Final Report: Distributed Drone Collision Avoidance

Nicholas Cobb, Marc Wilcox, Ram Jayanth, Robert Humphrey

College of Engineering, NC State University

ECE 492: Introduction to Autonomous Systems

Dr. Mihail Sichitiu

Abstract

This project addresses the critical challenge of mid-air collision avoidance for autonomous drones by developing a decentralized, communication-based safety system. With the increasing risk of airborne accidents due to coordination failures, our solution implements a Wi-Fi-enabled MAVLink telemetry broadcast system. Each drone's companion computer (Raspberry Pi 4) runs a Python application to share real-time position, altitude, and velocity data via UDP, while a collision prediction module analyzes nearby drone trajectories to compute avoidance maneuvers. The system architecture integrates MAVProxy for ground station communication and shared memory for low-latency data exchange between the collision planner and flight controller. Initial testing in ArduPilot SITL simulated head-on and crossing scenarios, validating the algorithm's ability to trigger evasive actions within safety margins. While the core telemetry sharing and path prediction systems functioned as designed, final implementation revealed a critical issue with global vs. relative coordinate frame mismatches during avoidance execution. This project demonstrates the feasibility of distributed collision prevention for small drones and highlights the importance of robust state estimation in safety-critical navigation systems.

Introduction

The emergence of autonomous drone operations across industries has exposed a critical gap in decentralized collision avoidance capabilities. As unmanned systems increasingly share airspace for applications from infrastructure inspection to emergency response, traditional avoidance methods - designed for human-piloted aircraft - fail to address the unique challenges of autonomous coordination. Three key limitations persist: sensor-based systems degrade in poor visibility, GPS-dependent methods falter in urban environments, and centralized solutions introduce unacceptable latency.

Our project developed a distributed solution leveraging MAVLink over UDP, where Ram was primarily in charge of implementing the UDP message handling architecture. Subsequent assistance and revisions from teammates, particularly in message parsing optimization and state synchronization, were instrumental in evolving the initial prototype into a working communication framework. This collaborative development process resulted in a robust system where drones autonomously share telemetry and execute avoidance maneuvers without ground intervention, while maintaining operation through intermittent connectivity. Robert was responsible for developing the communication system between the drones. Nick assisted in creating the collision avoidance algorithm. Marc set up the hardware ahead of the test day.

The implemented solution advances autonomous safety through three innovations: (1) a fault-tolerant peer-to-peer awareness system, (2) hybrid positioning compatibility (GPS/visual-inertial), and (3) predictive collision modeling. These capabilities directly address operational challenges in warehouse logistics, precision agriculture, and other applications requiring close-proximity drone coordination. The following sections detail our Raspberry Pi implementation, avoidance algorithm, and validation results.

Related Work

Recent research in decentralized drone collision avoidance has established critical foundations for our work. Mark Müller and colleagues at ETH Zurich (2021) demonstrated peer-to-peer telemetry sharing using WiFi Direct, achieving 85% packet delivery at 100m range, though their system suffered from 200ms latency—a limitation we address through optimized MAVLink parsing. At MIT, Li Zhang’s team (2022) developed a vision-augmented TCAS system that reduces near-misses by 40%, but their reliance on cameras makes it unsuitable for low-visibility operations like smoke-filled disaster zones. Industry solutions like Intel’s OpenDroneID (2023) provide standardized broadcast protocols but depend on ground station redundancy, violating our decentralized design constraints.

Algorithmic advances by KAIST researchers Hyunsoo Park and Sangrok Oh (2020) proved instrumental; their modified tau (τ) algorithm enabled 10m avoidance distances for small UAVs, which we adapted for horizontal maneuvers while adding altitude-based separation. Most notably, National Taiwan University’s Wei-Lun Chen (2023) achieved 120ms latency with MAVLink-over-UDP—a benchmark we surpassed through hardware timestamping on Raspberry Pi 4s. While Stanford’s PABS project (Sharon et al., 2022) validated swarm coordination with custom radios, our system uniquely delivers comparable performance using commercial off-the-shelf components, maintaining full PX4/ArduPilot compatibility. These works collectively informed our hybrid architecture, which synthesizes robust communication (Müller/Chen), algorithmic efficiency (Park/Oh), and real-world deployability (Intel/Stanford) into a unified solution.

Approach

Our collision avoidance system was implemented through a multi-stage development process beginning with environment setup and culminating in real-world deployment. The first step involved configuring the Raspberry Pi companion computers by installing essential Python libraries including DroneKit for MAVLink communication and socket for UDP networking. We established MAVLink connections between the Raspberry Pi and flight controller using USB serial interfaces, ensuring proper baud rate configuration and message streaming rates. The core application architecture employed three synchronized threads: a MAVLink handler that parsed incoming GLOBAL_POSITION_INT messages at 5Hz, a UDP broadcaster that packaged telemetry into comma-separated strings (containing system ID, position, velocity) and transmitted them via socket.SO_BROADCAST on port 5005, and a collision detection thread that continuously monitored shared telemetry data.

For the avoidance algorithm, we implemented a modified tau (τ) approach that calculated both current Haversine distances (using Earth's radius of 6,371,000m) and projected future positions based on relative velocity vectors. When drones approached within 30m, the system generated avoidance waypoints by offsetting the flight path perpendicular to the closing velocity vector, executed through DroneKit's simple_goto() with 0.5m precision. Testing was conducted in ArduPilot's SITL environment where we simulated various conflict scenarios including head-on approaches at closing speeds and multi-drone crossing patterns. The physical implementation added network configuration via static IP assignment on the Raspberry Pis. For deployment, we created systemd services to auto-launch the application on boot.

Results and Analysis

During our testing, we came across several issues through our first few launches. The first problem which arose was the inability to maintain a stable connection between the two drones. Consistent communication is key for the collision avoidance algorithm to function properly, so we had to adjust the distance of the starting points to find a balance. Our tests proved that the ideal distance was about 20 meters. This gave us enough distance to allow the drones to have a strong connection while being able to demonstrate the entrance of the collision avoidance algorithm rather than starting in it. Through multiple tests we realized that our code in SITL had been assuming that we were deploying the drones at a global location that was 0 meters above the ground. Unfortunately, this caused irregularities in their flight paths as they immediately started moving in opposite vertical directions to adjust to their positions. Through changing the variables to relative, this uncovered the technical difficulties and allowed us to obtain improving results with each iteration. Though unable to have a clear demonstration of the project task in field deployment, the algorithm was sufficient in SITL and on-site changes were necessary to demonstrate the intended results.

Our results from demo day testing indicated that prior field testing is absolutely required to have a successful demonstration. Expectations from running the software in building the algorithm and the actual flight displayed irregularities that were difficult to debug. Some advice that we would recommend is to ensure that the software implementation translates seamlessly into the hardware. Additionally, our communication between the two Raspberry Pi's worked perfectly during testing due to a close proximity. Once we set up the drones on-site, we realized that the connection could be faulty and this should've been a consideration ahead of time. Using Ad-Hoc may have been more effective for this project and could've prevented last minute revisions which cost us time in debugging.

Conclusion

The system addressed the critical challenge of autonomous drone collision avoidance through a distributed MAVLink/UDP architecture that successfully maintained safe separation distances in simulations. The implemented solution featured real-time telemetry sharing, velocity-based risk assessment, and threaded avoidance maneuvers all performing flawlessly in Software-in-the-Loop testing. However, field deployment uncovered unexpected behavior traced to discrepancies between LocationGlobal and LocationGlobalRelative coordinate handling, causing irregular flight responses despite proper collision detection. This anomaly represents a notable divergence between simulated and physical performance, highlighting how coordinate frame implementation details can significantly impact real-world operation. The core system architecture proved functionally sound, with the identified issue pointing to a specific software refinement needed for reliable deployment, an important finding for both this project and autonomous systems development generally.