

SO(3) Resetability Control Suite: User & Developer Guide



Version 3.1

1. Introduction

1.1. Purpose

The SO(3) Resetability Control Suite is a comprehensive dashboard for analyzing the "resetability" of rotational systems based on SO(3) mathematics. It processes telemetry data from files or live hardware to calculate key metrics, provide domain-specific insights, and visualize system attitude in real-time. This tool is designed for engineers, researchers, and developers working with robotics, spacecraft, and other systems where understanding rotational dynamics is critical.

1.2. Key Features

- **Dual-Mode Operation:**
 -  **Live Mode:** Processes data from a file being written in real-time by external hardware.
 -  **Simulation Mode:** Replays historical telemetry from a CSV at variable speeds for detailed analysis.
- **Direct Hardware Integration:** A built-in UI to connect to serial devices (like an Arduino/ESP32 with an IMU) for true live data logging without using the command line.
- **Core Metrics:** Autonomously computes **R** (resetability), **θ_{net}** (net rotation), and **Predicted Reset Benefit**.
- **Rich Visualization:** Includes live-updating 2D metric plots and an interactive 3D attitude cube.
- **Modular Domains:** Easily extensible for different applications (Robot, Spacecraft, etc.), with a quick-access switcher on the main page.
- **Professional UX:** Remembers the user's last selected domain between visits and features a customizable dark/light theme.
- **Automated Reporting:** Automatically generates a PDF summary at the end of a simulation run.
- **Advanced Analysis:** Dedicated tabs for post-mission event analysis (with replay) and robust Monte Carlo simulations.

2. Setup and Installation

Follow these steps to set up the project on your local machine.

2.1. Prerequisites

- Git (for cloning the project repository).
- Python 3.9+

2.2. Installation Steps

Step 1: Get the Code

Clone the project repository to your local machine using Git.

code Bash

downloadcontent_copy

expand_less

- `git clone <your-repository-url>`
- `cd resetability_suite`

Step 2: Create and Activate a Virtual Environment (venv)

It is crucial to create a clean, isolated environment for the project.

code Bash

downloadcontent_copy

expand_less

- `# Create the virtual environment folder`
- `python -m venv venv`
-
- `# Activate the environment (for Windows PowerShell)`
- `.\venv\Scripts\Activate.ps1`
-
- `# Activate the environment (for Linux/macOS)`
- `source venv/bin/activate`

After activation, your terminal prompt should begin with (venv).

Step 3: Install Required Libraries

The requirements.txt file lists all necessary Python packages.

code Bash

downloadcontent_copy

expand_less

- # Ensure you are in the project's root directory
- pip install -r requirements.txt

3. Running the Application

3.1. Launching the Dashboard

To launch the dashboard, ensure your virtual environment is activated and you are in the project's root directory. The most reliable method is to run Streamlit as a Python module:

code Bash

downloadcontent_copy

expand_less

- python -m streamlit run app_resetability_live.py

The application will automatically open in your default web browser.

3.2. Generating Sample Data

If you do not have a telemetry file, you can generate a sample CSV for testing.

code Bash

downloadcontent_copy

expand_less

- # Ensure your venv is active
- python generate_telemetry_csv.py

This will create a new file at data/telemetry.csv.




4. User Interface Guide

The application is organized into a main view with a top bar and a sidebar for detailed controls.

4.1. Main UI Controls

- **Quick Domain Switcher:** A top bar with buttons to instantly switch between application domains (Robot, Spacecraft, Booster, Gravity Test).
- **Sidebar Controls:** Contains all detailed settings for the Live/Simulation tab.
- **Live Data Logger:** A section in the sidebar for connecting to real hardware.
 1. Connect your serial device (e.g., an IMU programmed to send quaternion data).
 2. Click **"Scan for Serial Ports"** to find your device.
 3. Select the correct port from the dropdown and click **"Start Logging."**
 4. The app will now process data from this device in real-time when "Live Mode" is active.
-
- **Automated Reporting:**
 1. In the sidebar, ensure the **"Auto-generate report on stop"** toggle is enabled.
 2. Run a simulation using the "Start" button.
 3. When you click the **"Stop"** button, a final PDF report will be generated and a download button will appear in the sidebar.
-

4.2. Application Tabs

-  **Live / Simulation:** The main dashboard for real-time analysis and simulation replay.
-  **Analysis & Reports:** For post-mission analysis of logged reset events, with filtering, replay, and export options.
-  **Monte Carlo Simulation:** A tool for running robust statistical analyses with configurable parameters.

5. Project Architecture for Developers

The project is organized into a clean, modular structure to promote maintainability and extensibility.

- app_resetability_live.py: The main entry point that initializes Streamlit and organizes the UI.
- python/: The core Python package containing all application logic.

- ui_*.py: Each file corresponds to a tab in the UI.
- ui_helpers.py: Contains reusable functions for plotting, data loading, etc.
- core_math.py: The primary analysis engine.
- so3_reset.py: The foundational, dependency-free math library.
- domains/: A sub-package where each file defines the logic for a specific application domain.
-
- live_data_logger.py: A script to log data from serial ports, used by the main UI.
- tests/: A pytest suite for unit testing core functions.
- cpp/ & ros2_pkg/: (Optional) A C++ implementation and a ROS 2 package for on-robot deployment.

5.1. How to Add a New Domain

1. **Create a Domain File:** Add a new .py file (e.g., drone.py) in the python/domains/ directory.
2. **Implement Functions:** In the new file, create domain_info(), load_telemetry(path), and summarize_domain(results_df).
3. **Register Domain:** In app_resetability_live.py, import the new module and add it to the DOMAIN_MAP dictionary. The UI will update automatically.

6. Troubleshooting

6.1. ModuleNotFoundError on Startup

- **Problem:** The app crashes with an error like ModuleNotFoundError: No module named 'serial'.
- **Cause:** You have not activated the project's virtual environment (venv) before running the application. The system is using a different Python installation that does not have the required packages.
- **Solution:** Stop the app, activate the correct venv (.\\venv\\Scripts\\Activate.ps1 or source venv/bin/activate), and re-run the app using python -m streamlit run app_resetability_live.py.

6.2. VS Code Shows "Missing Import" Errors

- **Problem:** The editor shows red squiggles under imports like from python.ui_live....
- **Cause:** VS Code is not configured to use the Python interpreter from your project's venv.
- **Solution:** Use the command palette (Ctrl+Shift+P), type Python: Select Interpreter, and choose the interpreter located inside your project's venv folder.
-