REMIS: REal-time Maritime Intelligence from optical Satellite imagery - Executive Summary Report

Identifier: ST-VAKE-REMIS-RP-009

Version: 1.0

Date: 2024-04-17

Contractor: Science [&] Technology AS, Tordenskiolds gate 2, 0160 Oslo, Norway

Subcontractor: Vake AS, Universitetsgata 2, 0164 Oslo, Norway

Subcontractor: EmLogic AS, Strøket 3, 1383 Asker, Norway

Signatures

Prepared by:	@Andreas Thorvaldsen	Date: 17 Apr 2024
Issued by:	@Carolina Barrack Project Manager	Date: 17 Apr 2024
Approved by:		Date:

Table of Contents

Changelog

Version	Author	Affected sections	Reason	Date
1.0	Andreas Thorvaldsen (S&T)	All	Initial document creation	2024-04-17

Acronyms

ADCS	Attitude Determination and Control System
AI	Artificial Intelligence
AIS	Automatic Identification System (for ships)
AMD	Advanced Micro Devices
BRAM	Block Random Access Memory (small memory banks present in large numbers in FPGAs)
CCN	Contract Change Note
CNN	Convolutional Neural Network
DMA	Direct Memory Access
DP	DataPack
DPU	Deep learning Processing Unit
DRAM	Dynamic RAM
EO	Earth Observation
ESA	European Space Agency
FOV	Field Of View
FPGA	Field-Programmable Gate Array
fps	frames per second

GSD	Ground Sampling Distance
IP	Intellectual Property
L1	Level 1 (instrument data processing level)
L2	Level 2 (instrument data processing level)
LEO	Low Earth Orbit
MPSoC	Multi-Processor System-on-Chip
DRAM	Dynamic Random Access Memory
REMIS	REal-time Maritime Intelligence from optical Satellite imagery
RGB	Red-Green-Blue
RP	RePort
S&T or ST	Science [&] Technology AS (Oslo)
Tcl	Tool command language
TN	Technical Note
VHDL	VHSIC Hardware Description Language
VNIR	Visible and Near InfraRed
UltraRAM	UltraScale Random Access Memory (intermediate size memory banks in AMD/Xilinx FPGAs)

Purpose and Scope

This document provides a retrospective summary of the of the REMIS project and is a deliverable according to the contract [AD1] and the proposal [PROP]. The project was a collaboration between the companies S&T and Vake and funded by ESA. The company EmLogic was later brought into the project [CCN1] to provide expertise with FPGA programming. The aim of the project was to develop a prototype software/firmware "REMIS" implemented in FPGA, for real-time detection of ships in satellite imagery on board a satellite. Hope is that elements of the software/firmware and the techniques developed will be used in future in-orbit demonstrations and eventually form the basis of on-board processing software in operational missions.

The document is structured as follows:

- Chapter 1 presents the project objectives
- Chapter 2 describes how the project was carried out, the challenges faced and what has been implemented
- Chapter 3 presents the results, main conclusions and plans

Applicable documents

AD	Title	Reference	Date	Version
AD1	ESA Contract "Preparation of Enabling Space Technologies and Building Blocks"; "REMIS - REal-Time Maritime Intelligence from Optical Satellite Imagery"	No. 4000137746/21/NL/GLC/ces	2022-03-30	-
PROP	ESA IOD Proposal; REMIS: REal-time Maritime Intelligence from optical Satellite imagery	ST-ESA-IOD-REMIS-PROP-001	2021-11-26	2.0

CCN1	REMIS New WP 800 Add CCSDS 123 compression functionality in FPGA	CCN 1 of contract no. 4000137746/21/NL/GLC/ces	2023-01-19	-

Reference documents

RD	Title	Reference	Date	Version
MTR	REMIS - Mid-term Report	ST-VAKE-REMIS-TN-001	2023-02-09	1.3
FR	REMIS - Final Report	ST-VAKE-REMIS-TN-002	2024-04-17	1.3
DP	“REMIS datapack” containing source code, test imagery, trained ship detection model, results, etc.	ST-VAKE-REMIS-DP-003	2024-04-17	1.1
JF123	“An Efficient Real-Time FPGA Implementation of the CCSDS-123 Compression Standard for Hyperspectral Images” by J. Fjeldtvedt, M. Orlandić, T. A. Johansen	IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2018, doi.org/10.1109/JSTARS.2018.2869697, github.com/Jaffe-/ccsds123	2018	-

1 Project Objectives

With the continuously increasing data rates of space-born imaging instruments for Earth observation, and the less quickly increasing downlink bandwidth, it is becoming ever more advantageous to pre-process the raw instrument data on board the satellite, and depending on the content, avoid downlinking all the raw data. Such preprocessing may range from low-level operations like subsetting/region selection, binning and/or compression to basic L2 scientific retrievals. For example, the Sentinel-2 employs low-level image equalization (pixel non-uniformity correction) and compression. At the other end of the spectrum one finds the very recently launched HyTI (Hyperspectral Thermal Imager; NASA), which carries out full L1 processing on board.

With the advent of the Field-Programmable Gate Array (FPGA), it has become possible to carry out computationally intensive tasks with very modest power consumption. Computations that in a ground segment consume 100s of Watts can be implemented in an FPGA that only consumes 5-10W – at least in theory. The main challenge today is that FPGA programming is vastly more difficult than traditional computer programming, and this skill is only possessed by a minute, specially trained fraction of programmers. A strategic objective of the REMIS project was to contribute to “democratize” on-board processing by having a team of non-specialized software engineers implement relatively complex processing in FPGA as a proof of concept. The introduction of subcontactor EmLogic in the project (specialized FPGA engineers) somewhat contradicted this objective, but was deemed necessary to succeed with the development. The detailed technical objectives were the following 4:

High-level task: Ship detection

Of all high-level processing tasks than can be implemented on board, the one that would bring the most benefit is arguably ship detection (or *maritime intelligence* as we call it in project title). This is because the oceans are vast and an EO satellite in LEO spends the majority of its time with nothing to look at besides ocean. The ships are basically the only interesting piece of information in those images, but only occupy a tiny fraction of the pixels (<1 in a million). Identifying the ships on board and downlinking only those pixels would therefore extend the utility of the satellite without any significant increase in downlink volume. Project participant Vake specializes in ship detection in (downlinked) satellite imagery and routinely provides these detections to its clients in near real time, together with correlated AIS readings from other sources. Both general monitoring and discovery of violators of shipping regulations are interesting topics for Vake’s clients.

To make the detections, Vake employs deep machine learning models (CNNs). The detection processing currently takes place in cloud computers with GPUs using the open-source PyTorch library. Unfortunately, implementing a CNN processor in FPGA would be an undertaking too large for a project. Fortunately, though, AMD/Xilinx, the inventor and foremost producer of FPGAs, freely offers the Vitis AI

DPU, a CNN processor that is compatible with some of their FPGA chips. AMD/Xilinx also offers related tooling software, such as the Vitis AI PyTorch Quantizer, to facilitate the adoption of their CNN processor, which we have used in this project.

Preprocessing: Image equalization

Before imagery can be analysed with a CNN it must first be calibrated and made uniform – at least approximately. CNNs analyse spatially and radiometrially uniform images, thus both pixel non-uniformity and radiometric flat-fielding corrections are required. To make the imagery independent of illumination conditions, we have also included an elementary conversion from radiance to reflectance (so-called bottom-of-Rayleigh reflectance). This is convenient since the conversion takes the same form as the flat-fielding correction, and does not require detailed knowledge of the meteorological conditions. Furthermore, since small imagers typically have narrow FOV and operate with varying (and perhaps unstable) tilt angle to reach further from nadir, we also include an orthorectification step to resample the image to a constant, uniform GSD. Finally, we have included a spectral resampling/downsampling/binning step to approximately work around spectral differences between the satellite imager and the CNN model's training imagery – this last step is important since training imagery is only readily available from a select few satellites (predominantly Sentinel-2 and Landsat 8) and a new imager is unlikely to be spectrally equivalent to any of those.

Compared to the L1 and L2 processing carried out in a typical ground segment, our on-board approximation (which we call *equalization*) misses the most complicated steps: Straylight correction, smile correction and aerosol correction, which we deem too complex to bring on board. We nevertheless believe the equalized imagery to be sufficiently uniform for a CNN analysis, either because the missing corrections are small (straylight, smile), or because the un-corrected effects are present in the training imagery (e.g. aerosol) and thus well understood by the CNN.

Compression

Even if only a small portion of the imagery is downlinked, it is still desirable that it is compressed – preferably using lossy compression with a tuneable truncation threshold to disregard inherent measurement noise. In order to support this, we have incorporated an open-source CCSDS 123 compression processor [JF123] implemented in FPGA. This only support lossless compression, though. We have connected this to the output from the combined radiometric flat-fielding and reflectance conversion, where, in principle, no numerical information has been lost (orthorectification and spectral resampling are not reversible).

Demonstrator

The project should deliver a "demo" [DP] in the form of source code, imagery, data, results, logs and instructions on how to build it on a Linux PC and and run it on a AMD/Xilinx ZCU104 Evaluation Kit board. The source code should be richly commented and reveal any valuable tricks and know-how accumulated during the project, much of which isn't written in the project documentation. The ZCU104, which carries the ZU7EV UltraScale+ MPSoC FPGA chip, is well suited as it has a lot of room in the FPGA, 2GB DRAM, an SD card slot and DisplayPort and HDMI connectors that can be used for live demo output. Unfortunately, this board has more than doubled in price since the project started.

2 Execution and Implementation

The implementation of REMIS was carried out intermittently during the course of 2 years, about twice as long as the proposed schedule. As usual, several foreseen problems proved more difficult than foreseen and several unforeseen problems were encountered on the way, which we list here in the hope that it could benefit the reader *or* provoke the insightful reader to propose a solution to us:

1. Adapting imagery from one instrument to resemble imagery from another instrument (e.g. Sentinel-2 to Landsat 8 or PlanetScope SuperDove to Sentinel-2) is a complicated and difficult problem [MTR] – even if the band shapes are known precisely and and the band images are in the same units (i.e. radiance in W/m^2/sr), the fact that a target band isn't representable *exactly* as a linear combination of source bands leads to spurious-looking biases. We overcame this by retrieving statistics for the non-represented subspaces from a pair of near-contemporary images, but this technique isn't applicable if the target instrument is yet to be launched (since no pair of images will exist). Another solution must be invented to overcome this challenge.
2. AMD/Xilinx development software is not robust software. The stated Linux operating system versions supported (e.g. Ubuntu 20.04) should be taken literally, and so should the precise Vivado, PetaLinux and Vitis AI versions prescribed in a tutorial recipe. We experienced multiple very time-consuming and confusing errors due to slight version incompatibilities. Even with exact versions, some recipes do not work exactly as described. Modifications are best attempted only after the tutorial has been reproduced.

3. Linux kernel patching cannot be avoided. Rather than to write a new driver, we chose to generalize the xilinx_dma driver to support userspace control.
4. Userspace-driver-DMA-interrupt overhead is significant. Our xilinx_dma driver patch only implements commanding single DMA transaction at once, and every frame passed to the image equalization consists of two simultaneous transactions (transmit, receive).
   a. Running the image equalization in isolation yielded only ~50% of the theoretical throughput (240 MHz, i.e. 3662 fps), which puzzled us.
   b. By implementing a valid cycle counter at the entrance to the image equalization, we found that between the start and end of a transaction (frame), the throughput was ~98% of the theoretical, as expected.
   c. We believe the culprit for this slowdown is that it takes time from the DMA completion interrupt fires until the next transaction is commanded and starts. If this is the case, a work-around could be to instead command many transactions at once (e.g. a whole window of 256 frames) in what is called a “scatter-gather list”, which should be supported by the internals of the xilinx_dma driver.
   d. Since the DMA setup is specific to the demo (i.e. would not exist with an actual instrument), we have not experimented further to optimize this.
5. The Vitis AI C++ library is more functional than its Python wrapper, but it was fortunately simple to patch the wrapper to include what was missing. This saved us from having to reimplement the demo in C++.
6. On the ZCU104 board’s ZU7EV chip, Vitis AI unfortunately only supports 8 bit arithmetic, 8 bit CNN models and 8 bit images. Since our images are 16 bit, this required a work-around (in the spectral equalization). We cross our fingers that AMD/Xilinx will fix this as soon, as 8 bit image analysis in an operational setting would seem unacceptable.
7. The Vitis AI quantizer is not as advanced as one would hope. It only determines a power-of-2 scale factor for each CNN layer. An advanced quantizer should rather determine non-power-of-2 scale factors *per channel of each CNN layer*. We expect that this causes 1-1.5 bits of needless precision loss.
8. Vitis AI does not support buffer sharing between DPUs, nor buffer wrap-around (ring buffer). It was therefore necessary to programatically copy the last 54 equalized frames from the one DPU’s sliding window buffer to the beginning of the other DPU’s buffer, causing unnecessary performance loss. Again, we cross our fingers that AMD/Xilinx will fix this shortcoming soon.
9. Vivado does not support FPGA IP cores with top-level files in VHDL-2008. It was therefore necessary to write VHDL (-1993) wrappers for the image equalization and CCSDS 123 compression cores to be able to use them in the block design.

A flowchart of the implementation is shown in Figure 1. We note that the frame dimensions: 64 spectral rows; 1024 across-track columns, were chosen arbitrarily to resemble the test imagery used early in the development – 64 rows to resemble the 66 channels of the PRISMA satellite’s VNIR band (of which 3 are dead channels), and 1024 columns to resemble PRISMA’s 1000 columns (yet powers of 2 for implementation simplicity). Since the demo’s raw input imagery is prepared on the fly from Sentinel-2 and PlanetScope SuperDove images with much fewer channels, plain replication of channels to obtain 64 was used.

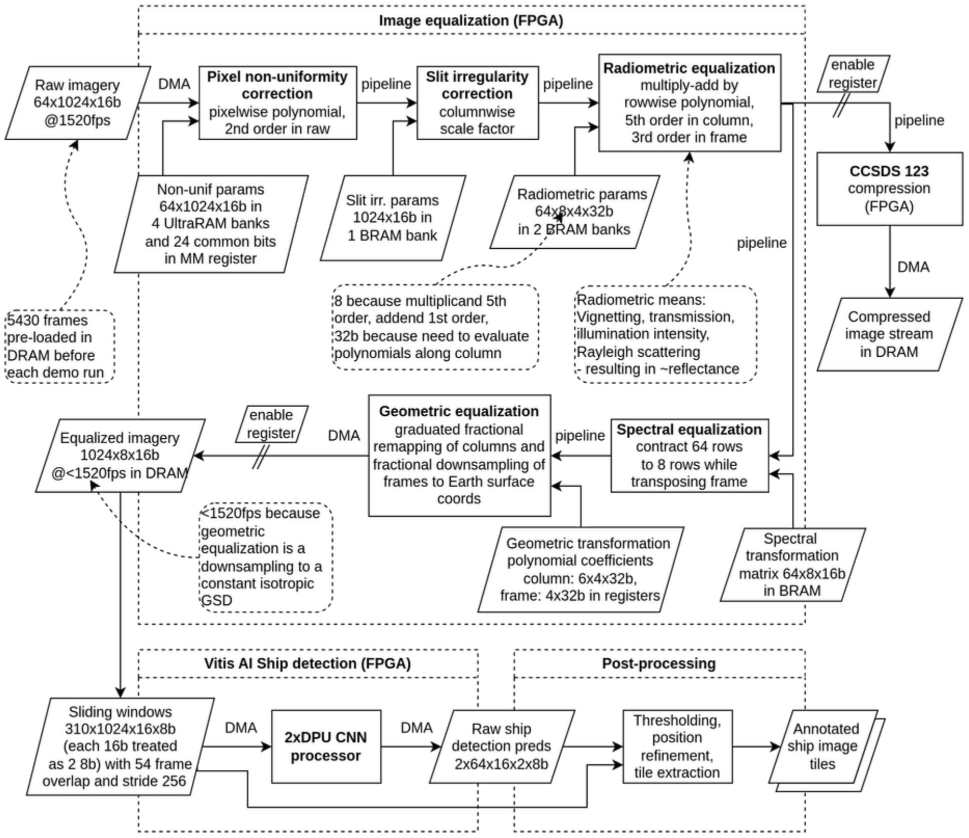
Image equalization (FPGA)
Raw imagery 64x1024x16b @1520fps
DMA
Pixel non-uniformity correction
pixelwise polynomial, 2nd order in raw
pipeline
Slit irregularity correction
columnwise scale factor
pipeline
Radiometric equalization
multiply-add by rowwise polynomial, 5th order in column, 3rd order in frame
enable register
pipeline
CCSDS 123 compression (FPGA)
DMA
Compressed image stream in DRAM
Non-unif params 64x1024x16b in 4 UltraRAM banks and 24 common bits in MM register
Slit irr. params 1024x16b in 1 BRAM bank
Radiometric params 64x8x4x32b in 2 BRAM banks
pipeline
5430 frames pre-loaded in DRAM before each demo run
8 because multiplicand 5th order, addend 1st order, 32b because need to evaluate polynomials along column
Radiometric means: Vignetting, transmission, illumination intensity, Rayleigh scattering - resulting in ~reflectance
Equalized imagery 1024x8x16b @<1520fps in DRAM
enable register
DMA
Geometric equalization
graduated fractional remapping of columns and fractional downsampling of frames to Earth surface coords
pipeline
Spectral equalization
contract 64 rows to 8 rows while transposing frame
<1520fps because geometric equalization is a downsampling to a constant isotropic GSD
Geometric transformation polynomial coefficients column: 6x4x32b, frame: 4x32b in registers
Spectral transformation matrix 64x8x16b in BRAM
Vitis AI Ship detection (FPGA)
Post-processing
Sliding windows 310x1024x16x8b (each 16b treated as 2 8b) with 54 frame overlap and stride 256
DMA
2xDPU CNN processor
DMA
Raw ship detection preds 2x64x16x2x8b
Thresholding, position refinement, tile extraction
Annotated ship image tiles

Figure 1: Flow chart of the implemented REMIS demo. In an actual deployment on board a satellite, the raw imagery would instead arrive from the instrument directly into the FPGA, typically over an LVDS connection, which would also serve to control and monitor the instrument (on/off, gain, frame rate, temperature readings, etc.). An additional connection would be needed to the main spacecraft computer to receive commands, orbit/attitude, software updates, detection model updates, parameter updates, and to dispose of ship image tiles, compressed images and logs destined for downlink. In addition, an “equalization parameter update” process would be needed that periodically updates the various parameters as instrument temperature, altitude, position, velocity, illumination and view change along the orbit – in particular the geometric parameters would need frequent update (~1 per second) since the ADCS continuously adjusts to maintain the desired pointing.

The FPGA components are implemented in VHDL-2008, the block design in Vivado Tcl, and the FPGA is operated at 240 MHz. The boot image is prepared with PetaLinux including 3 custom patches. The demo application [FR] is a very long Python script that interacts with the (patched) DMA driver, sets the image equalization’s parameters via memory-mapped IO and uses the (patched) Vitis AI libraries to drive the 2 DPUs. Test input imagery is loaded and prepared from files in JPEG2000 and GeoTIFF format stored on the SD card, equalization parameters are loaded from CSV files and the ship detection CNN is loaded from a file in the Vitis AI proprietary Xmodel format.

The CNN was trained on Vake’s database of labeled Sentinel-2 images, which contains 1000s of small image chips (some with, some without ships). The test images are whole Senitnel-2 images and PlanetScope SuperDove images. No training images appear in any of the test images. Since the test images are much wider than the frame, the demo divides each image into 42 partially overlapping strips 1024 columns wide and 5430 frames high. Note that the CNN, consisting of 18 layers, has a significantly simpler architecture than the models employed in Vake’s operational “ground segment”. This simplification was needed in order for Vitis AI to support the architecture and meet the timing requirements. Real time at 10m resolution is ~700 fps, thus the measured 1520 is actually comfortably faster. In the demo, it is anyway the image equalization and the copying between the window buffers that is the bottleneck, as mentioned. The simplicity of the CNN architecture together with the mentioned 8-bit quantization leads to a significant (but not severe) reduction in detection accuracy. Recent versions of Vitis AI are reported to support more advanced architectures, which should be explored.

3 Findings and Outlook

Although the REMIS project did not produce solutions to all the problems addressed (in particular points 1, 4, 6, 7 and 8 above), the project has demonstrated that on-board, real-time ship detection in satellite imagery should indeed be possible. The approximate L1 processing (our *image equalization*) implemented was designed to target hyperspectral imagery, which, however, is not ideal for ship detection. For ship detection, we would consider 3-8-channel multispectral with FOV several times wider than 1024px to be better suited. Since multispectral imagers don't have slits, but rater detector-deposited spectral filters and time-delay integration, somewhat different L1 processing would however be required. Moreover, the geometric equalization (orthorectification) may be redundant for ship detection, since ships already come in various sizes and shapes, and a geometrically distorted ship still resembles a ship (just a longer/shorter/wider/narrower one).

Several implementation time-saving assumptions were made in the image equalization, e.g. that all dimensions are powers of 2, and that there is enough pause between frames to carry out a parameter update. These assumptions would need to be eliminated (or verified) before it can become compatible with an actual instrument.

The following figures present examples of the accuracy of the the quantized ship detection model, overlaid the RGB reduction of the L1 test image processed. The detections were obtained by running the REMIS demo on the ZCU104.

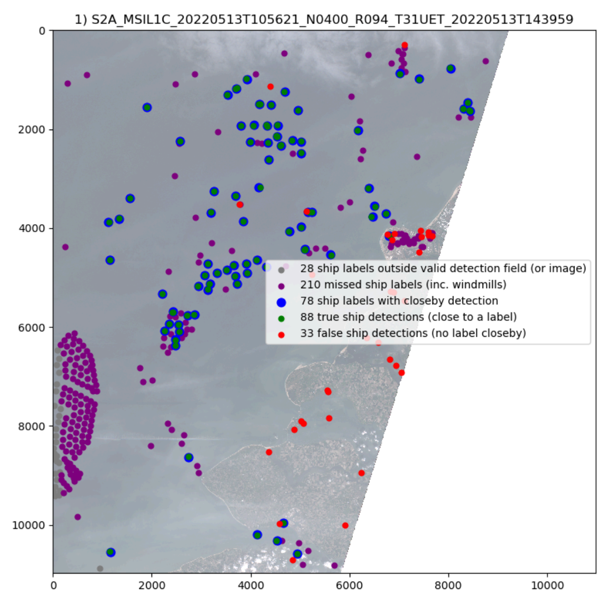

Figure 2: Ship detection accuracy in a Sentinel-2 test image of Rotterdam harbor with detection threshold set to 0. The background image is drawn with 50% opacity to increase contrast with the markings. The white area is outside the swath. The 6 seemingly false detections at sea are actually ships (missing labels). Many ships are missed, in particular inside the harbor, and we make several false detections on land. Although labeled, it is correct that the windmills on the left are missed. Lowering the threshold results in slighly more correctly detected ships, but also many more false detections on land.

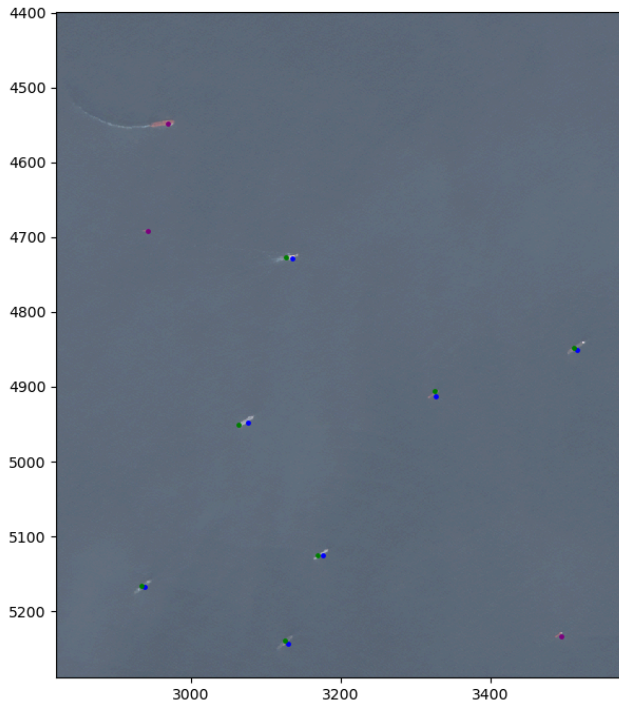

Figure 3: Zoom-in near the center of the image in Figure 2 showing 7 true ship detections and 3 missed ships. The background image is drawn with opacity 75%. Note that the detected location (green) tends to differ slightly both from the label (blue) and from the center of the ship.

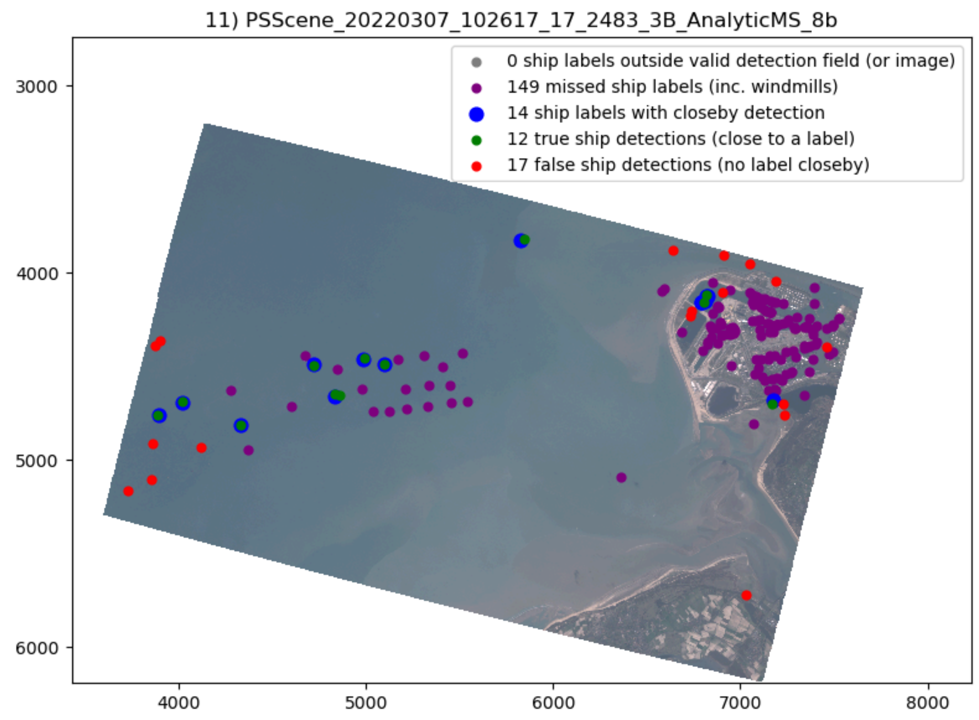

Figure 4: Ship detection accuracy in a PlanetScope SuperDove test image of Rotterdam harbor with detection threshold -30. Note that the CNN model has not been trained for this type of image (only Sentinel-2), and that the image is geometrically and spectrally resampled on the fly to resemble a Sentinel-2 image. The 9 seemingly false detections at sea are actually unlabeled ships. More than half of the ships are missed, though, so there is certainly room for improvement. The image is drawn with 75% opacity.

S&T and Vake are currently exploring possibilities for carrying out one or more IODs with the components developed in the project. Since ship detection CNNs seems to be in more demand than image equalization and compression, we are currently pursuing an IOD where we only provide a trained and quantized ship detection CNN model, for a host satellite which already has an instrument, approximate L1 processing and a CNN processor. Improving the model architecture and image adaptation (point 1) to achieve greater detection accuracy would be important challenges going forward. Besides this, we also hope to present the REMIS project at workshops.