AI Payload Processing Framework with Multi-Application Deployment for CubeSat FlatSat

Lab: Ω-Space Group, ORION Lab, National Technical University of Athens

Duration: 3-6 months

Level: Master Thesis

Background

Modern CubeSats increasingly rely on on-board AI/ML processing for autonomous operations and data reduction. This thesis focuses on developing a containerized payload processing framework for the CubeSat FlatSat testbed using Space ROS, with emphasis on deploying and managing multiple AI applications on edge computing hardware (NVIDIA Jetson). The work explores containerization strategies aligned with ESA's research into abstraction layers and hypervisors for satellite applications.

Objectives

Develop an AI payload processing system that:

1. Creates a containerized framework for deploying multiple AI/ML applications on NVIDIA Jetson
2. Implements application isolation and management based on telecommands from C&DH
3. Deploys one (ORION-Lab-developed) operational AI application (cloud detection) with end-to-end validation as well as one existing AI apllication, available online.
4. Integrates with Space ROS architecture for command/telemetry communication
5. Compares containerization vs. native ROS2 deployment approaches with trade-off analysis
6. Optimizes AI models for edge deployment (latency, power, memory constraints)

Scope of Work

Phase 1: AI Framework Development

Jetson Setup

- Configure NVIDIA Jetson with Ubuntu, Space ROS (ROS2), Docker/Podman, ML frameworks (TensorFlow/PyTorch, TensorRT)

Containerized Application Architecture

- Each AI application runs as Docker container with ROS2 inside (DDS over host network using --network=host)
- Containers built from ROS2 base images with AI dependencies added
- Standard interface: subscribe to camera topics, publish results to /ai/<app_name>/output
- **Note:** Each container includes its own ROS2 installation -standard Docker + ROS2 pattern

Application Manager (ROS2 node on host Jetson)

- Subscribes to /telecommand topic from C&DH
- Parses commands (e.g., START_APP cloud_detection, STOP_APP cloud_detection, LIST_APPS)
- Uses Docker SDK (Python) to start/stop/monitor containers
- Publishes application status to /payload/status topic
- Maintains application registry (JSON config file listing available apps)

Resource Monitoring

- Track CPU, GPU, memory usage per container
- Publish resource telemetry to /payload/resources topic
- Alert on resource threshold violations

AI Application Implementation - Cloud Detection

- **Phase 1a:** Develop and test with static images from memory/files (test dataset)
- **Phase 1b:** Integrate with RGB camera via existing ROS2 camera node (e.g., usb_cam)
- **Phase 1c:** Prepare for IR camera integration when hardware arrives
- Model training, optimization (quantization, pruning with TensorRT), deployment
- Containerized as reference AI application
- Publishes cloud detection results to /ai/cloud_detection/output topic

Phase 2: Alternative Deployment & Trade-off Analysis (Optional)

Native ROS2 Deployment

- Implement same AI applications as pure ROS2 nodes (no containers)
- Managed via ROS2 launch system
- Compare with containerized approach

Trade-off Analysis

- Resource overhead (memory, CPU, startup time)
- Isolation effectiveness
- Deployment ease and flexibility
- Performance (inference latency)
- Documented recommendations for CubeSat use cases

Integration & Testing

C&DH Integration

- Receive telecommands via /telecommand topic
- Send telemetry and AI results to C&DH via ROS2 topics
- End-to-end validation: telecommand → app start → inference → results → telemetry

Camera Integration

- Configure ROS2 camera drivers (RGB: usb_cam or v4l2_camera)
- Image preprocessing pipeline
- IR camera driver setup when hardware arrives

Performance Benchmarking

- Inference time, accuracy, power consumption
- Container overhead measurements
- Multi-application scenarios (running 2+ apps simultaneously)

Technical Requirements

- **Operating System:** Ubuntu 22.04 on NVIDIA Jetson
- **Middleware:** Space ROS (ROS2 Humble or Iron)
- **Containerization:** Docker or Podman
- **Programming Languages:** Python for AI/ML and ROS2 nodes, C/C++ for performance-critical components if needed
- **ML Frameworks:** TensorFlow/PyTorch, TensorRT for optimization
- **Communication:** ROS2 DDS over Gigabit Ethernet
- **Version Control:** All code contributed to project GitHub repository
- **Testing:** Unit tests, integration tests, performance benchmarks

Deliverables

1. Containerized AI payload framework running on NVIDIA Jetson
2. Application Manager with telecommand-based app control
3. Operational cloud detection AI model (containerized) with performance metrics
4. Camera integration (RGB working, IR ready for integration)
5. Integration with C&DH subsystem via Space ROS
6. Resource monitoring and telemetry system
7. (Optional) Native ROS2 deployment comparison and trade-off analysis
8. Source code, dockerfiles and documentation in GitHub repository
9. Master's thesis report documenting design, implementation, validation and performance analysis

Prerequisites

- Programming skills in Python, familiarity with C/C++
- Experience with machine learning frameworks (TensorFlow/PyTorch)
- Understanding of computer vision and image processing
- Familiarity with containerization (Docker) beneficial but not required
- ROS2 knowledge beneficial but not required
- Understanding of embedded systems and resource-constrained computing

Why This Thesis?

- **AI in Space:** Work on cutting-edge on-board AI for satellite applications
- **Modern Architecture:** Explore containerization for space systems (ESA research)
- **Edge Computing:** Optimize AI models for resource-constrained hardware
- **Hands-on ML:** Deploy real AI applications on actual hardware (NVIDIA Jetson)
- **Open Source:** Contribute to open-source CubeSat community (GPL-3.0)
- **Industry Relevance:** Skills applicable to space industry, edge AI, and robotics

Management

- Weekly coordination meetings ensure project tracking and knowledge sharing.
- All work is documented and shared via GitHub, following open-source guidelines and contributing to the open-source community.
- Master thesis can overlap with other theses and/or ongoing developments in the working group. Collaboration is encouraged but all topics have been designed to not introduce blocking points.

Contact

Simon Vellas: svellas@mail.ntua.gr

Alexis Apostolakis: alexis.apostolakis@gmail.com

Giorgos Athanasiou: georgios.athanasiou.ntua@gmail.com

Expected Start: January 2026