Technical Solution Document: AI Tools for Maritime Domain Awareness (Stage-1)
Project: PS-09 - Grand Challenge: AI tools for Maritime Domain Awareness
Stage: Stage-1 - End-to-End Dockerized Solution
Version: 1.0
Date: August 24, 2025

1. Introduction and High-Level Architecture

This document outlines a complete, end-to-end technical solution for Stage-1 of the Maritime Domain Awareness (MDA) challenge. The proposed solution is a modular, sequential pipeline designed to be packaged within a Docker container for a fully offline, automated workflow.
The architecture is composed of four primary modules that directly correspond to the challenge's objectives:
Data Ingestion & Pre-processing: Prepares raw satellite and AIS data for analysis.
Vessel Detection: Identifies vessels in satellite imagery using a deep learning model.
AIS-Detection Correlation: Matches detected vessels with AIS signals.
AIS Path Interpolation: Reconstructs high-fidelity vessel tracks from sparse data.
The entire system is designed with the specified evaluation metrics—Average Precision (AP) for detection, F1-Score for correlation, and Root Mean Squared Error (RMSE) for interpolation—as the primary drivers for algorithm selection.

2. Module-by-Module Technical Specification

2.1. Module 1: Data Pipeline and Pre-processing
This module is responsible for ingesting all raw data and preparing it for the core ML/AI algorithms.
Data Ingestion:
Imagery: Reads Sentinel-1 (SAR) TIFF and Sentinel-2 (EO) JP2 files using the rasterio and GDAL libraries.
AIS Data: Ingests CSV files containing AIS messages using the pandas library for efficient data manipulation.
Pre-processing Steps:
Land Masking (Critical Prerequisite): To ensure all processing is focused on "open-sea" detections, a land mask is applied first. The provided Natural Earth 10m coastline shapefile will be read using geopandas. The rasterio.mask function will then be used to set all land pixels in the imagery to a no-data value, effectively removing them from consideration by downstream modules.
Cloud Masking (for EO): To improve detection accuracy in optical imagery, a simple and effective offline cloud masking technique will be implemented. The Normalized Difference Snow Index (NDSI), calculated from the Blue and SWIR bands, will be used to identify and mask cloud cover.
SAR Signal Processing (for SAR): To enhance the signal-to-noise ratio in SAR imagery, two standard steps will be applied:
Radiometric Calibration: Converts raw pixel values to normalized radar backscatter (Sigma Naught, σ^o), making detections more consistent across different scenes.

Speckle Filtering: A Lee Filter will be applied to reduce the characteristic "speckle" noise in SAR images, which clarifies vessel shapes and reduces false alarms.

2.2. Module 2: ML/DL Model for Vessel Detection

This module performs vessel detection on the pre-processed imagery.

Algorithm: YOLOv9 (You Only Look Once, version 9)

Justification:

State-of-the-Art Accuracy: YOLOv9 represents the cutting edge in real-time object detection, offering superior Average Precision (AP) compared to previous versions, which is the primary evaluation metric for this task.

Offline & Self-Contained: A trained YOLO model consists of a single weights file (.pt) that runs without any external dependencies, making it ideal for the required offline, dockerized environment.

Versatility: The architecture is robust enough to be trained effectively on both EO and SAR data. Two distinct sets of weights will be trained and packaged: yolov9_eo.pt and yolov9_sar.pt. The pipeline will dynamically load the appropriate model based on the input image type.

Implementation:

The model will be trained on the provided datasets to detect a single class: ship.

The output will be a set of bounding box coordinates (in pixel space, later converted to geo-coordinates) and a confidence score for each detection.

This output will be formatted into the specified GeoJSON format as per Appendix-B.

2.3. Module 3: Algorithm for AIS-Detection Correlation

This module associates the vessel detections from Module 2 with AIS tracks.

Algorithm: Gated Nearest Neighbor with Kinematic Extrapolation

Justification:

High F1-Score: This method is designed to minimize both false positives and false negatives, which is essential for maximizing the F1-Score. It avoids naive matches by accounting for vessel movement.

Handles Spatio-Temporal Gaps: It explicitly calculates a vessel's likely position at the time of image capture, bridging the time gap between the AIS ping and the satellite pass.

Computational Efficiency: The algorithm is fast and efficient, making it suitable for processing large datasets within the time constraints of the challenge.

Implementation:

Extrapolation: For each AIS point, its position is projected forward or backward to the exact timestamp of the satellite image using its last known Speed Over Ground (SOG) and Course Over Ground (COG).

Gating: For each detected vessel, a circular "gate" (search radius) is established around its centroid. Only extrapolated AIS positions falling within this gate are considered potential matches. This step drastically reduces the search space and prevents illogical associations.

Assignment: Within the gate, the match is made by selecting the extrapolated AIS point with the minimum Euclidean distance to the detected vessel's centroid.

The final correlations are written to correlation.csv as specified.

2.4. Module 4: Algorithm for Path Interpolation
This module reconstructs a smooth, high-resolution path from sparse AIS points.
Algorithm: Cubic Hermite Spline Interpolation
Justification:
Minimizes RMSE: The primary metric, RMSE, heavily penalizes deviations from the true path. Unlike linear interpolation which fails on curves, a Cubic Hermite Spline uses the position and velocity (derived from SOG/COG) at each point. This ensures the interpolated path is smooth (C1 continuous) and accurately models vessel turns, leading to a significantly lower RMSE.
Kinematically Sound: The resulting path respects the physics of vessel movement, making it more realistic than purely geometric methods.
Offline and Self-Sufficient: This method requires no external data—only the sparse AIS track itself—making it perfect for the offline requirement.
Implementation:
The scipy.interpolate.CubicHermiteSpline function will be used.
For each pair of consecutive AIS points in a track, the positions (lat, lon) and the velocity vectors (calculated from SOG and COG) will be used as inputs.
The spline will be used to generate intermediate points at a desired temporal resolution. SOG and COG for these new points will be linearly interpolated.
The final high-resolution track is written to interpolation.csv as specified.

3. Dockerized Solution and Deployment

The entire solution will be delivered as a single, self-contained Docker image for evaluation.
Base Image: ubuntu:24.04
Dependencies: The Dockerfile will manage the installation of all system dependencies (e.g., python3.11, pip, libgdal-dev) and Python libraries (torch, rasterio, pandas, geopandas, numpy, scikit-learn, scipy, ultralytics).
Artifacts: The trained YOLOv9 model weights (.pt files) and the Natural Earth land-mask shapefile will be copied into the image during the build process, ensuring complete offline functionality.
Execution: An entrypoint.sh script will orchestrate the entire workflow, executing the Python scripts for each module in the correct sequence and ensuring all outputs are generated in the specified formats and directories.