Test Plan Document

HALO: High Autonomous Low-SWaP Operations

Team Members

Sloan Hatter (shatter2022@my.fit.edu)
Blake Gisclair (bgisclair2022@my.fit.edu)

Faculty Advisor

Dr. Ryan T. White (rwhite@fit.edu)

September 29, 2025

Table of Contents

1. Introduction

1.1. Purpose

This test plan document discusses the testing plan and strategy that will be used to verify the features of HALO. The test plan outlines test cases, expected results, and verification methods used to validate those results. This document provides a record of all testing strategies intended for use in validating system functionality, along with potential risks the system may encounter and mitigation strategies for such cases.

1.2. Scope

This test plan covers testing the system's real-time data processing proficiency, accurate object detection capabilities, and low-SWaP hardware deployability. We will be testing to ensure that the model can correctly identify and locate components of a satellite system in real-time while running on a low-SWaP computer.

1.3. Objective

The objective of this test plan is to provide test cases that expose any potential issues or bugs that the system may encounter during run time and provide solutions to said issues before product deployment to ensure smooth operations.

2. Test Items

Model Quantization Integrity

- Verify that the 1-bit quantized model produces outputs that are consistent with the baseline model.
- Test resilience to bit-level errors in weights and activations.

Detection Accuracy

- Validate the detection of orbital objects across standard datasets.
- Confirm detection accuracy and robustness against false positives.

Performance on Low-SWaP Hardware

- Measure inference latency, throughput, and frame rate on Raspberry Pi and Jetson.
- Validate memory footprint and power consumption within specified constraints.

System Reliability & Stability

- Validate continuous inference over long durations without crashes or memory leaks.
- Test the model under different hardware conditions.

Deployment & Configuration Testing

- Verify installation and startup scripts run successfully on target hardware.
- Confirm that the model auto-loads and initializes correctly after reboots or test crashes.

3. Test Cases

3.1. Model Quantization Integrity

Test Case 1: Run inference on the baseline model and the binary model using the same dataset.

- **Expected Outcome:** Accuracy difference $\leq 2\%$
- **Alternative Outcome:** Accuracy difference $> 2\%$

Test Case 2: Inject a single-bit flip error into quantized weights.

- **Expected Outcome:** Degraded accuracy, but the system still classifies
- **Alternative Outcomes:**
  - The model completely crashes
  - Silent corruption leads to random, unusable outputs

3.2. Detection Accuracy

Test Case 1: Input orbital object space images.

- **Expected Outcome:** Bounding boxes overlap with ground truth, Intersection over Union (IoU) ≥ 0.4
- **Alternative Outcomes:**
  - No objects are detected
  - Incorrect localization, IoU < 0.4
  - Multiple duplicate detections for a single object

Test Case 2: Input images of an empty sky.

- **Expected Outcome:** No detections returned
- **Alternative Outcome:** False positives, imaginary objects detected

Test Case 3: Input images of starfields.

- **Expected Outcome:** Stars are ignored, no detections are returned
- **Alternative Outcome:** False positives, stars are mistaken for objects

3.3. Performance on Low-SWaP Hardware

Test Case 1: Run inference on Raspberry Pi/Jetson.

- **Expected Outcome:** Throughput ≥ 30 frames per second (FPS)
- **Alternative Outcome:** FPS < throughput threshold

Test Case 2: Measure wattage usage

- **Expected Outcome:** System pulls between 10-20 watts
- **Alternative Outcome:** System exceeds wattage threshold

3.4. System Reliability & Stability

Test Case 1: Run the system under simulated CPU/GPU stress.

- **Expected Outcome:** Model still produces results within the latency bound threshold
- **Alternative Outcome:** Model produces detections, but too slowly and outside of the threshold

Test Case 2: Device enters thermal throttling

- **Expected Outcome:** Slower inference, but the system remains functional
- **Alternative Outcome:** Device overheats and shuts down

3.5. Deployment & Configuration Testing

Test Case 1: Install system on a clean low-SWaP device

- **Expected Outcome:** Dependencies get installed, and the model runs
- **Alternative Outcomes:**
  - Dependency conflicts prevent installations
  - Model loads but fails inference

Test Case 2: Reboot the device after installation

- **Expected Outcome:** Model starts automatically and resumes inference
- **Alternative Outcome:**
  - Model does not autoload
  - Model loads but fails initialization

4. Test Environment

4.1. Minimum Hardware/Software Requirements

Raspberry Pi AI HAT+

4.2. Recommended Hardware/Software Requirements

Jetson Xavier NX Series 8 GB

5. Risks and Contingencies

5.1. Potential Risks

5.1.1. Accuracy Degradation Due to Quantization

Reducing the model to a 1-bit quantization may cause significant accuracy loss compared to the baseline model. Some orbital objects might be missed, or false positives might increase. This would compromise mission reliability and integrity as incorrect detections could waste resources.

5.1.2. Performance Bottlenecks on Low-SWaP Hardware

Inference speed is too low, leading to the system's inability to process real-time data streams, causing delayed or missed detections.

5.1.3. Silent Failures

If the model's process fails silently, it can lead to inaccurate object detection or no object detection at all.

5.2. Mitigation Strategies

5.2.1. Training to Recoup Losses in Accuracy

Deploying training such as FP32 training and Post Training Quantization (PTQ) will help preserve accuracy and recoup any losses while reducing the model. Pre-deployment testing will be run extensively with representative orbital datasets as well.

5.2.2. Code Optimization/Hardware Swap

Code optimization with hardware-specific libraries, as well as the use of model pruning in tandem with quantization. If the Raspberry Pi AI HAT+ proves to have insufficient software capabilities, then the model will be deployed and run on one of the Jetson alternatives.

5.2.3. System Health Monitoring

Implementation of health monitoring, error logging alerts, and redundant checks can help identify silent failures. If the system stops producing detections for too long, the system will be killed and rebooted.

6. Success Criteria

6.1. All Functional Requirements Verified by Test Cases

All functional requirements from the Requirement Document must be tested and verified through the defined Test Items and Test Cases outlined in this Test Plan document. Specific criteria include:

- **Model Quantization integrity**: Binary quantization reduces model size and resource consumption while also maintaining accuracy within the specified tolerance threshold.
- **Detection Accuracy**: The system correctly detects and localizes orbital objects with an IoU ≥ 0.4 and a low false positive rate.
- **Performance on Low-SWaP Hardware**: The system achieves real-time throughput of at least 30 FPS while keeping wattage usage between 15 and 20 watts.
- **System Reliability & Stability**: The system runs under CPU/GPU strainage and has a fallback plan if system overload occurs.

- **Deployment & Configuration Testing**: The model is fully installed without any setbacks and starts back up successfully after rebooting.