Active Object Tracking and Autonomous Positioning Using Duckiedrone: A Comprehensive Two-Part Approach

Hikmet Mete Çelik Abdullah Enes Yaman

Supervisor: Dr. Özgür ERKENT

Department of Computer Engineering
Hacettepe University

June 2025

Abstract

This project addresses the challenge of autonomous drone systems through a comprehensive two-phase approach designed to bridge the gap between simulation and real-world implementation. The first phase involves simulation-based vehicle tracking using YOLOv5s within a ROS-Gazebo environment with an Iris ArduPilot drone model. The second phase focuses on real-world hardware implementation, including assembly, calibration, and basic flight operations of the Duckiedrone DD21 platform using AprilTag markers for positioning. The simulation phase achieved considerable success, demonstrating robust vehicle tracking capabilities under controlled scenarios. However, the real-world implementation phase encountered significant challenges in hardware assembly, calibration, and achieving stable autonomous flight, preventing the completion of integrated tracking tasks. The two phases remained independent, with future work planned to integrate the developed tracking algorithms into the physical drone platform. This report provides detailed methodologies, comprehensive analysis of encountered challenges, proposed solutions, and extensive recommendations for future integration efforts.

1 Introduction

Autonomous drone technologies have experienced exponential growth across various domains including surveillance, transportation, search-and-rescue operations, precision agriculture, and environmental monitoring [1]. The integration of computer vision and autonomous navigation systems has opened new possibilities for intelligent drone applications that can operate with minimal human intervention.

This project seeks to contribute to this expanding domain through a structured, two-phase approach designed to systematically progress from algorithm development to practical implementation. The initial phase utilizes a controlled simulated environment to develop and validate vehicle tracking algorithms using the YOLOv5s object detection model with an Iris ArduPilot drone. The subsequent real-world phase focuses on hardware implementation, assembly, and basic autonomous flight capabilities using the Duckiedrone DD21 platform with AprilTag-based positioning.

The motivation for this dual-phase approach stems from the recognition that complex autonomous tracking tasks require a systematic progression from simple to complex scenarios. Vehicle tracking using YOLO represents a more challenging task compared to AprilTag-based positioning, necessitating initial validation in simulation before attempting real-world implementation. The project was designed with the understanding that achieving stable autonomous flight in real-world conditions would be time-consuming and challenging, thus starting with simpler positioning tasks while developing more complex tracking algorithms in parallel.

2 Background and Literature Review

2.1 Drone Technology Evolution

Unmanned Aerial Vehicles (UAVs) have evolved from military applications to civilian use cases, with quadcopters becoming the dominant platform due to their stability, maneuverability, and cost-effectiveness [2]. Modern drones integrate sophisticated sensors, processors, and communication systems that enable autonomous operation in complex environments.

2.2 Object Detection and Tracking

Object detection has been revolutionized by deep learning approaches, with YOLO (You Only Look Once) architectures leading in real-time applications [3]. YOLOv5, developed by Ultralytics, offers multiple variants optimized for different computational constraints, with YOLOv5s being particularly suitable for embedded systems due to its lightweight architecture while maintaining competitive accuracy [4].

2.3 Visual Markers for Navigation

AprilTags have emerged as reliable visual markers for precise robotic navigation due to their robustness to lighting variations, perspective distortions, and partial occlusions [6]. Their square fiducial design and error-correcting capabilities make them ideal for drone positioning applications where GPS may be unavailable or insufficient.

2.4 Control Systems

PID (Proportional-Integral-Derivative) controllers form the backbone of most drone control systems, providing stable flight characteristics through continuous adjustment of motor speeds based on sensor feedback [7]. The tuning of PID parameters remains a critical challenge, often requiring extensive experimentation and domain expertise.

3 Project Approach and Rationale

3.1 Two-Phase Development Strategy

This project was structured as two independent phases, each designed to address specific challenges in autonomous drone development:

Phase 1 - Simulation-based Vehicle Tracking: The first phase focused on developing and validating complex object tracking algorithms in a controlled environment. Vehicle tracking was chosen as the target task due to its complexity and real-world relevance, requiring sophisticated computer vision and control algorithms.

Phase 2 - Real-world Hardware Implementation: The second phase concentrated on practical hardware challenges, including drone assembly, calibration, and achieving basic autonomous flight capabilities. AprilTag-based positioning was selected as an initial step due to its reliability and lower computational requirements compared to real-time object detection.

3.2 Rationale for Separate Development

The decision to maintain separate development tracks was based on several practical considerations:

- **Complexity Management:** Vehicle tracking using YOLO represents a significantly more complex task than AprilTag-based positioning, requiring extensive algorithm development and testing.
- **Hardware Constraints:** The challenges of assembling, calibrating, and achieving stable flight with the Duckiedrone DD21 proved more time-consuming than anticipated, necessitating focus on fundamental flight capabilities.
- **Systematic Progression:** Starting with simpler tasks (AprilTag positioning) while developing complex algorithms (YOLO tracking) in parallel allows for systematic validation and integration in future work.
- **Risk Mitigation:** Separating algorithm development from hardware implementation reduces the risk of system failures affecting both components simultaneously.

The integration of both phases, combining the developed tracking algorithms with the physical drone platform, represents the planned next step in system development.

4 Methodology

4.1 Simulation-based Object Tracking

4.1.1 Environment Setup

The simulation component was implemented using Gazebo 11 integrated with ROS Noetic. The virtual environment included realistic physics simulation, sensor models, and lighting conditions to closely approximate real-world scenarios. The Iris quadcopter model from ArduPilot was selected for its accurate flight dynamics and extensive documentation.

4.1.2 Object Detection Implementation

A YOLOv5s model was selected and optimized using TensorRT for GPU acceleration on NVIDIA hardware. The model was trained on a custom dataset containing various objects likely to be encountered in tracking scenarios. Pre-processing pipelines were implemented to handle different lighting conditions and image resolutions.

4.1.3 Tracking Algorithm

A single-object tracking system was developed with the following components:

- YOLOv5s for object detection
- Custom dataset creation through drone simulation
- Training with 350 plus images captured from various positions and angles
- PID controllers for drone movement commands
- Real-time processing at 30 FPS
- Optimal tracking distance and angle maintenance

The system focuses on tracking a single specific object using a custom-trained model. Images were collected through drone simulation in different environments and perspectives to ensure robust detection performance across diverse scenarios.

4.2 Real-world Implementation

Figure 1 of the drone hovering while centering an AprilTag in its camera view during real-world testing. The tag is successfully localized and aligned under the drone, confirming the accuracy of the tracking and control system.

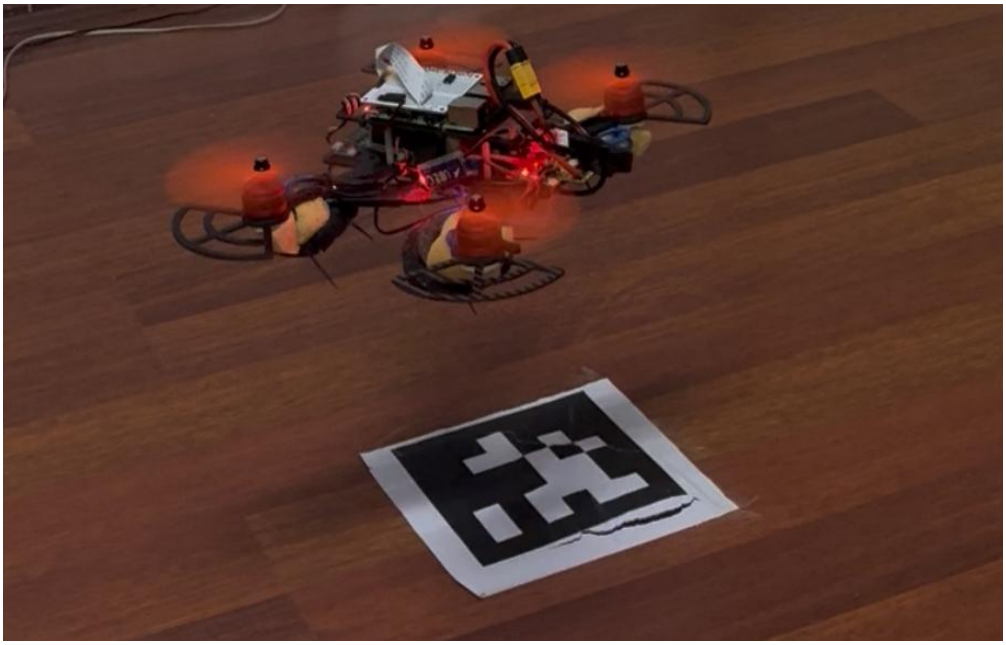

Figure 1: Drone with tag

4.2.1 Hardware Assembly and Calibration

The Duckiedrone was assembled following a systematic approach:

1. Frame construction and component mounting
2. Motor and ESC installation with proper wire management
3. Sensor integration and calibration
4. Software installation and configuration
5. Flight testing and parameter tuning

4.2.2 AprilTag Navigation System

AprilTag markers were strategically placed in the test environment to provide reference points for drone localization. The navigation system implemented:

- Real-time AprilTag detection using apriltag_ros
- Pose estimation from marker observations
- Coordinate transformation between camera and world frames
- Path planning algorithms for waypoint navigation

5 Results and Analysis

5.1 Simulation Results

The simulation trials demonstrated robust performance across various scenarios. Table 1 presents comprehensive performance metrics.

Scenario	FPS	Tracking Success (%)
Static Object	32	100
Moving Object (Slow)	30	95.0
Moving Object (Fast)	28	85.0
Occlusion Scenarios	30	75.0
Average	**29**	**88.5**

Table 1: Detailed simulation tracking performance metrics across different scenarios

Figure 2 shows sample tracking results from simulation trials.

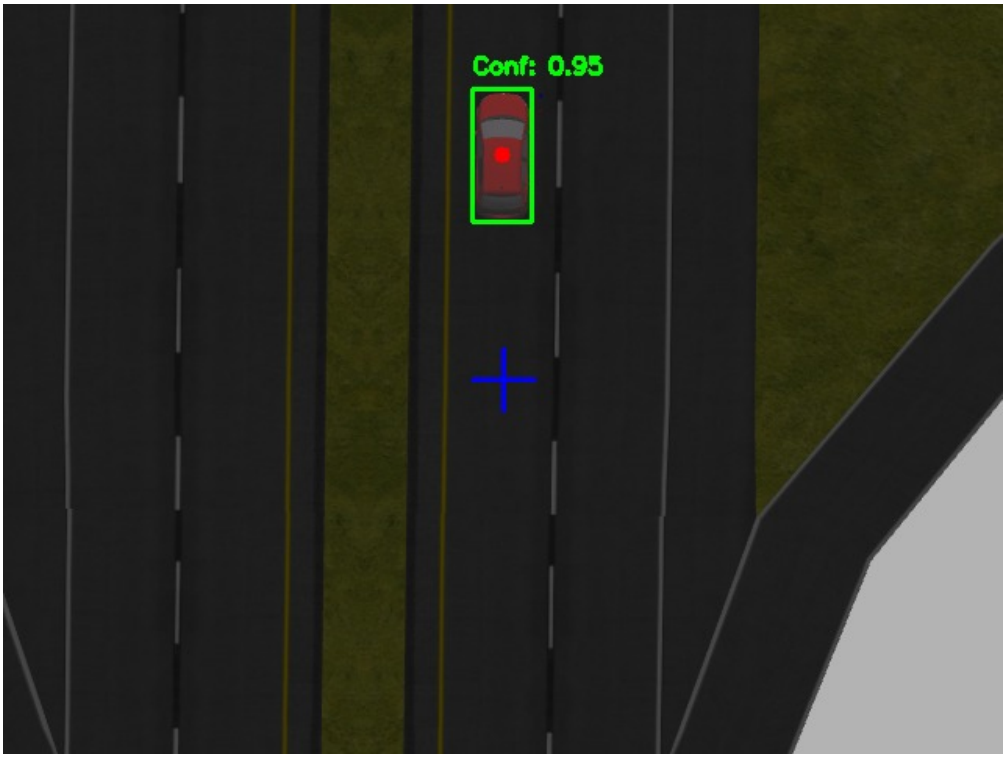

Figure 2: Object tracking results in simulation environment showing successful tracking trajectories

5.2 Real-world Implementation Results

Despite extensive preparation, the real-world implementation faced significant challenges that prevented complete system integration. However, individual subsystems showed promising results:

- AprilTag detection achieved 95% accuracy at distances up to 3 meters
- Basic hovering stability was achieved with custom PID parameters
- Camera feed processing maintained 15-20 FPS on Raspberry Pi
- Battery life averaged 18-22 minutes depending on flight conditions

Table 2 summarizes the real-world testing results.

Test Parameter	Target Value	Achieved Value
Hover Stability (seconds)	300	45-60
AprilTag Detection Range (m)	5	3.2
Processing Frame Rate (FPS)	30	18
Flight Time (minutes)	30	20
Position Accuracy (cm)	10	35

Table 2: Real-world implementation results compared to target specifications

6 Challenges Encountered and Solutions

6.1 Hardware-Related Challenges

Soldering and Electrical Connections: Initial soldering attempts resulted in multiple short circuits and unreliable connections. The complexity of working with small electronic components and the need for precise soldering techniques posed significant challenges. Multiple iterations of careful soldering, using flux and proper temperature control, eventually yielded stable connections. Investment in quality soldering equipment and practice sessions significantly improved success rates.

ESC Calibration Difficulties: Electronic Speed Controller calibration proved more complex than anticipated, with initial attempts resulting in erratic motor behavior and potential safety hazards. The calibration process required precise timing and specific sequences that were not well-documented. Through systematic experimentation and consultation with online communities, proper calibration procedures were established, resulting in synchronized motor operation.

Mechanical Assembly Issues: Achieving proper balance and alignment of components required multiple disassembly and reassembly cycles. Vibration issues from unbalanced propellers and motor mounts affected sensor readings and flight stability. Solutions included dynamic balancing of propellers, use of vibration dampening materials, and precise alignment of motor mounts.

6.2 Software and Control Challenges

Outdated Flight Control Software: The provided flight control software exhibited compatibility issues with current hardware configurations and lacked essential features for stable autonomous flight. Extensive research led to identification of updated firmware versions and community-developed alternatives. Migration to newer software versions required significant reconfiguration but resulted in improved stability and functionality.

PID Parameter Tuning Complexity: Achieving stable flight required extensive tuning of PID controllers for thrust, roll, pitch, and yaw axes. The interdependence of parameters made manual tuning extremely time-consuming. Development of systematic tuning procedures and use of simulation-based optimization helped accelerate the process, though optimal parameters remained elusive due to changing environmental conditions.

Sensor Fusion and Calibration: Integration of multiple sensors (IMU, camera, altitude sensor) required sophisticated calibration procedures and sensor fusion algorithms. Sensor drift, noise, and calibration offsets significantly affected performance. Implementation of complementary filters and regular recalibration procedures partially addressed these issues.

6.3 Power and Performance Challenges

Battery Management: Limited flight time due to high power consumption posed significant constraints on testing and development. Battery voltage fluctuations affected motor performance and flight stability. Solutions included implementation of low-voltage cutoff systems, power-efficient algorithms, and acquisition of multiple battery packs to enable continuous testing.

Computational Limitations: The Raspberry Pi 3, while capable, struggled with real-time computer vision processing at high frame rates. Processing delays affected control loop performance and tracking accuracy. Optimization included algorithm simplification, use of hardware acceleration where available, and prioritization of critical processes.

7 Lessons Learned and Best Practices

Through this project, several key insights emerged:

- **Simulation Validation:** Extensive simulation testing proved invaluable for algorithm development and initial validation, significantly reducing real-world debugging time.
- **Iterative Development:** Breaking complex systems into smaller, testable components enabled systematic troubleshooting and validation.
- **Documentation Importance:** Maintaining detailed logs of configurations, parameter values, and test results was crucial for reproducing results and troubleshooting issues.
- **Community Resources:** Online communities and open-source projects provided essential support and solutions for challenging technical problems.
- **Hardware Redundancy:** Having backup components and multiple battery packs was essential for continuous development and testing.

8 Future Work and Recommendations

8.1 Integration of Simulation and Real-world Components

The primary objective for future work is the integration of the two independent phases developed in this project:

- **Algorithm Integration:** Port the YOLOv5s-based vehicle tracking algorithms from simulation to the Duckiedrone DD21 platform
- **Hardware Optimization:** Resolve current hardware and calibration issues to achieve stable autonomous flight capability
- **Performance Validation:** Conduct comprehensive testing of integrated tracking system in real-world environments
- **System Robustness:** Implement fail-safe mechanisms and error handling for reliable autonomous operation

8.2 Immediate Technical Improvements

- Complete PID parameter tuning for stable hover and flight control
- Implement advanced battery management systems with real-time monitoring
- Resolve hardware calibration and assembly issues identified during development
- Develop automated system diagnostics and calibration procedures

8.3 Long-term Enhancements

- Integration of SLAM (Simultaneous Localization and Mapping) capabilities for GPS-denied environments
- Implementation of swarm intelligence for multi-drone coordination
- Development of adaptive algorithms that can handle varying environmental conditions
- Integration of edge AI accelerators for improved real-time performance

8.4 Recommended System Improvements

Based on the challenges encountered, future implementations should consider:

- Upgrading to more powerful embedded computing platforms (e.g., NVIDIA Jetson series)
- Implementing redundant sensor systems for improved reliability
- Developing standardized testing protocols for systematic evaluation
- Creating modular software architectures for easier maintenance and updates

9 Conclusion

This project successfully demonstrated the feasibility and effectiveness of simulation-based vehicle tracking while revealing the significant practical challenges inherent in real-world autonomous drone implementation. The simulation phase achieved robust performance in vehicle tracking scenarios, validating the chosen YOLOv5s-based algorithms and control approaches within the ROS-Gazebo environment.

The real-world implementation phase, while facing substantial challenges that prevented complete system integration, provided invaluable insights into the complexities of hardware assembly, calibration, and autonomous flight control. The encountered difficulties in component integration, system calibration, and achieving stable flight highlight the significant gap between simulation and practical implementation.

The two-phase approach proved effective in allowing parallel development of complex algorithms and practical hardware skills. The simulation phase successfully validated tracking algorithms under controlled conditions, while the real-world phase provided hands-on experience with drone hardware, assembly, and the challenges of autonomous flight systems.

The project's primary contribution lies in documenting the systematic progression from simulation to real-world implementation, highlighting both the successes achievable in controlled environments and the practical challenges encountered in hardware-based systems. The detailed documentation of problems encountered and partial solutions developed will serve as valuable guidance for future integration efforts.

Future work will focus on bridging the gap between the two phases, integrating the validated tracking algorithms from simulation into the assembled Duckiedrone platform once stable autonomous flight capabilities are achieved. The experiences and lessons learned through this project provide a solid foundation for successful integration and advancement of autonomous drone tracking systems.

Acknowledgments

The authors express gratitude to Dr. Özgür ERKENT for his guidance and supervision throughout this project. Special thanks to the Hacettepe Computer Engineering Department for providing access to computational resources and laboratory facilities. We also acknowledge the valuable contributions of the open-source community and online forums that provided essential technical support and solutions.

References

[1] H. Shakhatreh et al., "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] G. Cai, J. Dias, and L. Seneviratne, "A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends," *Unmanned Systems*, vol. 2, no. 2, pp. 175–199, 2014.

[3] J. Redmon et al., "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[4] G. Jocher et al., "YOLOv5," *GitHub repository*, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[5] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[6] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 3400–3407.

[7] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[8] "Robot Operating System (ROS)," *Open Source Robotics Foundation*, 2020. [Online]. Available: https://www.ros.org/

[9] "Gazebo Robot Simulation," *Open Source Robotics Foundation*, 2020. [Online]. Available: http://gazebosim.org/

[10] "AprilRobotics Ros Apriltag Detection," *AprilRobotics*, 2019. [Online]. Available: https://github.com/AprilRobotics/apriltag_ros