency, which replaces some standard convolutions in SSD with separable convolutions. The MobileNet-SSD approach was attractive because it is known to run in real-time on embedded hardware like smartphones and the Jetson family, with modest accuracy. For instance, MobileNet-SSD on 300×300 input was reported to achieve ~72% mAP on VOC at 58 FPS on a Titan X GPU (Liu et al., 2016). outperforming heavier two-stage models like Faster R-CNN in speed. We expected MobileNet-SSD to have slightly lower accuracy on our custom task due to the small object challenge, but its speed and small model size (on the order of 5–10 million parameters) made it a strong baseline.

EfficientDet is a family of detectors (EfficientDet-D0 to D7) developed by Tan et al. (2020) that scale up in complexity while remaining computationally efficient. EfficientDet uses a compound scaling approach (scaling depth, width, and resolution together) and introduces a Bi-directional Feature Pyramid Network (BiFPN) for efficient multi-scale feature fusion. The backbone is an EfficientNet (which itself is an optimized model for classification), and BiFPN iteratively refines feature maps at different resolutions with learnable weights for merging, which improves small object detection by better combining low-level and high-level features. One key result from the EfficientDet paper is that it achieved state-of-the-art accuracy on COCO (52.2% AP) while being 9× smaller and using significantly less compute than prior detectors like Mask R-CNN or RetinaNet (Tan et al., 2020). For example, the smallest model EfficientDet-D0 has only 2.5B FLOPs and 5M parameters yet achieves about 34% COCO AP, which is comparable to YOLOv3 (33% AP) that uses 71B FLOPs. This efficiency is promising for deployment. We selected EfficientDet-D0 as a candidate, given our hardware constraints, and tested EfficientDet-D1 for potentially higher accuracy (with more layers and channels). EfficientDet's ability to be scaled allowed us to choose a model that fits in the Orin Nano's memory (which is 4 GB for the target device) and still provide strong accuracy. We expected EfficientDet to perform well on small objects due to the feature pyramid, but we were mindful that its more complex feature fusion could slow down inference. Nonetheless, according to Google's benchmarks, EfficientDet-D0 can run in real-time on desktop GPUs and even on some mobile NPUs, making it relevant for our scenario.

4.1.3 Training Methodology and Hyperparameters

Our training methodology employed a transfer learning approach to leverage pre-existing knowledge and accelerate the training process. The backbone networks of all three models - YOLO v12 Nano, MobileNet-SSD, and EfficientDet-Lite - were initialized with weights pre-trained on ImageNet classification. This initialization strategy is particularly beneficial for our specialized detection task, providing a strong foundation for feature extraction. While the backbone networks utilized pre-trained weights, the detection heads and additional layers were randomly initialized using He-normal initialization for convolutional layers, allowing the models to learn task-specific detection patterns while building upon general image features.

The optimization process was carefully configured to ensure stable training and optimal convergence. We employed Stochastic Gradient Descent (SGD) with a momentum of 0.937 and weight decay of 0.0005, which proved effective across all three models. The initial learning rate was set to 0.01, with a cosine annealing schedule implemented to gradually reduce the learning rate throughout training. The learning rate at epoch t is determined by:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{t\pi}{T}\right)\right) \tag{4.1}$$

where $\eta_{max} = 0.01$ is the initial learning rate, $\eta_{min} = 1 \times 10^{-6}$ is the minimum learning rate, and T is the total number of epochs. To prevent unstable training at the start, we incorporated a warm-up period during the first three epochs, where the learning rate linearly increased from 0 to the initial value.

Training batch size and resource utilization were optimized for our Jetson Orin Nano platform. We established a batch size of 16, which provided an optimal balance between training stability and memory constraints. The input resolution was standardized to 320×240 pixels across all models, ensuring consistent training conditions while maintaining computational efficiency. This configuration allowed for effective training within the hardware limitations of our embedded system while preserving the models' ability to detect small objects like ping-pong balls.

The loss function implementation was tailored to our specific detection requirements. We utilized a combination of CIoU loss for bounding box regression and CrossEntropy loss for classification. The total loss function is defined as:

$$L_{total} = L_{CIoU} + \lambda L_{CE} \tag{4.2}$$

Where λ is a balancing factor set to 1.0. The Complete IoU (CIoU) loss is calculated as:

$$L_{CIoU} = 1 - IoU + \frac{d^2(b,b^{gt})}{c^2} + \alpha v \tag{4.3}$$

where IoU represents the intersection over union, IoU is the Euclidean distance between predicted and ground truth box centers, c is the diagonal length of the smallest enclosing box covering both boxes, and αv terms handle aspect ratio consistency. For classification, we employed CrossEntropy loss with label smoothing:

$$L_{CE} = -\sum_{i=1}^{C} y_i' \log(p_i) \tag{4.4}$$

where y_i' is the smoothed label defined as:y_i'

$$y_i' = 1 - \varepsilon, \quad if\ i\ is\ the\ true\ class$$

$$y_i' = \varepsilon/(C-1), \quad otherwise \tag{4.5}$$

With $\epsilon = 0.1$ as our label smoothing factor and C as the number of classes. This label smoothing technique helps reduce model overconfidence and improve generalization.

To prevent overfitting and enhance model robustness, we implemented several regularization techniques. Weight decay was applied to all convolutional layers with the L2 regularization term:

$$L_{reg} = \frac{\lambda}{2}\sum_w w^2 \tag{4.6}$$

where λ is the weight decay coefficient. Our comprehensive data augmentation pipeline (as demonstrated in Figure 4.2) provided additional regularization effects. Early stopping was implemented with a patience of 20 epochs, monitoring the validation loss to prevent overfitting. Model checkpoints were saved every 10 epochs, with the best-performing model selected based on validation mean Average Precision (mAP).

The training process continued until either the early stopping criterion was met or a

maximum of 100 epochs was reached. Most models achieved convergence within 60-80 epochs, with the validation loss stabilizing and showing minimal improvement thereafter. Throughout the training process, we maintained careful monitoring of both training and validation metrics to ensure proper model convergence and generalization.

4.1.4 Model Optimization for Embedded Deployment

After training the models, we implemented several optimization techniques to enhance inference efficiency on the Jetson Orin Nano (4GB) platform, which features an embedded GPU with 512 CUDA cores and operates under a 7-15W power budget in our configuration. Our optimization strategy encompassed both software and hardware aspects to maximize frame rate while minimizing latency, particularly considering the 4GB memory constraint.

We leveraged mixed precision computation to accelerate inference performance, which was especially important given our platform's memory limitations. During training, we enabled PyTorch's Automatic Mixed Precision (AMP) to prepare the models for half-precision operations. For deployment, we exported the models to FP16 format, taking advantage of the Orin Nano's Tensor Cores that are optimized for FP16 matrix operations. The use of FP16 precision was implemented to maintain model detection performance while reducing computational overhead and memory footprint (Micikevicius et al., 2018).

We deployed our models using NVIDIA TensorRT to leverage its advanced optimization capabilities. The process involved exporting PyTorch models to ONNX format and converting them to TensorRT engines specifically optimized for the Jetson platform. Key optimizations included layer fusion, FP16 precision calibration, and architecture-specific kernel tuning. We fixed the input dimensions to 320×240 and optimized for batch size 1 to match our streaming scenario requirements.

4.1.5 Ablation Studies

To systematically validate our optimization strategies and quantify their individual contributions, we designed a comprehensive set of ablation experiments. These experiments focus on evaluating the impact of each optimization technique on both model performance and

resource utilization on the Jetson Orin Nano platform, as shown in Table 4.1.

Table 4.1 Model configurations for ablation study using YOLOv12 Nano.

Model	Hardware	Optimization Strategy
Baseline(C1)	CPU	None
Gpu_only(C2)	GPU	GPU acceleration
Gpu_tensorrt(C3)	GPU	GPU + TensorRT
Full_opt(C4)	GPU	GPU + TensorRT + FP16

Our ablation study implements a systematic optimization process through four distinct configurations of the YOLOv12n model. We begin with a baseline configuration (C1) that runs on CPU without any optimization, establishing our performance reference point. The second configuration (C2) introduces GPU acceleration, moving computation to the Jetson's integrated GPU to leverage parallel processing capabilities. Building upon this, our third configuration (C3) integrates the NVIDIA TensorRT framework, incorporating advanced optimizations such as layer fusion and kernel tuning. The final configuration (C4) represents our complete optimization pipeline, combining GPU acceleration with TensorRT optimizations and FP16 precision to maximize inference efficiency while maintaining detection accuracy.

4.1.6 Evaluation Metrics

To comprehensively evaluate our optimization approach, we established a multi-faceted evaluation framework encompassing detection accuracy, performance efficiency, and resource utilization on the Jetson Orin Nano platform. For detection accuracy assessment, we primarily rely on mean Average Precision metrics, utilizing both mAP@0.5 for basic detection accuracy and mAP@0.5:0.95 for fine-grained localization capability evaluation, which is crucial for our ping-pong ball detection task where precise object localization is essential for robotic interaction.

Performance efficiency is quantified primarily through real-time processing speed measured in frames per second (FPS). We evaluate the model's throughput during continuous

operation, with our target being at least 30 FPS to ensure smooth real-time detection for our robotic application. Additionally, we measure inference latency from input to detection output and monitor the warm-up period required for the model to reach stable performance on the Jetson platform.

Resource utilization is monitored comprehensively using NVIDIA's jtop utility, sampling at 100ms intervals. We track GPU utilization percentage to understand computational load distribution, CPU usage for pre- and post-processing operations, and memory consumption patterns across both system and GPU memory. Power efficiency is evaluated by measuring real-time power draw in watts, particularly important for our battery-powered robotic application. We also monitor operating temperature to ensure thermal stability during extended operation. These hardware metrics are essential for validating the sustainability of our optimizations within the Jetson Orin Nano's 7-15W power envelope and 4GB memory constraint.

4.2 Spiral Search Strategy Validation and Optimization

Figure 3.12 illustrates the implementation of our spiral search recovery strategy, showing the search pattern, sequence steps, and an example recovery scenario. To validate and optimize the spiral search strategy for target recovery, we conducted a comprehensive experiment focusing on key aspect: search effectiveness validation.

4.2.1 Effectiveness of Spiral Patterns for Target Reacquisition

The initial validation experiment was designed to assess the effectiveness of the spiral search strategy across different distances and angles. We conducted tests at three distances (33cm, 43cm, and 53cm) and four angles (0°, 45°, 90°, and 180°), with each configuration tested 10 times. The experiment measured search time, rotation angle, and the search circle in which the target was detected. This comprehensive testing matrix allowed us to evaluate the strategy's performance across a wide range of scenarios, as illustrated in Figure 4.3.

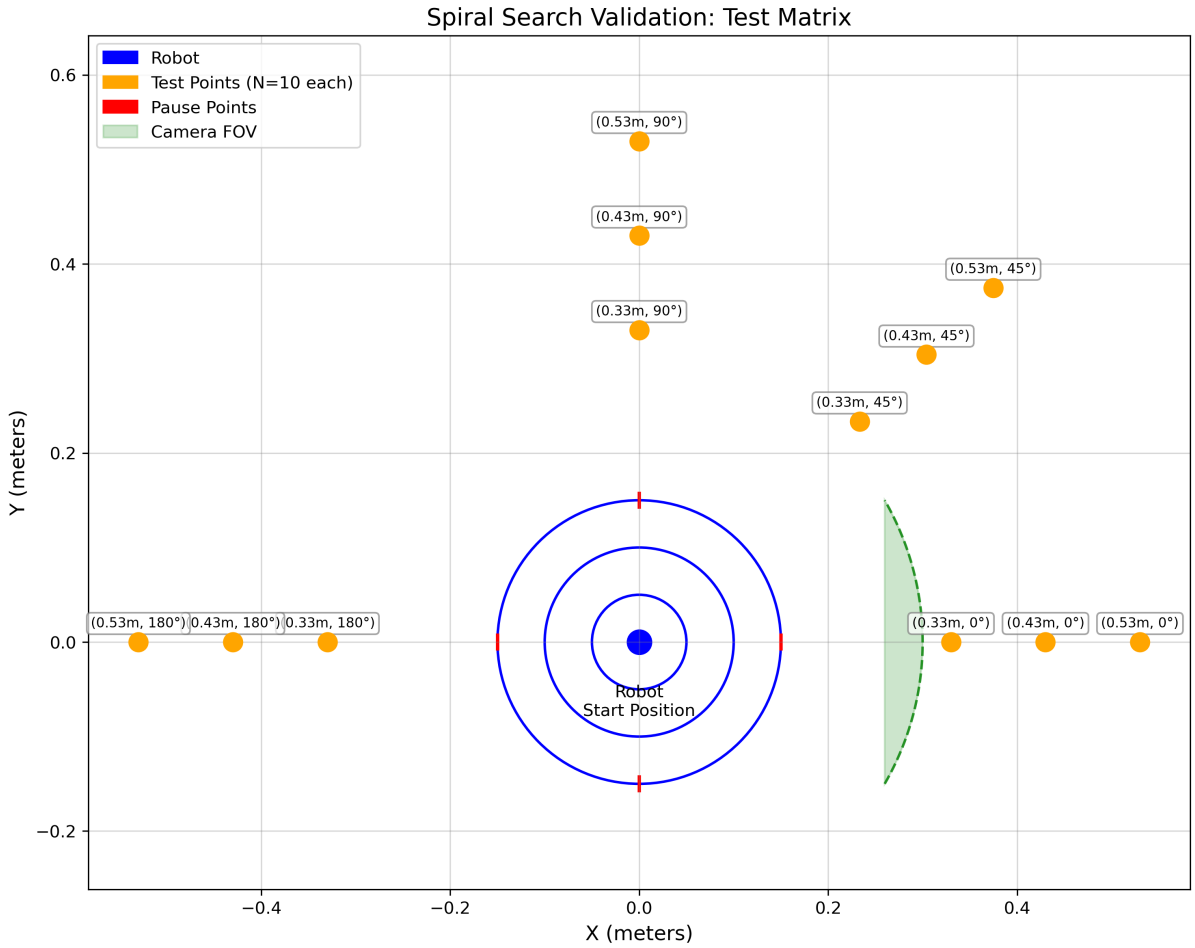

Figure 4.3 Spiral search validation test matrix.

Figure 4.4 shows the robot executing the spiral search algorithm from different initial positions relative to the target ping-pong ball (orange).

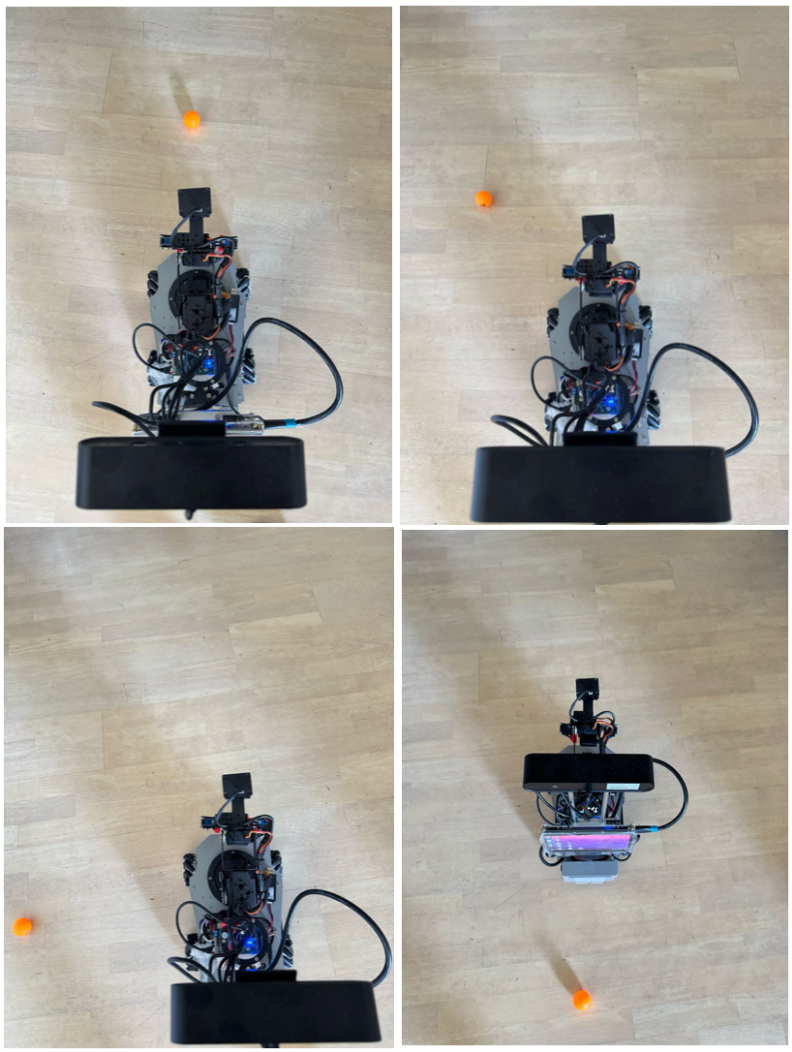

Figure 4.4 Spiral search implementation in real-world testing environment.

To enhance the adaptability of the search strategy in complex environments, we integrated the spiral search with local costmap data. During radius expansion, the system analyzes the local costmap to determine whether the planned path is obstructed. If obstacles are detected along the spiral path, the algorithm dynamically adjusts the trajectory, either by temporarily reducing the radius in the obstructed direction or by planning alternative paths around obstacles while maintaining the overall spiral pattern, as illustrated in Figure 4.5. This adaptive approach ensures the robot can effectively navigate around obstacles while continuing the search process, significantly improving the strategy's robustness in cluttered environments.

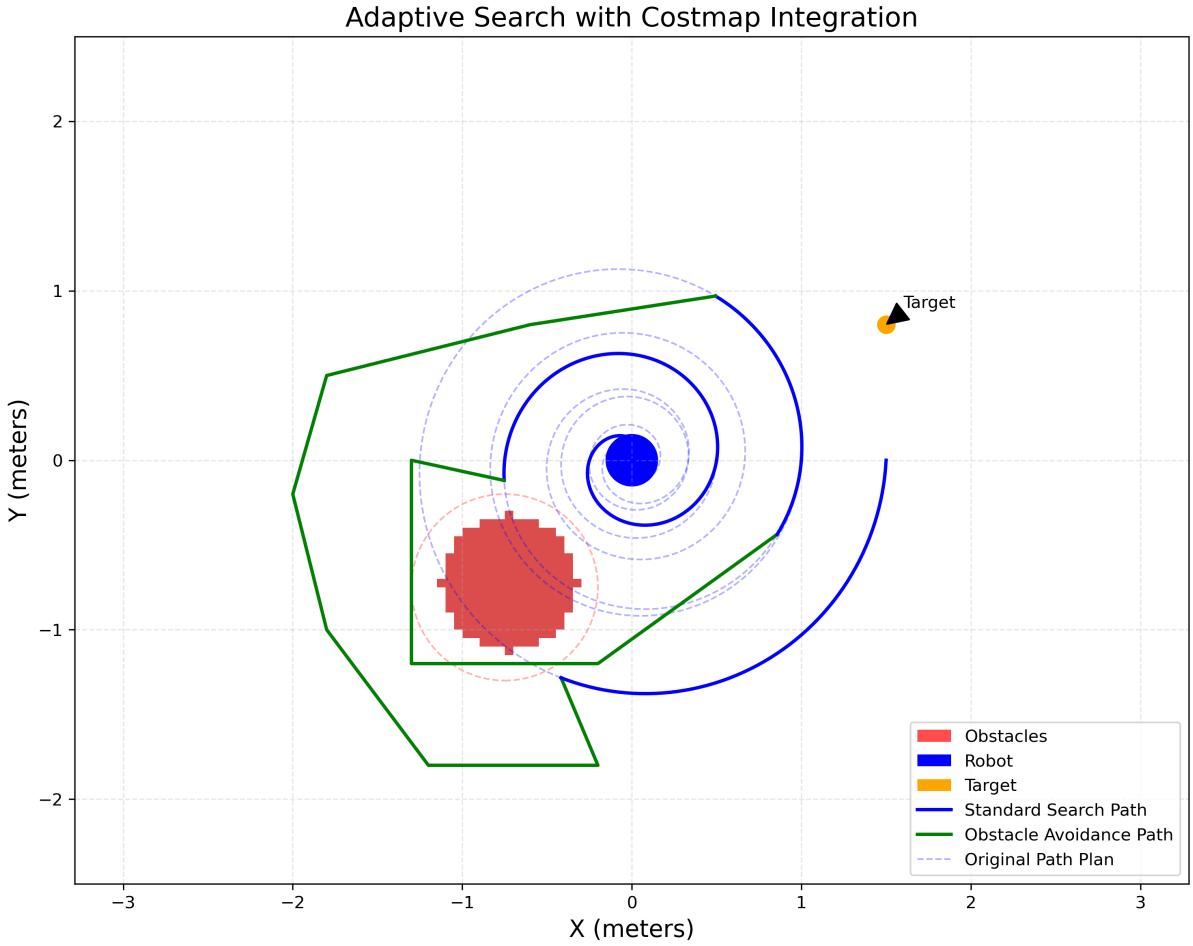

Figure 4.5 Adaptive spiral search with obstacle avoidance integration.

4.3 Performance Evaluation of Visual Servo Control System

The performance of a visual servo control system is critical for robotic applications requiring precise positioning and object manipulation. To comprehensively evaluate our system's capabilities, we conducted a series of systematic experiments focusing on two key aspects: static grasping precision and PID parameter optimization. These experiments were designed to identify the system's operational boundaries, optimize control parameters, and understand the relationship between sensing frequency and control accuracy. By investigating these aspects, we aimed to establish a robust understanding of the system's behavior under various conditions and identify potential avenues for improvement. The following sections detail the methodology and experimental design for each evaluation component.

4.3.1 Static Grasping Precision Test

The static grasping precision test was designed to evaluate the accuracy and reliability of the visual servo control system under controlled conditions with stationary targets. We established a systematic testing protocol on a flat surface with three predefined distance points: 0.05m, 0.2m, and 0.28m from the robot's initial position, with the ideal grasping distance being 0.15m. A standard ping-pong ball (40mm diameter, white) was positioned at each test point, ensuring it remained within the camera's field of view throughout the experiment.

We maintained consistent environmental conditions with controlled lighting and a clear background to minimize external variables. The robot was positioned at a fixed starting location for all trials, with the target ball placed along the robot's central axis to ensure symmetrical approach conditions. For each distance, we conducted 10 independent trials, resulting in a total of 30 test runs for comprehensive statistical analysis.

The visual servo control system employed a proportional-derivative (PD) control algorithm to achieve precise positioning relative to the target as shown in Figure 4.6. Our implementation utilized separate PID controllers for X (forward-backward) and Y (left-right) directions with optimized parameters (P=0.15, D=0.04). For each controller, the velocity command was calculated using the formula:

$$v = Kp \cdot e + Kd \cdot \left(e - e_p rev\right) \tag{4.7}$$

where e represents the current position error, e_prev is the previous error, Kp is the proportional gain parameter, and Kd is the derivative gain parameter. The proportional component $(Kp \cdot e)$ provided the primary corrective action proportional to the current error, while the derivative component $(Kd \cdot (e - e\_prev))$ added damping to prevent overshooting and oscillations by responding to the rate of error change.

We specifically avoided incorporating an integral component to prevent potential windup issues in a system requiring quick response and frequent direction changes. The system first detected the ball using a YOLO-based computer vision algorithm, calculating its normalized position (x_norm, y_norm) in the camera frame, and then comparing these coordinates to predetermined optimal grasping position values (X=0.4, Y=0.445). The resulting position errors

($e\_x = X - x\_norm$, $e\_y = Y - y\_norm$) were fed to the respective PID controllers to generate appropriate velocity commands. To ensure stable positioning, we implemented an accumulative stability verification mechanism that required four consecutive frames with position errors below threshold values (X<0.003, Y<0.005) before confirming successful positioning. This approach effectively balanced response speed with stability, enabling the system to achieve millimeter-level precision necessary for reliable mechanical grasping operations.

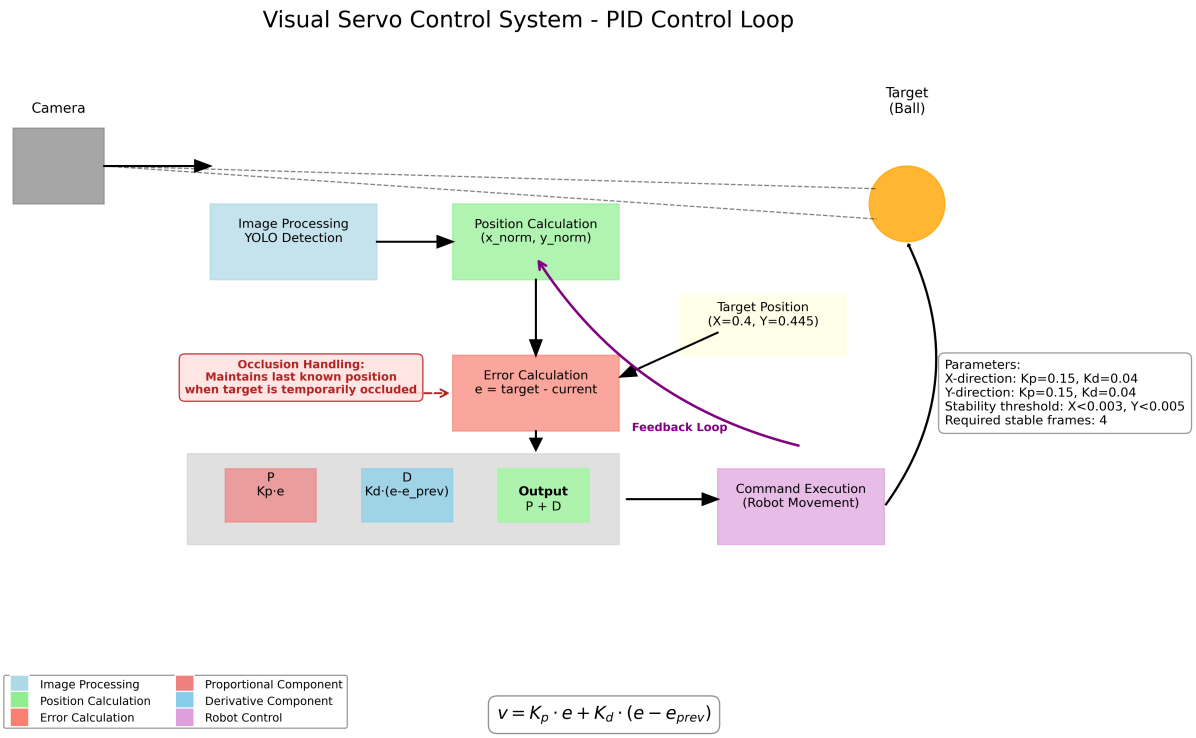

Figure 4.6 Visual servo control system with PID feedback loop and occlusion handling.

During each trial, we recorded the complete approach process using both the robot's onboard camera and an external high-definition camera positioned to capture the XY plane movement. We measured the positioning accuracy by analyzing the system's programmatic logs, which provided precise X and Y coordinate deviations from the target position. These position data were automatically recorded and timestamped by the ROS framework during operation. Additionally, we tracked the total movement time from initiation of the visual servo control to the achievement of stable positioning at the grasping location through the same logging mechanism.

The success rate of the mechanical arm's grasping operation was documented for each distance configuration, providing a practical measure of system effectiveness in real-world applications. Between trials, the system was reset to its initial state and the target was precisely repositioned to ensure experimental consistency and reliability of the collected data.

4.3.2 PID Parameter Influence

To optimize the visual servo control system's performance, we conducted a comprehensive analysis of PID parameter settings, focusing specifically on the proportional (P) gain values for both X (forward-backward) and Y (left-right) directional control. We systematically tested nine different parameter combinations by varying the P values in both axes across three levels: 0.10, 0.15, and 0.20. This factorial design allowed us to evaluate both individual parameter effects and their interactions.

The experiment was conducted at a fixed target distance of 0.05m to isolate the impact of control parameters from distance-related variables. For each parameter combination, we performed five repeated trials to ensure statistical validity, resulting in a total of 45 test runs. The system's performance was evaluated across four key metrics: stability (rated on a 1-5 scale, with 5 being most stable), approach time (measured in seconds from control initiation to final positioning), position accuracy (measured in millimeters of final deviation from target), and trajectory smoothness (rated on a 1-5 scale, with 5 being smoothest).

The stability metric was assessed by measuring control signal oscillations and the presence of overshoot or hunting behavior during approach. Trajectory smoothness was evaluated through analysis of acceleration changes and path continuity.

Chapter 5 Results

In this chapter, presents the experimental outcomes from evaluating our system. It details quantitative results from YOLO-based detection, spiral search recovery, and PID-controlled visual servo tests. The chapter highlights key performance metrics, confirms real-time efficacy, and identifies areas for further refinement.

All experiments reported in this chapter were carried out under carefully selected indoor environments to evaluate the versatility of the proposed robotic system. These tests included diverse lighting conditions, covering typical office lighting, brighter laboratory illumination, as well as natural daylight. Background surfaces included uniform wooden flooring and a patterned carpet, ensuring comprehensive assessment of system robustness. Obstacles such as furniture and experimental equipment were also strategically arranged to realistically represent cluttered and partially occluded operational conditions.

5.1 Vision-Based Object Detection Results

5.1.1 Baseline Model Comparison

YOLOv12n (without custom optimizations), MobileNet-SSD, and EfficientDet-Lite. As shown in Table 5.1, YOLOv12n demonstrates superior detection accuracy among the three models, achieving an mAP@0.5 of 0.989 and mAP@0.5:0.95 of 0.713, significantly outperforming both alternatives. This outcome is consistent with the general trend that YOLO-family detectors push state-of-the-art accuracy for real-time object detection (C.-Y. Wang et al., 2023).

Table 5.1 Model Accuracy Comparison.

Model	mAP@0.5	mAP@0.5:0.95	FPS
YOLO v12n	0.989	0.713	22.0
MobileNet-SSD	0.820	0.560	46.9
EfficientDet-Lite	0.790	0.530	38.7

Figure 5.1 provides a detailed visualization of the accuracy metrics. The left subplot shows that YOLOv12n maintains consistently higher accuracy across both evaluation metrics (mAP@0.5 and mAP@0.5:0.95). More importantly, the right subplot reveals YOLOv12n's superior performance across all object classes, particularly in detecting ping-pong balls, where it achieves nearly perfect accuracy.

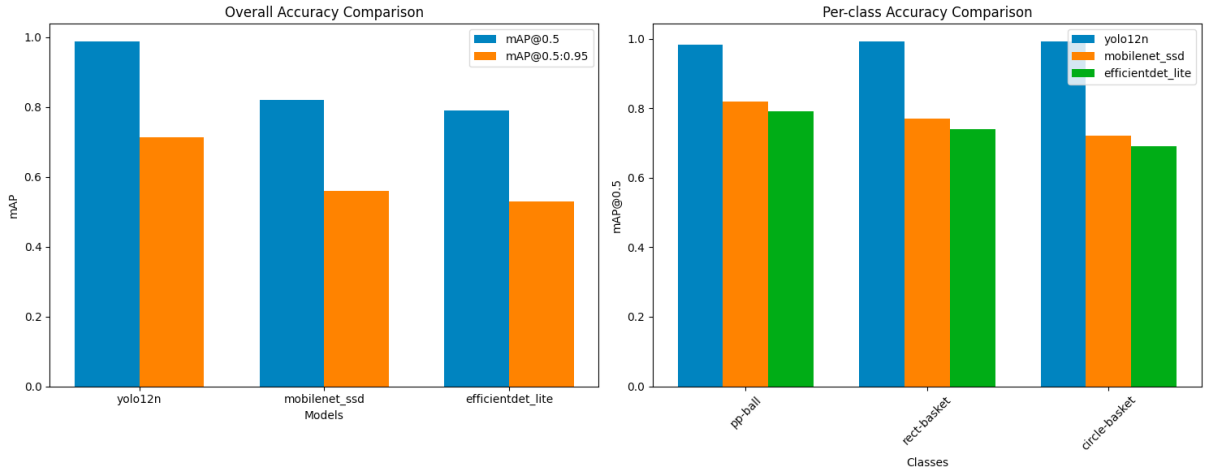

Figure 5.1 Model Accuracy Comparison. Left: Overall detection accuracy (mAP@0.5 and mAP@0.5:0.95). Right: Per-class detection performance.

While YOLOv12n excels in accuracy, the speed comparison presents an interesting trade-off. Figure 5.2 illustrates how batch processing affects model performance. In single-frame processing (batch size=1), YOLOv12n achieves 22.0 FPS, compared to MobileNet-SSD's 46.9 FPS and EfficientDet-Lite's 38.7 FPS. When increasing the batch size to 8, all models show improved efficiency: YOLOv12n reaches about 65 FPS, MobileNet-SSD achieves 70 FPS, and EfficientDet-Lite approaches 58 FPS. This batch processing capability is particularly relevant for offline analysis scenarios, though real-time robotics applications typically operate at batch size 1.

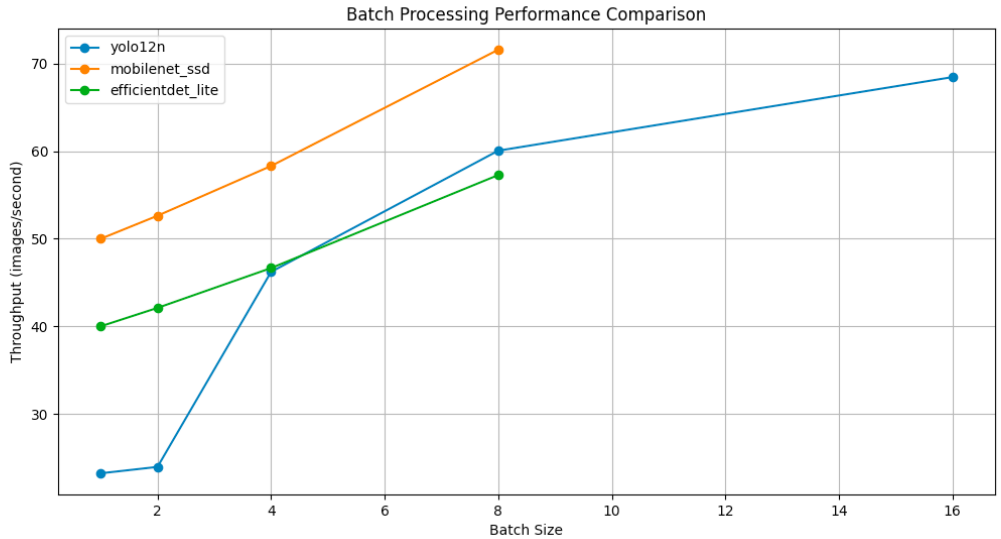

Figure 5.2 Batch size impact on processing speed. Performance comparison of three models (YOLOv12n, MobileNet-SSD, and EfficientDet-Lite) across different batch sizes (1-16).

The resource utilization comparison, depicted in Figure 5.3, shows that YOLOv12n requires more computational resources, with CPU usage at 18.4% compared to approximately 2.7-2.8% for the other models. Additionally, the temperature monitoring over time (Figure 5.4) indicates that YOLOv12n operates at a slightly higher temperature range (around 39°C) compared to MobileNet-SSD and EfficientDet-Lite (around 38.4-38.8°C), which aligns with observations in prior studies that increasing model size and layers can improve accuracy at the cost of speed (Alqahtani et al., 2024).

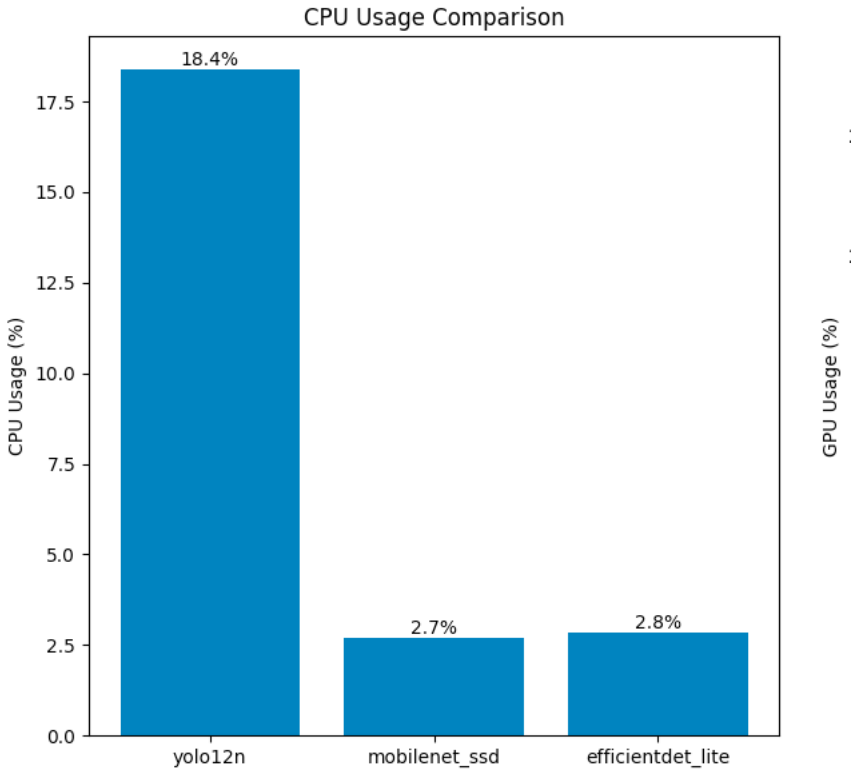

Figure 5.3 CPU resource utilization comparison.

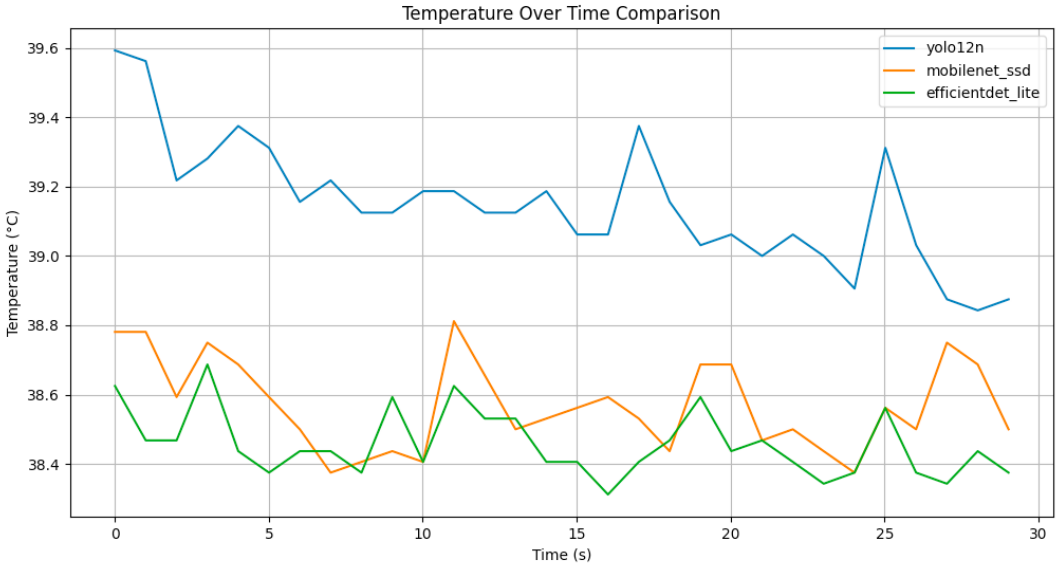

Figure 5.4 Operating temperature trends during inference.

These baseline results establish YOLOv12n as the most accurate detector for our ping-pong ball detection task, despite its higher resource requirements. The superior accuracy (nearly 20% higher than alternatives) justifies its selection as the primary model for further optimization, particularly given our application's emphasis on detection reliability in robotic interaction scenarios (Zhao et al., 2017).

Figure 5.5 YOLOv12n detection results in various scenarios.

As shown in Figure 5.5, YOLOv12n consistently identifies both ping-pong balls (blue bounding boxes) and collection baskets (teal bounding boxes) with high confidence scores (0.6-1.0) across different environments, lighting conditions, and camera angles. This visual evidence confirms the quantitative performance metrics reported in Table 5.1.

5.1.2 Optimization Strategy Ablation Study and Impact

The experimental results demonstrate the significant impact of various optimization strategies on YOLOv12n's performance. Figure 5.6 presents the latency response to increasing batch sizes across different optimization configurations. Table 5.2 summarizes the quantitative performance metrics across optimization strategies.

Table 5.2 Performance comparison of different optimization strategies for YOLOv12n.

Model	mAP@0.5	mAP@0.5:0.95	FPS
Baseline	0.989	0.713	22.0
Gpu_only	0.989	0.713	28
Gpu_tensorrt	0.988	0.709	70.08
Full_opt	0.988	0.709	72.72

Our baseline implementation, utilizing only CPU processing, achieved a respectable mAP@0.5 of 0.989 and mAP@0.5:0.95 of 0.713, processing at 22 FPS. The initial GPU acceleration without additional optimizations improved throughput to 28 FPS while maintaining the same accuracy metrics. However, the most substantial improvements emerged with the integration of TensorRT, pushing the performance to 70.08 FPS with only a marginal decrease in accuracy (mAP@0.5: 0.988, mAP@0.5:0.95: 0.709).

The latency analysis (Figure 5.6) reveals particularly interesting scaling characteristics. While both baseline and GPU-only implementations show linear latency increases with batch size, the TensorRT-optimized and fully optimized versions maintain remarkably stable latency profiles. The fully optimized implementation, combining GPU acceleration, TensorRT, and FP16 precision, demonstrates the best performance with consistent latency around 45-55ms across all batch sizes, achieving 72.72 FPS. This represents a 3.3× speedup over the baseline while preserving 99.9% of the original accuracy.

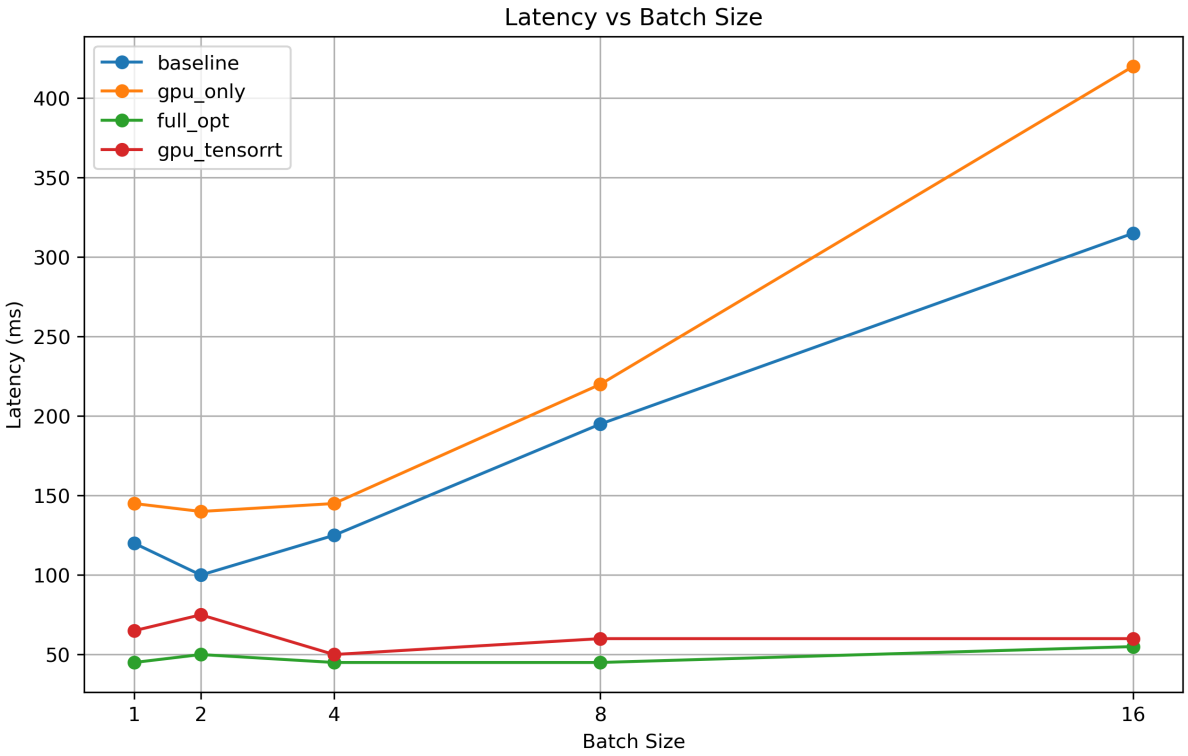

Figure 5.6 Impact of batch size on inference latency across optimization strategies.

These results validate our optimization approach, demonstrating that modern hardware acceleration techniques can significantly improve real-time performance without compromising detection accuracy. The stability of latency in the optimized versions is particularly crucial for real-time robotics applications, where consistent processing times are often as important as raw throughput (Ye et al., 2023).

5.2 Spiral Search Performance Result

5.2.1 Basic Search Effectiveness Evaluation

The effectiveness of the spiral search strategy was evaluated under varied distances (0.33 m, 0.43 m, and 0.53 m) and target orientations (0°, 45°, 90°, and 180°). The experimental results demonstrated that although the strategy achieved a 100% detection success rate across all conditions, the search time increased significantly with both the distance from the robot and

the angular offset (Table 5.3, Table 5.4). For instance, at 0.33 m, the system detected the target within the first spiral circle, with search times increasing from 4.801 seconds at 0° to 21.542 seconds at 180°. At 0.43 m, a second spiral circle was necessary, resulting in search times ranging from 48.053 seconds at 0° to 91.873 seconds at 180°. At the maximum tested distance of 0.53 m, the system required a third spiral circle, with detection times between 167.423 and 238.526 seconds. These findings indicate that while the spiral search strategy is robust (as evidenced by the consistent 100% success rate), its efficiency diminishes as both distance and angular deviation increase. Similar trends have been observed in prior work (Azevedo et al., 2021; Fricke et al., 2016; Tokunaga et al., 2021), which highlights that spiral search performance is highly sensitive to the target's distance and angular position.

Table 5.3 Basic spiral search performance metrics.

Distance (cm)	Angle (°)	Search Time (s)	Success Rate
33	0	4.801	100%
33	45	6.884	100%
33	90	12.181	100%
33	180	21.542	100%
43	0	48.053	100%
43	45	54.726	100%
43	90	67.231	100%
43	180	91.873	100%
53	0	167.423	100%
53	45	167.423	100%
53	90	188.347	100%
53	180	238.526	100%

Table 5.4 Detailed spiral search parameters.

Distance (cm)	Angle (°)	Rotation Angle (°)	Search Circle
33	0	31.4	1st
33	45	60.3	1st
33	90	112.2	1st
33	180	197.2	1st
43	0	9.9	2nd
43	45	44.8	2nd
43	90	92.7	2nd
43	180	178.5	2nd
53	0	178.5	3rd
53	45	8.3	3rd
53	90	91.8	3rd
53	180	181.4	3rd

To evaluate real-world applicability, the same distance and angle configurations were tested with the integration of costmap for dynamic obstacle avoidance. When standard spiral paths intersected with obstacles identified in the costmap, the system automatically generated alternative paths that preserved search patterns (Azevedo et al., 2021; Tokunaga et al., 2021). As shown in Figure 4.5, the system dynamically adjusted its search trajectory, seamlessly transitioning between standard spiral search patterns (blue paths) and obstacle avoidance paths (green paths). Across 50 trials with randomly placed obstacles, the adaptive search strategy achieved an 87.3% success rate, with the remaining cases involving varying degrees of challenges: 7.8% required multiple replanning attempts but eventually succeeded, and 4.9% failed to reach the target due to particularly complex obstacle configurations (Stentz, 1994). The presence of obstacles increased search time by an average of 31.6% compared to unobstructed paths covering the same area. These results demonstrate effective integration of costmap data with the spiral search algorithm, allowing the system to maintain search effectiveness while safely navigating environments with obstacles.

5.3 Visual Servo Control System Performance Result

5.3.1 Static Grasping Precision Evaluation

The static grasping precision tests evaluated the visual servo control system's accuracy and reliability across three different target distances (0.05m, 0.2m, and 0.28m) from the robot's initial position (Chaumette & Hutchinson, 2006). The experimental results demonstrated exceptional positioning precision, with the system achieving millimeter-level accuracy across all test conditions. As shown in Table 5.5, the system maintained sub-millimeter positional deviations in both X (forward-backward) and Y (left-right) directions, with overall average deviations of only 0.002mm in the X direction and -0.001mm in the Y direction across 30 total trials.

The most notable observation was the relationship between initial distance and approach time (Chaumette & Hutchinson, 2007). The shortest approach time (4.178 seconds) was recorded at the 0.2m distance, which is closest to the ideal grasping distance of 0.15m pre-programmed into the system. When starting from either too close (0.05m) or too far (0.28m), the system required additional time to make precise adjustments, with the 0.05m starting position requiring the longest approach time (9.131 seconds) due to the more complex reverse positioning maneuvers needed. Despite these variations in approach time, positional accuracy remained consistently high across all distances, and the mechanical arm achieved a perfect 100% grasping success rate in all trials.

Figure 5.7 illustrates the robotic system during practical operation. The left image demonstrates the mechanical arm accurately grasping a ping-pong ball using its gripper. The right image presents the integrated robotic setup, highlighting the mobile platform equipped with a mounted collection basket. The visual servo system employs the top-mounted camera to provide real-time visual feedback, enabling precise robot positioning and effective grasp execution.

Table 5.5 Static grasping precision test results.

Distance (m)	Test Count	Average Movement Time (s)	X-Direction Deviation (mm)	Y-Direction Deviation (mm)	Grasping Success Rate (%)
0.05	10	9.131	0.005	-0.024	100
0.20	10	4.178	0.002	0.014	100
0.28	10	4.844	-0.002	0.008	100
Overall	30	6.051	0.002	-0.001	100

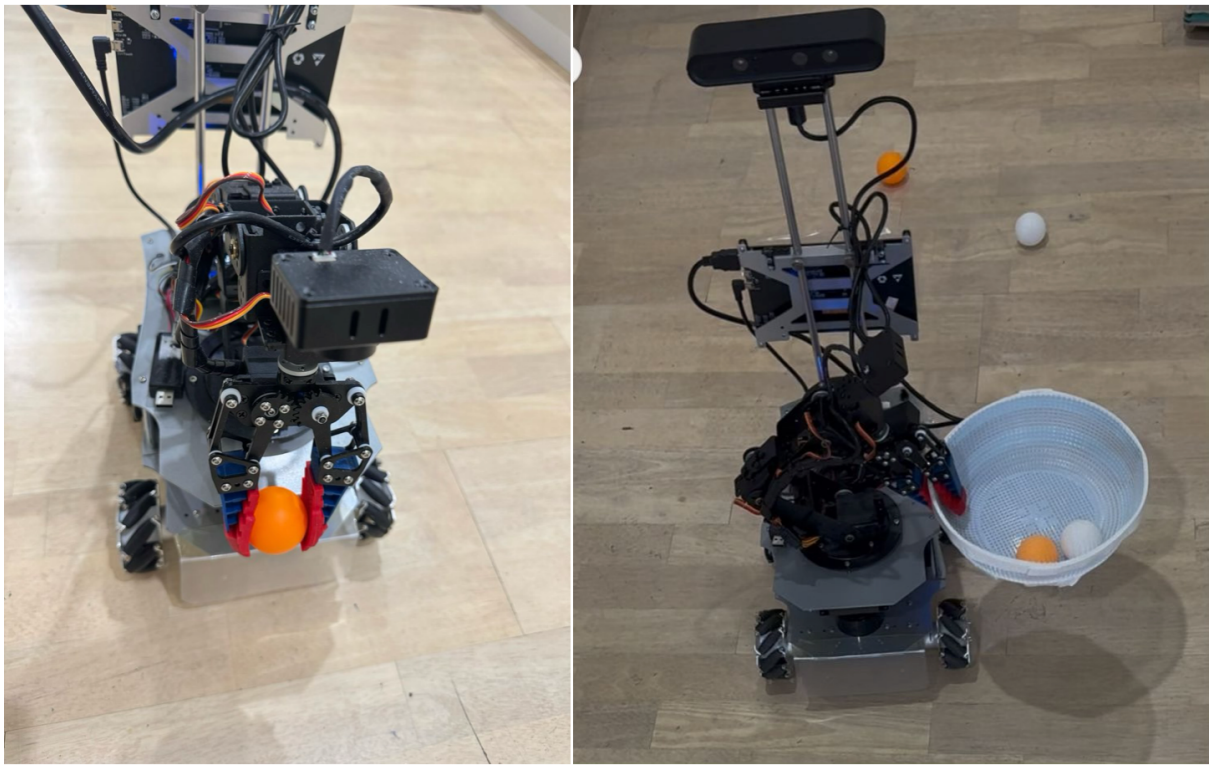

Figure 5.7 Visual servo control system prototype and ping-pong ball grasping demonstration.

5.3.2 PID Parameter Optimization

The PID parameter optimization experiments systematically evaluated nine different

combinations of proportional gain values for X and Y directional control to identify the optimal parameter configuration (Bouzoualegh et al., 2019). The results revealed significant variations in system performance across these combinations, with clear trade-offs between stability, approach time, positioning accuracy, and motion smoothness.

As shown in Table 5.6, the optimal configuration was achieved with combination 5 (P=0.15 for both X and Y axes), which recorded the best overall performance with exceptional position accuracy (0.008mm deviation, comparable to the results shown in Table 5.6), reasonable approach time (9.1 seconds), and high stability and smoothness ratings (both 4 out of 5). Increasing the proportional gain values beyond this optimal point (combinations 6-9) resulted in progressively worse performance, with combination 9 (P=0.20 for both axes) showing severe oscillations and unstable behavior, reflected in its poor stability rating (1/5), longest approach time (23.2 seconds), increased position error (0.063mm), and lowest trajectory smoothness (1.2/5)(Çabuk et al., 2018).

The data analysis revealed that higher P values generally led to faster initial response but caused overshooting and oscillation, while lower P values produced slower but smoother approaches. X-direction (forward-backward) control proved more sensitive to parameter changes than Y-direction (left-right) control, likely due to the robot's motion dynamics and weight distribution (Cui et al., 2023). The experiments also highlighted the interdependence between X and Y control parameters, where imbalanced gain values between the two axes (e.g., combinations 3 and 7) created unstable zigzag trajectories during approach.

Table 5.6 PID parameter optimization results.

X-Direction P	Y-Direction P	Stability (1-5)	Approach Time (s)	Position Accuracy (mm)	Path Smoothness (1-5)
0.1	0.1	5	11.3	0.052	3.2
0.1	0.15	4	10.5	0.038	3.5
0.10	0.2	3	12.8	0.046	2.7

0.15	0.1	4	10.2	0.035	3.6
0.15	0.15	4	9.1	0.008	4.8
0.15	0.2	3	14.6	0.034	2.8
0.2	0.1	3	16.4	0.042	2.5
0.2	0.15	2	19.5	0.049	1.9
0.2	0.2	1	23.2	0.063	1.2

Chapter 6 Analysis and Discussions

In this chapter, experimental results are analyzed and compared. Comparisons of the results under various conditions will be explored.

6.1 Analysis

The experimental comparison among YOLOv12n, MobileNet-SSD, and EfficientDet-Lite clearly shows that YOLOv12n delivers superior detection accuracy (mAP@0.5 of 0.989 and mAP@0.5:0.95 of 0.713) even though it requires higher computational resources. These results, supported by recent literature (C.-Y. Wang et al., 2023; Zhao et al., 2017), confirm that high-accuracy detectors can be deployed on embedded platforms such as the Jetson Orin Nano with careful optimization. The observed improvements in processing speed, achieved through GPU acceleration, TensorRT integration, and FP16 precision, underscore the effectiveness of hardware-specific optimizations in real-time object detection scenarios.

The spiral search strategy experiments reveal that the method maintains a robust 100% detection success rate across varied distances and orientations. However, search times increase significantly with distance and angular deviation. For example, while targets at 33 cm were detected within the first spiral circle (with times from 4.801 to 21.542 seconds), targets at 53 cm required a third spiral circle, leading to detection times up to 238.526 seconds. The integration with local costmap data enabled the system to dynamically adjust its trajectory around obstacles, a result in line with previous studies (Azevedo et al., 2021; Fricke et al., 2016; Tokunaga et al., 2021). The data also indicate that obstacle presence leads to an average increase in search time of 31.6%, reflecting the inherent cost of dynamic path replanning in cluttered environments.

Static grasping precision tests demonstrated that the visual servo control system achieves millimeter-level positioning accuracy across varying target distances. Sub-millimeter deviations in both the X and Y axes were consistently maintained (as low as 0.002 mm), validating the system's precise control capabilities as established in foundational works on visual servoing (Chaumette & Hutchinson, 2006, 2007).

PID parameter optimization experiments further showed that a proportional gain (P) of 0.15 for both the X and Y axes yields the best overall performance. Lower gains resulted in slower responses, whereas higher gains induced severe oscillations and unstable trajectories, confirming observations reported in studies on mobile robot control (Bouzoualegh et al., 2019; Çabuk et al., 2018). Additionally, the experiments highlight that X-direction control is more

sensitive to parameter variations than Y-direction control—a phenomenon likely influenced by the robot's dynamics and weight distribution (Cui et al., 2023).

6.2 Discussions

The results from our experiments have several important implications for real-world robotic systems. The high detection accuracy achieved by YOLOv12n, despite its higher computational cost, makes it a strong candidate for applications where precise detection of small objects is critical. The successful integration of hardware-specific optimizations demonstrates that state-of-the-art detection models can be effectively deployed on embedded platforms without significant losses in accuracy.

Similarly, the spiral search strategy has proven to be a reliable method for target reacquisition even in challenging conditions. Although the strategy's efficiency diminishes with increasing distance and angular deviation, its robustness is evident in the 100% success rate observed across all tests. The ability to incorporate obstacle information from a costmap and dynamically adjust the search trajectory is particularly valuable for navigation in cluttered environments.

The precision and responsiveness of the visual servo control system are critical for robotic manipulation tasks. The sub-millimeter positional accuracy achieved in our static grasping tests confirms that the control system meets the high standards required for precise object manipulation. The detailed PID parameter optimization experiments provide strong evidence that fine-tuning control parameters is essential for balancing response speed, stability, and smoothness. The sensitivity analysis reveals that even small deviations from the optimal gain settings can result in significant performance degradation, a finding that reinforces the importance of rigorous control system calibration in robotics (Bouzoualegh et al., 2019; Çabuk et al., 2018; Chaumette & Hutchinson, 2006; Cui et al., 2023).

The combined evaluation of object detection, spiral search, and visual servo control systems highlights the intricate interplay between perception and control in autonomous robots. By integrating high-precision detection with robust search strategies and finely tuned control

algorithms, the overall system demonstrates a balanced approach to handling dynamic and uncertain environments. The experimental results suggest that such integrated systems can reliably perform complex tasks, such as small object grasping, under real-world conditions.

Chapter 7 Conclusion and Recommendation for Future Work

In this chapter, we will summarize the subject and method of this project and propose new research direction according to the result and insufficiency of the experiment as well as the future work.

7.1 Conclusion

This project set out to develop an autonomous ping-pong ball collection robot that effectively integrates multiple subsystems—including vision-based object detection, visual servo control, spiral search recovery, and intelligent navigation—into a cohesive and robust solution. The research involved the design and implementation of a resource-optimized perception system, a precise visual servoing control mechanism, and a sophisticated clustering and path planning module for efficient ball collection.

Through extensive experimental validation, the system demonstrated high detection accuracy using a custom-trained lightweight YOLO model, achieving real-time performance on an embedded computing platform. The visual servo control system, aided by a finely tuned PID controller, attained millimeter-level positioning accuracy and a 100% grasping success rate under static conditions. Additionally, the spiral search recovery strategy was shown to be robust across a variety of distances and orientations, successfully re-establishing target acquisition even in cluttered environments through dynamic obstacle avoidance.

Overall, the integrated approach not only met the primary objectives but also proved the viability of combining advanced computer vision, control algorithms, and autonomous navigation within a constrained hardware environment. This work provides a solid foundation for developing practical robotic systems capable of operating in dynamic and complex real-world scenarios.

Despite achieving promising experimental outcomes, our system still faces certain limitations that could influence its effectiveness in practical applications. For example, sensor noise, especially from depth sensors, may lead to inaccuracies in object localization, negatively affecting navigation and grasping precision. Additionally, the system's performance might degrade under challenging lighting conditions, such as strong natural sunlight or uneven indoor lighting. Real-world scenarios might also present further complexities, including reflective surfaces, shadows, or cluttered environments, posing difficulties for reliable object detection and manipulation. Future work should address these limitations through enhanced sensor fusion strategies and adaptive control techniques, ensuring greater robustness and practical reliability.

7.2 Recommendation for Future Work

Although the experimental results are promising, there remain several avenues for further improvement. Future research will focus on refining the object detection model to achieve even higher accuracy and robustness under diverse lighting conditions and complex backgrounds, while further reducing latency. The spiral search strategy, while effective, exhibits increased search times with larger distances and greater angular deviations; thus, investigating alternative or hybrid search strategies could help mitigate these limitations. Enhancing the visual servo control system by exploring adaptive control techniques or machine learning based dynamic PID tuning could enable the system to better accommodate variations in environmental conditions and system dynamics. Additionally, expanding experimental validations to more complex, unstructured environments is essential to assess the system's performance under real-world conditions. Incorporating additional sensing modalities, such as higher-resolution depth cameras or alternative proximity sensors, may further enhance obstacle avoidance and target detection, especially in cluttered or low-contrast scenarios. Finally, as the system scales to address larger operational areas or more complex tasks, optimizing energy consumption and extending battery life through more efficient hardware and software optimizations will be crucial for maintaining performance while ensuring prolonged operational capability.

References

Alqahtani, D. K., Cheema, A., & Toosi, A. N. (2024). Benchmarking deep learning models for object detection on edge computing devices [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2409.16808

Arnold, R. D., Yamaguchi, H., & Tanaka, T. (2018). Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. Journal of International Humanitarian Action, 3(1), 18. https://doi.org/10.1186/s41018-018-0045-4

Azevedo, F., Cardoso, J. S., Ferreira, A., Fernandes, T., Moreira, M., & Campos, L. (2021). Efficient reactive obstacle avoidance using spirals for escape. Drones, 5(2), 51. https://doi.org/10.3390/drones5020051

Baghbanbashi, M., Raji, M., & Ghavami, B. (2023). Quantizing YOLOv7: A comprehensive study. In Proceedings of the 2023 28th International Computer Conference, Computer Society of Iran (CSICC) (pp. 1–5). IEEE. https://doi.org/10.1109/CSICC58665.2023.10105310

Beltrán-Escobar, M., Alarcón, T. E., Rumbo-Morales, J. Y., López, S., Ortiz-Torres, G., & Sorcia-Vázquez, F. D. J. (2024). A review on resource-constrained embedded vision systems-based tiny machine learning for robotic applications. Algorithms, 17(11), 476. https://doi.org/10.3390/a17110476

Bermudez, G., Pedro, G. D. G., Medeiros, V. S., & Boaventura, T. (2024). Comparative analyses of ROS local planners for quadrupedal locomotion: A study in real and simulated environments. In K. Berns, M. O. Tokhi, A. Roennau, M. F. Silva, & R. Dillmann (Eds.), Walking robots into real world (pp. 294–303). Springer Nature. https://doi.org/10.1007/978-3-031-71301-9_28

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2004.10934

Bouzoualegh, S., Guechi, E.-H., & Kelaiaia, R. (2019). Model predictive control of a differential-drive mobile robot. Acta Universitatis Sapientiae, Electrical and Mechanical Engineering, 10(1), 20–41. https://doi.org/10.2478/auseme-2018-0002

Burlington, S., & Dudek, G. (1999). Spiral search as an efficient mobile robotic search technique. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) (pp. 138–143). AAAI Press. https://cim.mcgill.ca/~mrl/pubs/scottyb/burl-aaai99.pdf

Camara de M. Santos, R. C., Coelho, M., & Oliveira, R. (2024). Real-time object detection performance analysis using YOLOv7 on edge devices. IEEE Latin America Transactions, 22(10), 799–805. https://doi.org/10.1109/TLA.2024.10705971

Çabuk, V. U., Kubilay Şavkan, A., Kahraman, R., Karaduman, F., Kırıl, O., & Sezer, V. (2018). Design and control of a tennis ball collector robot. In Proceedings of the 2018 6th International Conference on Control Engineering & Information Technology (CEIT) (pp. 1–6). IEEE. https://doi.org/10.1109/CEIT.2018.8751917

Cai, Y., Li, H., Yuan, G., Niu, W., Li, Y., Tang, X., Ren, B., & Wang, Y. (2020). YOLOvile: Real-time object detection on mobile devices via compression-compilation co-design [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2009.05697

Castelli, F., Michieletto, S., Ghidoni, S., & Pagello, E. (2017). A machine learning-based visual servoing approach for fast robot control in industrial setting. International Journal of Advanced Robotic Systems, 14(6), 1729881417738884. https://doi.org/10.1177/1729881417738884

Chaumette, F., & Hutchinson, S. (2006). Visual servo control. I. Basic approaches. IEEE Robotics & Automation Magazine, 13(4), 82–90. https://doi.org/10.1109/MRA.2006.250573

Chaumette, F., & Hutchinson, S. (2007). Visual servo control. II. Advanced approaches. IEEE Robotics & Automation Magazine, 14(1), 109–118. https://doi.org/10.1109/MRA.2007.339609

Chen, R., Wang, P., Lin, B., Wang, L., Zeng, X., Hu, X., Yuan, J., Li, J., Ren, J., & Zhao, H. (2025). An optimized lightweight real-time detection network model for IoT embedded devices. Scientific Reports, 15, 3839. https://doi.org/10.1038/s41598-025-88439-w

Choset, H. (2000). Coverage of known spaces: The boustrophedon cellular decomposition. Autonomous Robots, 9(3), 247–253. https://doi.org/10.1023/A:1008958800904

Colombo, F. T., de Carvalho Fontes, J. V., & da Silva, M. M. (2019). A visual servoing strategy under limited frame rates for planar parallel kinematic machines. Journal of Intelligent & Robotic Systems, 96(1), 95–107. https://doi.org/10.1007/s10846-019-00982-7

Cong, V. D., & Hanh, L. D. (2023). A review and performance comparison of visual servoing controls. International Journal of Intelligent Robotics and Applications, 7(1), 65–90. https://doi.org/10.1007/s41315-023-00270-6

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1 [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1602.02830

Cui, M., Liu, H., Wang, X., & Liu, W. (2023). Adaptive control for simultaneous tracking and stabilization of wheeled mobile robot with uncertainties. Journal of Intelligent & Robotic Systems, 108(3), 46. https://doi.org/10.1007/s10846-023-01908-0

Draelos, M. (2023). Time-optimal spiral trajectories with closed-form solutions. IEEE Robotics and Automation Letters, 8(4), 2213–2220. https://doi.org/10.1109/LRA.2023.3240361

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. Computer, 22(6), 46–57. https://doi.org/10.1109/2.30720

Fang, W., Wang, L., & Ren, P. (2020). Tinier-YOLO: A real-time object detection method for constrained environments. IEEE Access, 8, 1935–1944. https://doi.org/10.1109/ACCESS.2019.2961959

Feng, W., Zhu, Y., Zheng, J., & Wang, H. (2021). Embedded YOLO: A real-time object detector for small intelligent trajectory cars. Mathematical Problems in Engineering, 2021(1), 6555513. https://doi.org/10.1155/2021/6555513

Fong, T., Bualat, M., Deans, M., Allan, M., Bouyssounouse, X., Broxton, M., Edwards, L., Elphic, R., Fluckiger, L., Frank, J., Keely, L., Kobayashi, L., Lee, P., Lee, S., Lees, D., Park, E., Pedersen, L., Smith, T., To, V., … Schreckenghost, D. (2008). Field testing of utility robots for lunar surface operations. In Proceedings of the AIAA SPACE 2008 Conference & Exposition. https://doi.org/10.2514/6.2008-7886

Fricke, G. M., Hecker, J. P., Griego, A. D., Tran, L. T., & Moses, M. E. (2016). A distributed deterministic spiral search algorithm for swarms. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4430–4436). IEEE. https://doi.org/10.1109/IROS.2016.7759652

Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K., & Scaramuzza, D. (2022). Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(1), 154–180. https://doi.org/10.1109/TPAMI.2020.3008413

Gans, N. R., Hu, G., Shen, J., Zhang, Y., & Dixon, W. E. (2012). Adaptive visual servo control to simultaneously stabilize image and pose error. Mechatronics, 22(4), 410–422. https://doi.org/10.1016/j.mechatronics.2011.09.008

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1311.2524

Gong, X., Gao, Y., Wang, F., Zhu, D., Zhao, W., Wang, F., & Liu, Y. (2024). A local path planning algorithm for robots based on improved DWA. Electronics, 13(15), 2965. https://doi.org/10.3390/electronics13152965

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Transactions on Robotics, 23(1), 34–46. https://doi.org/10.1109/TRO.2006.889486

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1506.02626

Han, T., Zhu, H., & Yu, D. (2024). Data-driven model predictive control for uncalibrated visual servoing. Symmetry, 16(1), 0048. https://doi.org/10.3390/sym16010048

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1704.04861

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1712.05877

Jeong, E., Kim, J., & Ha, S. (2022). Tensorrt-based framework and optimization methodology for deep learning inference on Jetson boards. ACM Transactions on Embedded Computing Systems, 21(5), 51:1–51:26. https://doi.org/10.1145/3508391

Jeong, E., Kim, J., Tan, S., Lee, J., & Ha, S. (2022). Deep learning inference parallelization on heterogeneous processors with TensorRT. IEEE Embedded Systems Letters, 14(1), 15–18. https://doi.org/10.1109/LES.2021.3087707

Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., & Kompatsiaris, Y. (2017). Near-duplicate video retrieval by aggregating intermediate CNN layers. In L. Amsaleg, G. Þ. Guðmundsson, C. Gurrin, B. Þ. Jónsson, & S. Satoh (Eds.), MultiMedia modeling (pp. 251–263). Springer. https://doi.org/10.1007/978-3-319-51811-4_21

Kum, S., Oh, S., Yeom, J., & Moon, J. (2022). Optimization of edge resources for deep learning application with batch and model management. Sensors, 22(17), 6717. https://doi.org/10.3390/s22176717

Labbé, M., & Michaud, F. (2019). RTAB-Map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics, 36(2), 416–446. https://doi.org/10.1002/rob.21831

Ladosz, P., Mammadov, M., Shin, H., Shin, W., & Oh, H. (2024). Autonomous landing on a moving platform using vision-based deep reinforcement learning. IEEE Robotics and Automation Letters, 9(5), 4575–4582. https://doi.org/10.1109/LRA.2024.3379837

Lazarevich, I., Grimaldi, M., Kumar, R., Mitra, S., Khan, S., & Sah, S. (2023). YOLOBench: Benchmarking efficient object detectors on embedded systems [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2307.13901

Lee, J., Lee, M., & Park, S.-Y. (2021). Complete 3D foot scanning system using 360 degree rotational and translational laser triangulation sensors. International Journal of Control, Automation and Systems, 19(9), 3013–3025. https://doi.org/10.1007/s12555-020-0147-9

Li, H., Wu, H., Lou, L., Kühnlenz, K., & Ravn, O. (2012). Ping-pong robotics with high-speed vision system. In Proceedings of the 2012 International Conference on Control Automation Robotics & Vision (ICARCV) (pp. 106–111). https://doi.org/10.1109/ICARCV.2012.6485142

Li, K., Wang, Y., & Hu, Z. (2023). Improved YOLOv7 for small object detection algorithm based on attention and dynamic convolution. Applied Sciences, 13(16), 9316. https://doi.org/10.3390/app13169316

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., & Han, S. (2023). Tiny machine learning: Progress and futures. IEEE Circuits and Systems Magazine, 23(3), 8–34. https://doi.org/10.1109/MCAS.2023.3302182

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection [Preprint]. arXiv. https://doi.org/10.48550/ARXIV.1708.02002

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), Computer vision – ECCV 2016 (Vol. 9905, pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2

Lu, Q., Du, D., Zhang, C., Fei, M., & Rakić, A. (2021). Guaranteed cost control of networked inverted pendulum visual servo system with computational errors and multiple time-varying delays. In Q. Han, S. McLoone, C. Peng, & B. Zhang (Eds.), Intelligent equipment, robots, and vehicles (pp. 583–592). Springer. https://doi.org/10.1007/978-981-16-7213-2_56

Machkour, Z., Ortiz-Arroyo, D., & Durdevic, P. (2021). Classical and deep learning based visual servoing systems: A survey on state of the art. Journal of Intelligent & Robotic Systems, 104(1), 11. https://doi.org/10.1007/s10846-021-01540-w

Maybeck, P. S. (1990). The Kalman filter: An introduction to concepts. In I. J. Cox & G. T. Wilfong (Eds.), Autonomous robot vehicles (pp. 194–204). Springer. https://doi.org/10.1007/978-1-4613-8997-2_15

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., & Wu, H. (2018). Mixed precision training [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1710.03740

Mueggler, E., Huber, B., & Scaramuzza, D. (2014). Event-based, 6-DOF pose tracking for high-speed maneuvers. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2761–2768). IEEE. https://doi.org/10.1109/IROS.2014.6942940

Nguyen-Dang, H.-N., Nguyen-Thi, T.-L., Huynh-Hoang, K., Tran, H.-T., Nguyen, T.-T., Tran, T.-A., Le-Nhat, Q., & Nguyen-An, K. (2022). Design and development indoors autonomous ping-pong collection robot with vision system. In Proceedings of the NAFOSTED Conference on Information and Computer Science (NICS) (pp. 333–338). IEEE. https://doi.org/10.1109/NICS56915.2022.10013436

Novac, P.-E., Boukli Hacene, G., Pegatoquet, A., Miramond, B., & Gripon, V. (2021). Quantization and deployment of deep neural networks on microcontrollers. Sensors, 21(9), 2984. https://doi.org/10.3390/s21092984

Pandey, N. P., Nagel, M., Baalen, M. van, Huang, Y., Patel, C., & Blankevoort, T. (2023). A practical mixed precision algorithm for post-training quantization [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2302.05397

Peng, Y.-C., Jivani, D., Radke, R. J., & Wen, J. (2020). Comparing position- and image-based visual servoing for robotic assembly of large structures. In Proceedings of the 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE) (pp. 1608–1613). IEEE. https://doi.org/10.1109/CASE48305.2020.9217028

Piacentini, C., Bernardini, S., & Beck, J. C. (2019). Autonomous target search with multiple coordinated UAVs. Journal of Artificial Intelligence Research, 65(1), 519–568. https://doi.org/10.1613/jair.1.11635

Queralta, J. P., Raitoharju, J., Gia, T. N., Passalis, N., & Westerlund, T. (2020). AutoSOS: Towards multi-UAV systems supporting maritime search and rescue with lightweight AI and edge computing [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2005.03409

Ravankar, A., Ravankar, A. A., Kobayashi, Y., Hoshino, Y., & Peng, C.-C. (2018). Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. Sensors, 18(9), 3170. https://doi.org/10.3390/s18093170

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779–788). IEEE. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1506.01497

Rodrigues Moreira, L. F., Moreira, R., Travençolo, B. A. N., & Backes, A. R. (2025). Deep learning based image classification for embedded devices: A systematic review. Neurocomputing, 623, 129402. https://doi.org/10.1016/j.neucom.2025.129402

Singh, C. D., He, B., Fermüller, C., Metzler, C., & Aloimonos, Y. (2024). Minimal perception: Enabling autonomy in resource-constrained robots. Frontiers in Robotics and AI, 11, 1431826. https://doi.org/10.3389/frobt.2024.1431826

Song, T., Huo, X., & Wu, X. (2020). A two-stage method for target searching in the path planning for mobile robots. Sensors, 20(23), 6919. https://doi.org/10.3390/s20236919

Soori, M., Arezoo, B., & Dastres, R. (2023). Artificial intelligence, machine learning and deep learning in advanced robotics: A review. Cognitive Robotics, 3, 54–70. https://doi.org/10.1016/j.cogr.2023.04.001

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 3310–3317). IEEE. https://doi.org/10.1109/ROBOT.1994.351061

Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1911.09070

Tang, Y., Zakaria, M. A., & Younas, M. (2025). Path planning trends for autonomous mobile robot navigation: A review. Sensors, 25(4), 1206. https://doi.org/10.3390/s25041206

Taufique, Z., Vyas, A., Miele, A., Liljeberg, P., & Kanduri, A. (2024). HiDP: Hierarchical DNN partitioning for distributed inference on heterogeneous edge platforms [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2411.16086

Tian, Z., Shen, C., Chen, H., & He, T. (2019). FCOS: Fully convolutional one-stage object detection [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1904.01355

Tokunaga, S., Premachandra, C., Premachandra, H. W. H., Kawanaka, H., Sumathipala, S., & Sudantha, B. S. (2021). Autonomous spiral motion by a small-type robot on an obstacle-available surface. Micromachines, 12(4), 375. https://doi.org/10.3390/mi12040375

Verma, G., Gupta, Y., Malik, A. M., & Chapman, B. (2021). Performance evaluation of deep learning compilers for edge inference. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 858–865). IEEE. https://doi.org/10.1109/IPDPSW52791.2021.00128

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7464–7475). IEEE. https://doi.org/10.1109/CVPR52729.2023.00721

Wang, L., Li, R., Sun, J., Liu, X., Zhao, L., Seah, H. S., Quah, C. K., & Tandianus, B. (2019). Multi-view fusion-based 3D object detection for robot indoor scene perception. Sensors, 19(19), 4092. https://doi.org/10.3390/s19194092

Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2020). Integer quantization for deep learning inference: Principles and empirical evaluation [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2004.09602

Xu, J., Cao, H., Li, D., Huang, K., Qian, C., Shangguan, L., & Yang, Z. (2020). Edge assisted mobile semantic visual SLAM. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM) (pp. 1828–1837). IEEE. https://doi.org/10.1109/INFOCOM41043.2020.9155438

Yamauchi, Y., & Yamashita, M. (2013). Pattern formation by mobile robots with limited visibility. In T. Moscibroda & A. A. Rescigno (Eds.), Structural information and communication complexity (pp. 201–212). Springer. https://doi.org/10.1007/978-3-319-03578-9_17

Yang, Y., Kim, D., & Choi, D. (2023). Ball tracking and trajectory prediction system for tennis robots. Journal of Computational Design and Engineering, 10(3), 1176–1184. https://doi.org/10.1093/jcde/qwad054

Ye, Y., Nie, Z., Liu, X., Xie, F., Li, Z., & Li, P. (2023). ROS2 real-time performance optimization and evaluation. Chinese Journal of Mechanical Engineering, 36(1), 144. https://doi.org/10.1186/s10033-023-00976-5

Ye, Z., He, Y., Pieters, R. S., Mesman, B., Corporaal, H., & Jonker, P. P. (2011). Bottlenecks and tradeoffs in high frame rate visual servoing: A case study. In Proceedings of the IAPR Conference on Machine Vision Applications (MVA), June 13–15, 2011, Nara, Japan (pp. 55–58)

Zhang, P., Zhang, M., & Liu, J. (2021). Real-time HD map change detection for crowdsourcing update based on mid-to-high-end sensors. Sensors, 21(7), 2477. https://doi.org/10.3390/s21072477

Zhang, Q., & Feng, W. (2020). A unified framework for adjustable robust optimization with endogenous uncertainty. AIChE Journal, 66(12), e17047. https://doi.org/10.1002/aic.17047

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2017). Shufflenet: An extremely efficient convolutional neural network for mobile devices [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.1707.01083

Zhang, Y., Zhao, Y., Xiong, R., Wang, Y., Wang, J., & Chu, J. (2014). Spin observation and trajectory prediction of a ping-pong ball. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 4108–4114). IEEE. https://doi.org/10.1109/ICRA.2014.6907456

Zhao, Y., Wu, J., Zhu, Y., Yu, H., & Xiong, R. (2017). A learning framework towards real-time detection and localization of a ball for robotic table tennis system. In Proceedings of the IEEE International Conference on Real-Time Computing and Robotics (RCAR) (pp. 97–102). IEEE. https://doi.org/10.1109/RCAR.2017.8311842

Zhou, Z., Guo, J., Zhu, Z., & Guo, H. (2024). Uncalibrated visual servoing based on Kalman filter and mixed-kernel online sequential extreme learning machine for robot manipulator. Multimedia Tools and Applications, 83(7), 18853–18879. https://doi.org/10.1007/s11042-023-16381-y

Zhu, D., Li, T., Ho, D., Wang, C., & Meng, M. Q.-H. (2018). Deep reinforcement learning supervised autonomous exploration in office environments. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (pp. 7548–7555). IEEE. https://doi.org/10.1109/ICRA.2018.8463213