Autonomous quadrotors Landing System on Mobile Platform

Software Architecture and Containerized Deployment

September 2, 2025

Abstract

This document presents the architectural design of a simulation system for autonomous landing of quadrotors on mobile platforms. The system integrates ArUco fractal marker pose estimation and ultra-wideband (UWB) technology for precise positioning. The deployment approach uses Docker containerization for reproducibility and portability.

Contents

1 ROS System Architecture

1.1 Modular Design Philosophy

The system is built as a ROS package with clear module separation to ease maintenance and evolution.

Hierarchical Package Structure

The `quadrotors_landing_vision` ROS package is organized into functional modules (control, positioning, simulation, UGV).

1.2 Configuration Management

Key YAML configurations:

Configuration File	Functionality
`drone_fsm.yaml`	Finite state machine parameters for drone behavior
`landing_controller.yaml`	PID and log-polynomial controller configuration
`aruco_estimator.yaml`	Fractal marker pose estimation parameters
`uwb_lqr_vanc_estimator.yaml`	UWB system configuration with LQR filtering
`uwb_simulator.yaml`	Ultra-wideband signal simulation parameters
`ugv_control.yaml`	Mobile ground platform control settings

Table 1: System configuration files

1.3 Core Control Scripts

Drone Control: `drone_fsm.py`, `landing_controller.py`
Positioning: `aruco_estimator.py`, `uwb_lqr_vanc_estimator.py`
Simulation: `uwb_simulator.py`, `ugv_control.py`

2 Specialized Python Modules

2.1 MAVROS Interface Layer

`offboard.py`: position/velocity commands, mode switching, vehicle state.

2.2 Environmental Disturbance Modeling

`env_disturbances.py`: wind, lighting, interference.

2.3 Advanced UWB Interpolation

`rmse_interpolation.py`, `sigma_interpolation.py`.

3 Containerized Deployment Strategy

3.1 Repository and Layout

```
git clone https://github.com/ethan-bns24/quadrotors_landing_vision
cd quadrotors_landing_vision/docker
```

Listing 1: Clone and enter docker folder

3.2 Build and Run (Compose)

```
docker-compose build
```

Listing 2: Build image fresh

```
docker-compose up
```

Listing 3: Start container

3.3 Access the Devel Container

```
docker exec -it <container_id> bash
```

Listing 4: Open a shell in the running container

3.4 Workspace Setup and Build (inside container)

```
# Build the catkin workspace
cd /quadrotors_landing_vision/workspace
catkin build
source devel/setup.bash

# Ensure environment and PX4 paths
source /entrypoint.sh
```

Listing 5: Source env and build catkin

3.5 Launch Simulation

```
roslaunch quadrotors_landing_vision quadrotors_landing_vision.launch positioning
    :="aruco" world_name:="sim_windy.world"
```

Listing 6: Launch with parameters

4 Conclusion

This setup aligns commands with the current project name (`quadrotors_landing_vision`), ensures the entrypoint is executable during Docker builds, and makes ROS scripts runnable. It mirrors the original workflow while keeping your renaming intact.