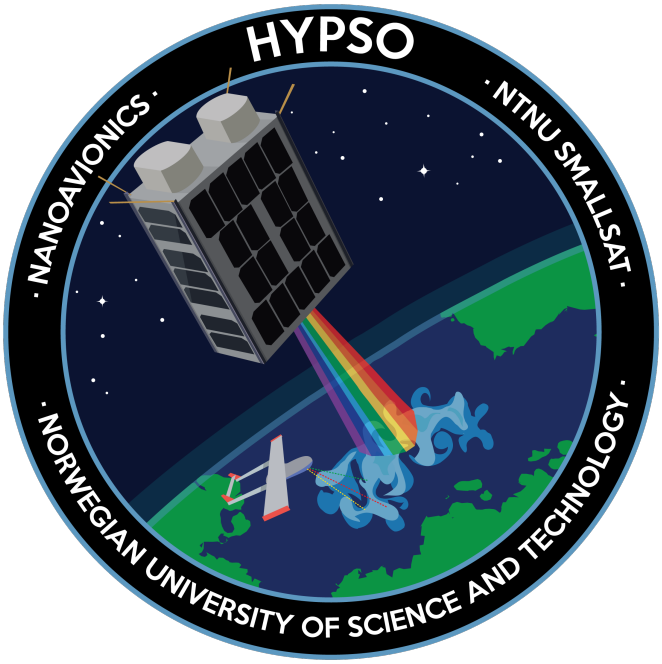
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Jonas Gjendem Røysland

# Real-time classification onboard the HYPSO-1 satellite

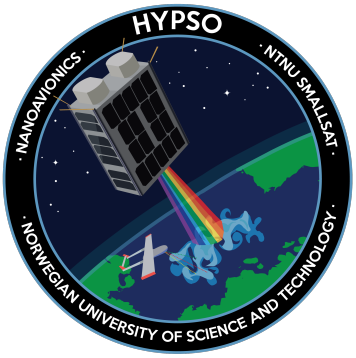
Master's thesis in Electronics Systems Design and Innovation
Supervisor: Milica Orlandic
Co-supervisor: Joseph Garrett
June 2023




NTNU
Norwegian University of
Science and Technology

Jonas Gjendem Røysland

# Real-time classification onboard the HYPSO-1 satellite




Master's thesis in Electronics Systems Design and Innovation
Supervisor: Milica Orlandic
Co-supervisor: Joseph Garrett
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology


# Real-time classification onboard the HYPSO-1 satellite

Jonas Gjendem Røysland

June 2023

Master Thesis

Department of Electronics Systems
Norwegian University of Science and Technology

Supervisor 1: Milica Orlandic
Supervisor 2: Joseph L. Garrett

# Abstract

This thesis undertakes the design and development of the classification module in the image processing pipeline onboard the HYPSO-1 satellite. The goal of the module is to detect different classes of land, water, and clouds on a hyperspectral image captured onboard the satellite. One reason for doing this is that a labeled image is 99.8% smaller than the original hyperspectral image and will make the downlink time down from minutes to seconds. This can help the ground station determine if the whole HSI capture should be downlinked or not. It also gives faster information if an image contains algae bloom that could be further investigated by autonomous agents. To make the classification possible it is implemented the machine learning algorithm Support Vector Machine with a Binary Decision Tree to predict the pixels. Since the images are captured on different areas of the Earth with different sun angles and exposure, radiometric calibration is developed to calculate the radiance and reflectance in the image. The module is first tested on a computer with a training set containing 10 different hyperspectral images for testing accuracy and execution time. After this, it is tested on target hardware to test the execution time, and behavior is as expected. Lastly, the implementation is uplinked and tested on the HYPSO-1 satellite. 4 HSI captures are labeled onboard and downlinked to verify the labeled image. From the labeled results it is discussed that it can be used to detect errors in the capture and send coordinates to autonomous agents for further investigations. It's concluded that labels give an indication of what is on the surfaces of the capture, but could have improvements in the data set for fewer labeling errors.

# Sammendrag

Denne oppgaven foretar seg designet og utviklingen av klassifiserings modulen i bildeprosseringen på HYPSO-1 satelliten. Oppgaven til modulen er å lage et klassifisert bilde som innholder klasser som land, vann og skyer ut i fra et hyperspektral bilde tatt på satelliten. Dette blir prossesert direkte på satelliten og gjort siden det klassifiserte bilde er 99.8% av størrelsen av det orginale hyperspektrale bildet. Dette gjør at informasjon om bildet ned til bakkestasjonen går fra minutter til noen sekunder. Dette kan være med å bestemme om bakkstasjonen ønsker å laste ned det hyperspektrale bildet eller la være. Det gir også raskere informasjon for hvilken koordinater autonome agenter skal sendes til for videre undersøkelse. For å gjøre klassifisering direkte på satelliten er det utvilket maskinlæringsalgoritmen støttevektormaskiner med binært beslutningstre for å forutse de ulike klassene. Siden de hyperspektrale bildene er tatt i ulike deler av verdene med ulike solvinkler og eksponering er det uviklet radiometrisk kalibrering for regne ut utstråling og refleksjonen i bildet. Modulen blir trent og testet først med et datsett med 10 ulike hyperspektrale bilder for å teste nøyaktighet og kjøretid på en datamaskin. Etter blir det testet på samme maskinvare som satelliten LidSat for sjekke kjøretid, men også om oppførsel er den samme. Til slutt blir modulen testet på HYPSO-1 satelliten der 4 klassifiserte bilder blir lastet ned og verifisert at modulen har oppførsel som forventet. Ut i fra resultatene kan de klassifiserte bildene bli brukt til å detektere feil i hyperspektrale bildene, men også til å gi indikasjon hvor de autonome agentene skal bli sendt til for videre undersøkelese. Forbedringspotensiale som kan gjøres videre er å forbedre datasettet som har blir brukt for å få mindre klassifiseringsfeil i bildene.

# Acknowledgment

I would like to thank my supervisor Millica Orlanic and co-supervisor Joseph L. Garret for the many meetings we had discussing the project. Also giving me guidance, resources and technical input throughout the master thesis. This thesis would not be possible without you.

Also like to thank, Dennis Langer and Simen Berg helping me test my implementation on target hardware Lidsat at SmallSat NTNU. And to the most importance, let my implementation by tested directly on the HYPSO-1 satellite and downlinking the results I needed for the thesis.

Thanks to the people on the software space segment sprint meeting for coming up with the critique of my implementation and pushing for a better workflow throughout the master thesis. Special thanks to Sivert Bakken and Roger Birkeland, for leading the meetings and also coming with expertise in software development.

Thanks to all the past and current members of the HYPSO team for making it possible to test cool stuff onboard a satellite in space. It wouldn't be possible to test this stuff without years of hard work.

Lastly, thanks to all my fellow students at Orbit NTNU for all the technical knowledge I have gained in software development and space technology. The social stuff, good discussion, memes, and the fun we had the roof.

# Contents

# Acronyms

**ASV** Autonomous Surface Vehicles. 6

**AUV** Autonomous Underwater Vehicles. 6

**BDT** Binary Decision Tree. 11, 23, 24, 27, 28, 32–36, 38, 40, 42, 44, 47, 50, 51

**ELO-Hyp** Efficient Learning and Optimization Tools for Hyperspectral Imaging Systems. 24

**FPGA** Field Programmable Gate Array. 6

**GCC** Gnu C Compiler. 8

**GT** Ground Truth. 17, 19, 35, 38, 39, 41, 42, 44, 51

**HAB** Harmful Algae Bloom. 3, 5, 11, 21, 50, 52

**HSI** Hyperspectral Image. iv, 1–3, 5, 7, 9, 12, 14, 16–21, 25–31, 35, 36, 44, 46, 48, 50–52

**HYPSO** Hyper-Spectral Small Satellite for Ocean Observation. 1, 2, 4, 6, 9, 21, 32

**IoT** Internet of Things. 21

**LEO** Low Earth Orbit. 1, 6

**MDC** Minimum distance classification. 10

**ML** Machine Learning. 3, 7, 9, 12, 19, 51

**MLC** Maximum likelihood classification. 10

**NTNU** Norwegian University of Science and Technology. 1, 6, 7

**OPU** Onboard Processing Unit. 6, 7, 27, 48

**OS** Operative System. 8

**RGB** Red, green and blue. 17, 44

**SDR** Software Defined Radio. 6, 21, 52

**SVM** Support Vector Machine. 9–13, 19, 23, 28, 35

**SVMBDT** Support Vector Machine with Binary Decision Tree. v, 9, 17, 23, 24, 28, 35, 51, 52

**UAV** Unmanned Aerial Vehicles. 5, 6, 21

**UHF** Ultra High Frequency. 6

**UN** United Nations. 3, 50

**WHISPERS** Workshop for hyperspectral image and signal processing: Evolution in remote sensing. 3, 58

# Chapter 1

# Introduction

In recent years small satellites have become more affordable to launch into space. And within the next decades, it is expected the amount of satellites in Earth's orbit largely increases [12]. In some areas, they are used for global communication, space observation, and global monitoring like remote sensing. Remote sensing is about measuring and capturing reflected sunlight on the Earth's surface. One of the subjects that could help monitor environmental and climate change when doing remote sensing is harmful algae blooms [31]. By observing these areas environmental changes can be observed.

## 1.1 Motivation

One of the organizations having the objective of doing observations of algae booms is Hyper-Spectral Small Satellite for Ocean Observation (HYPSO) team at SmallSat Labrotatry at NTNU in Trondheim, Norway. HYPSO is currently operating the satellite HYPSO-1 which has an orbital altitude of 540km, Low Earth Orbit (LEO)[18]. To monitor the algae bloom, a hyperspectral imager onboard the satellite a capturing these areas. The images are called Hyperspectral Image (HSI) that have 120 spectral bands in them, making it easier to separate different surfaces from one another. Due to the high amount of spectral bands, the HSI has a size of 151MB and 78.6MB when compressed. The compressed HSI usually takes 2-3 orbital passes, when the satellite is in the local horizon of the ground station in Trondheim. Where each orbital pass is between 7-10 minutes, whereas orbit times around the Earth are around 90 minutes [6]. An illustration of how the HYPSO-1 is operated with the ground station is viewed at Figure 1.1. Sometimes the HSI captures only clouds or has pointing errors. This makes the HSI useless when wanted to be further analyzed. If the ground station knew this before the image was downlinked, other HSI captures could be prioritized.

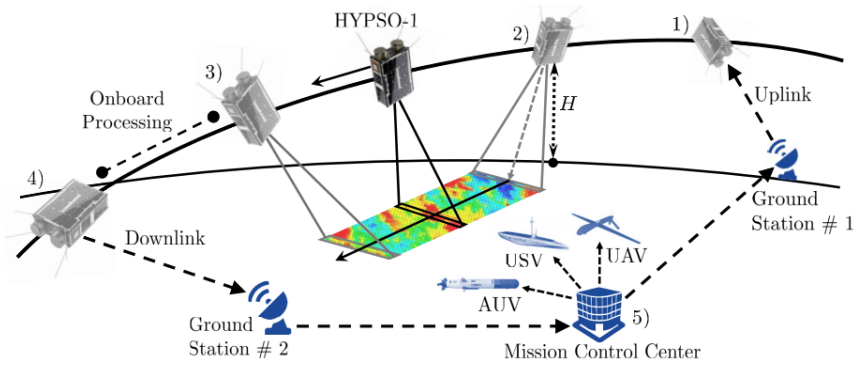



Figure 1.1: CONOPS for HYPSO-1, where HYPSO-1 captures a HSI and then downlinks it to ground station [18].

One solution to give the ground station faster information on the HSI capture is producing a labeled image onboard that can be downlinked. The pixels in the labeled image would contain what type of surface the HSI had captured with different classes. The labeled image will have a size of 327kB with up to 16 classes. The downlink speed between HYPSO-1 and the ground station is 1Mpbs, making the downlink time of the labels 2-3 seconds [18]. The ground station would then get fast information if the HSI capture should be downlinked or not. An illustration of a HSI capture is classified and labeled onboard the satellite is viewed in Figure 1.2.

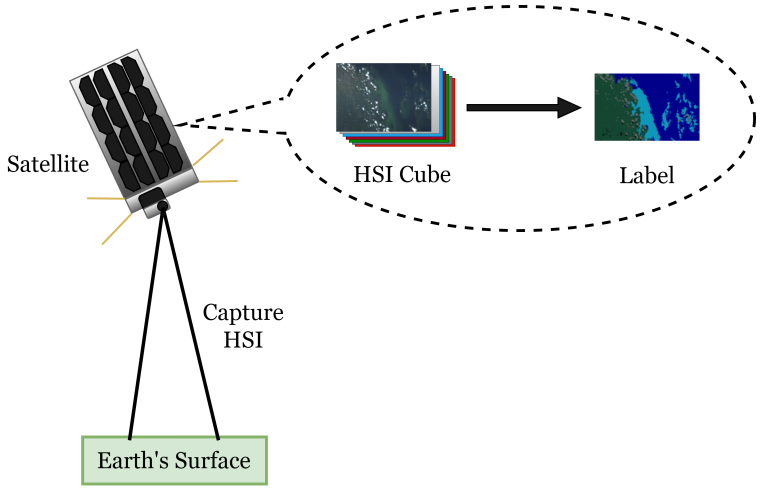



Figure 1.2: Illustration of HSI capturing the Earth's surface and after processing it to a labeled image onboard the satellite.

The thesis will start by introducing remote sensing, the HYPSO organization, and what factors must be considered when designing onboard classification. After this show what theory and design are chosen for the classification and necessary calibration for the HSI.

## 1.2 UN's sustainability goals

In this section, it will be looked at how HSI and the labeling of these images can benefit the United Nations (UN)'s sustainability goals.


(a) Goal 6 (b) Goal 14 (c) Goal 15

Figure 1.3: UN's sustainability goals 6, 14 and 15.

Harmful Algae Bloom (HAB) is often occurring in freshwater and brackish water systems and occur where water is warm, still, and nutrient-rich. The algae blooms can harm the health of humans and animals. Due to the rising global temperature, these algae blooms can occur more often. Since their lives fish in these waters, they often carry these HAB when humans and animals are eating them [33].

By labeling and mapping where HAB is located in the HSI, it could give a better understanding of which areas and the time stamp these are located in the water. By doing this local areas could be noted of the HAB, and help achieve the goals 6, 14, and 15. These goals are shown in Figure 1.3.

## 1.3 Contributions

In this thesis, it's implemented a classification module onboard the HYPSO-1 satellite and is the first Machine Learning (ML) algorithm to be executed onboard the satellite. Making it possible to see that a ML algorithm can be executed onboard a satellite. The classification module will be further executed and tested after the delivery of the thesis, to see what other areas the labeled HSI can be used for.

In the classification module, it is also implemented radiometric calibration onboard the satellite. This can also be used for future projects where calibration of HSI is necessary.

For training the ML model in the classification module, 10 HSI captures have been labeled for this thesis. They are produced since a desired data set that could train a ML algorithm to work for every HSI captures, has not been produced before. This data set could also be used to train other ML algorithms for other future projects.

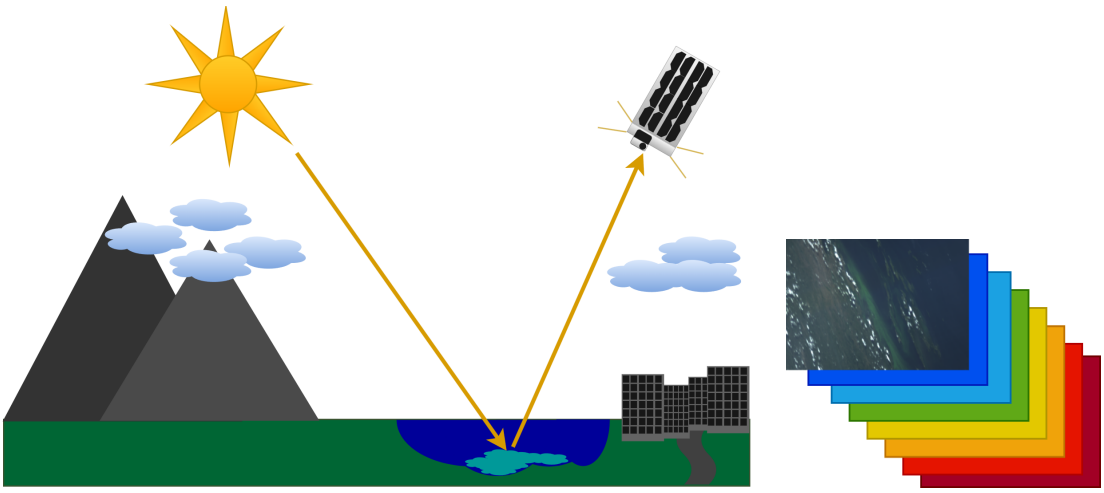
Together with the master thesis, it has been produced a paper for the 2023 WHISPERS conference. The proceeding paper submitted is found at Appendix B and submitted to [36].

# Chapter 2

# Remote Sensing, HYPSO and Classification

This chapter will be explained what remote sensing is and what hyperspectral images are used with it. Further, explore the HYPSO organization and the satellites. Also, explain what kind of design decision needs to be made when implementing onboard classification for the HYPSO satellite.

## 2.1 Remote Sensing and Hyperspectral Images



(a) Remote Sensing (b) HSI Cube

Figure 2.1: (a) Illustration of a satellite using solar reflectance to capture the Earth's surface. (b) Illustration of an HSI cube having multiple spectral bands in the electromagnetic spectrum.

Remote sensing is about monitoring the characteristics of the Earth's surface. For doing the remote sensing satellites orbiting around the Earth can be used for global scales monitoring. For local scales platforms such as Unmanned Aerial Vehicles (UAV), can give more detailed characteristics of the surfaces. Multiple viewpoints from different platforms can give an accurate characteristic of agriculture like growth in crops or HAB [35]. An illustration of how a satellite monitors the surface of the Earth with reflected sunlight is shown in Figure 2.1 (a).

Remote sensing in recent years HSI, has become more popular due to the smaller sizes and cheaper cost of the hyperspectral imagers. Some of the spacecraft that have been launched into orbit around the Earth are EnMap, DESIS, HyspIRI, PRISMA, and HYPSO [22]. The reason for using the HSI in remote sensing is due to its narrow bands that are represented in the electromagnetic spectrum. Each pixel that is captured in an image will have an accurately reflected spectrum in the wavelength range. The HSI is represented as cubes that contain many spectral bands in them, giving it a high spectral resolution. This gives great spectral information about the different surfaces that are captured, making it easier to distinguish objects from one another [23]. An illustration of an HSI cube is shown in Figure 2.1 (b).

## 2.2 HYPSO

HYPSO is an organization that is part of the Small Satellite Lab at NTNU in Trondheim. It consists of post. docs, PhD-student, and bachelor and master students[18]. HYPSO's first satellite HYPSO-1 was launched from Florida with the SpaceX rocket Falcon-9 on January 13th, 2022, into sun-synchronous LEO. Onboard the satellite is a hyperspectral imager, that aims to collect data of oceanographic observations to support marine research. The data collection aims to support autonomous agents such as UAV, Autonomous Surface Vehicles (ASV), and Autonomous Underwater Vehicles (AUV) [6].

For HYPSO's next satellite HYPSO-2, it will have the same concepts as HYPSO-1 but also include a Software Defined Radio (SDR). The SDR is sending radio frequency in the Ultra High Frequency (UHF)(400 MHz) bands, that will enable direct communication with the other autonomous agents [9]. In section 4.3 it will be further explored how a labeled HSI could be used to determine the coordinates for the autonomous agents.

## 2.3 HYPSO satellite specifications

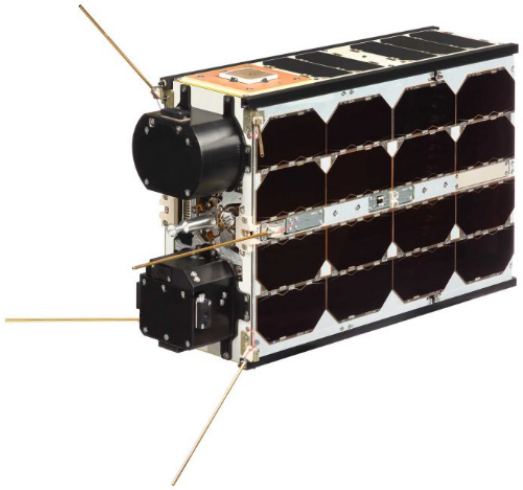


Figure 2.2: M6P multi-purpose nano-satellite bus [24].

HYPSO-1 is a 6U CubeSat, where 1U is 10cm×10cm×10cm. It is a 6U Platform (M6P) commercial available nanosatellite bus provided by NanoAvionics, shown in Figure 2.2. The satellite has an Onboard Processing Unit (OPU) that consists of two core ARM processors and a Field Programmable Gate Array (FPGA) that is dedicated to the onboard processing. The OPU makes it possible to do software updates, even if the satellite is in orbit [18]. The memory the OPU has is 1GB ram, where 50% of this is reserved [13]. The radio communication between the satellite and

ground station is a 2.4GHz IQ Spacecom S-band Transiever that can downlink data with a rate of 1Mpbs [18]. When the hyperspectral imager is capturing HSIs the pixels contain 120 spectral bands with the range 430 nm to 800 nm in the electromagnetic spectrum [6].

Before new software is uplinked to the satellite, this needs to be tested on the same hardware as the satellite. On NTNU Small Sat Lab there is a setup called LidSat, that enables this testing [5].

## 2.4 Classification onboard the satellite

To process the HSI onboard the satellite, an image processing pipeline is implemented. A purposed pipeline for HYPSO-1 is shown in Figure 2.3. Many of these modules are already implemented onboard but are not actively working classification modules, which are marked in red. In this thesis, it will be done design decisions, implementation, and testing of this classification module, to be actively executed onboard the satellite.

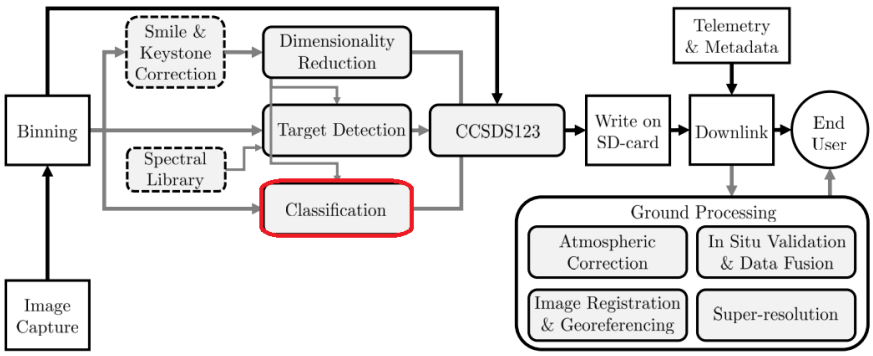



Figure 2.3: Proposed onboard image processing pipelines for HYPSO-1 [18], where the classification module is marked with red.

When classifying of HSI onboard, pixel-wise ML techniques are widely used. The benefit of these is that they have high accuracy, but also low memory usage [3]. Due to the limited memory available on the OPU, the classification module needs to use less than 50% of 1GB. The ML algorithm also needs to have a low classification time so that the labeled image can be downlinked within one orbital pass.

The project thesis written in the fall of 2022 [28] it was designed and tested a Support Vector Machine with Binary Decision Tree that had a low classification time and high overall accuracy when labeling HSI, compared to other MLs. This algorithm will be further developed in this thesis to be implemented in the classification module.

## 2.5 Software and GCC

The Operative System (OS) on the OPU is an Embedded Linux System that is called PetaLinux [7]. For making applications for the imaging pipeline the programming language C is used. This is used due to better memory control and faster execution time when the application is executed. To make the program be executed on the target hardware, a compiler is used. For the programming language C, the compiler Gnu C Compiler (GCC) is the most popular of choice. The achieved goal of the program is that has the fastest execution time possible, but still have the behavior as desired. To make the program have a faster execution time an optimization flag is provided by GCC. The flag that is found that is reducing the execution time the most is the -O3 flag [15]. This will be further explored at chapter 6 when building the classification module with and without -O3.

# Chapter 3

# Theory and algorithms

Will in this chapter explore the theory of the different algorithms for the classification module onboard processing pipeline on the HYPSO satellite. The first is the ML algorithm Support Vector Machine with Binary Decision Tree (SVMBDT). This will do the classification and make the labeled HSI. Due to the HSIs being captured around the Earth with different solar radiance and reflectance, radiometric calibration is introduced. This will be explained how this can benefit the SVMBDT when doing the classification.

## 3.1 Support Vector Machine

For classifying with ML algorithms, it's called the supervised learning approach. This means analyzing a given data set and making a model that can separate different data into different classes that are desired. One of these ML algorithms is called Support Vector Machine (SVM), which is a kernel-based ML model that is a powerful tool to solve practical binary classification problems. When doing the separation there is a maximum margin hyperplane in the higher dimensional space called feature space [10]. In the Figure 3.1 linear separation in feature space is illustrated.

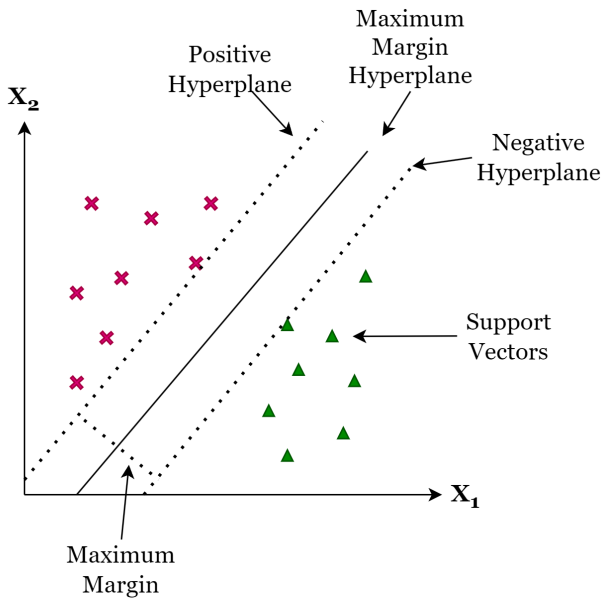



Figure 3.1: Illustration of SVM in feature space separating binary of two classes.

From the figure above, the linear separation has a maximum margin that is the goal of maximizing the distance between the two classes. The two classes are determined then above the positive hyperplane (1) and below the negative hyperplane $(-1)$. This is mathematically shown in the Equation 3.1.

$$w \cdot x + b = \pm 1, \tag{3.1}$$

where $w$ is the weights, $x$ is the input data and $b$ is the intercept [10].

When the classification has multiple features it is shown as:

$$\sum_{i=1}^{n} w_i \cdot x_i + b, \tag{3.2}$$

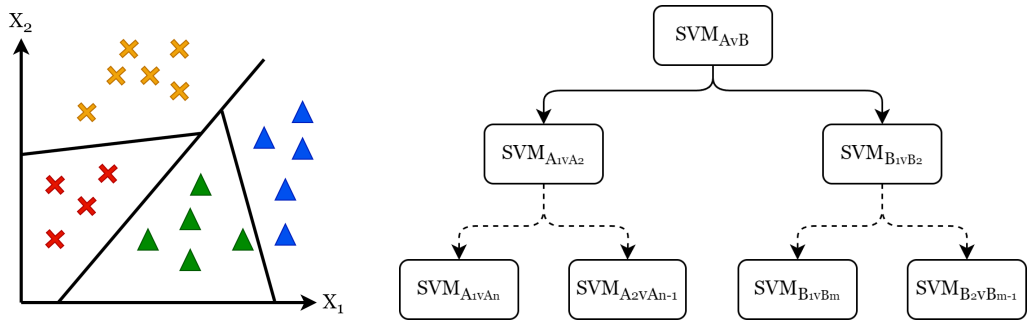
where $n$ is the number of features.

The SVM is trained with data set with a $m$ number of samples. The data set contains input vectors $x_i$ together with associated label vectors $y_i$. The $y_i$ contains the class number, and $x_i$ is the input data with $n$ features. The data set is then given as:

$$(x_0, y_0), (x_1, y_1), ..., (x_m, y_m) \tag{3.3}$$

The benefit of linear SVM compared to other machine learning algorithms is that the time complexity is $\mathcal{O}(n)$ compared to Minimum distance classification (MDC) with $\mathcal{O}(n + \log n)$ and Maximum likelihood classification (MLC) with $\mathcal{O}(n \log n)$ [27].

### 3.1.1 Binary Decision Tree




(a) Feature space (b) BDT General Structure

Figure 3.2: Illustration of SVMBDT in (a) separating two classes and then two subclasses in feature space with the BDT. (b) BDT general structure, where A and B are the main classes with n and m amount of subclasses.

Since SVMs are doing binary separation, this only separates two classes. To have the possibility of doing multiple classes a decision function needs to be utilized. The two most popular decision functions for SVMs are one-vs-one and one-vs-rest [2]. The one-vs-one is comparing all the classes to each other, and the one-vs-rest is comparing one class against all the other classes. One other decision function for SVMs also exists and it is called Binary Decision Tree (BDT). The BDT is designed to have one or more classes separating each other for every layer. This makes it have $M-1$ SVM classifier in one tree structure [21]. The tree structure can be designed for different scenarios. Making it possible to have classes and subclasses separated from each other that have similar attributes. One example of this is separating clean water from water with a HAB. The design of the tree is also affecting the execution time when predicting. If the tree is designed with a class with a long travel path the execution time will proportionally increase. The design structure of the BDT will be further explored in chapter 6.

### 3.1.2 Sparse SVM

When the HSI cubes are captured onboard the satellite 120 spectral bands in every pixel. The number of spectral is for the SVM several features $n$ and will affect the computational time when calculating Equation 3.2. This makes the execution time ($t_{\text{execution}}$) proportional with the number of bands ($n_{\text{bands}}$).

$$t_{\text{execution}} \propto n_{\text{bands}} \tag{3.4}$$

To restrict the number of bands used, Sparse SVM can be utilized to do band selection. For doing the sparse selection the $L_1$-Norm is restricting the number of bands [16]. These spectral bands that are selected are then only considered when calculating Equation 3.2.

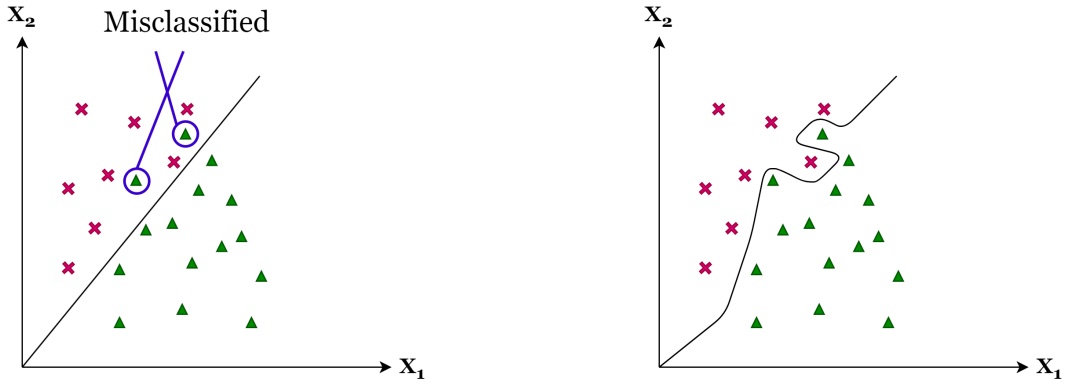
### 3.1.3 Using SVMs in real-time systems.

The benefit of using Linear SVM is its fast prediction making it have lower power consumption but also has high accuracy with few training data compared to other ML algorithms. Often real-time systems have a limited power budget and the algorithms need to be fast. Using SVM for a real-time system can often have those requirements that are needed for these real-time systems.

Some real-time systems that use SVM are:

- For tracked mobile robots to classify real-time terrain estimation to improve autonomous navigation [32].
- Detection system of scratches, cracks, or tears on conveyor belts on visual saliency [19].
- For the usage to diagnose breast cancer [20].
- Intelligent fault diagnosis method for Lithium-ion batteries [37].

### 3.1.4 Pros and cons with Linear Kernel




(a) Misclassification of Linear SVM with overfitting.

(b) Correctly classified with Non-Linear SVM not causing overfitting

Figure 3.3: Illustration of linear vs non-linear SVM in feature space.

For SVM there are two types of kernel functions defined as linear and non-linear, that separate in the feature space. The linear that is shown Equation 3.2 and non-linear kernel as the RBF (Radial Basis Function) that is shown in Equation 3.5 [4].

$$k(x_i, x) = exp(-\gamma |x_i - x|^2) \tag{3.5}$$

Generally, kernel SVMs have pretty high computational complexity when doing execution time. When chosen for a real-time system the non-linear kernel as the RBF would high a higher execution time than the linear one, due to the more complex separation. The cons with the linear kernel are that has a higher chance of causing overfitting when doing non-linear separation in feature space. This is where non-linear has the benefit of linear making them less likely for overfitting and misclassification when labeling HSIs [25]. The difference between the kernels is illustrated in Figure 3.3.

## 3.2 Radiometric calibration

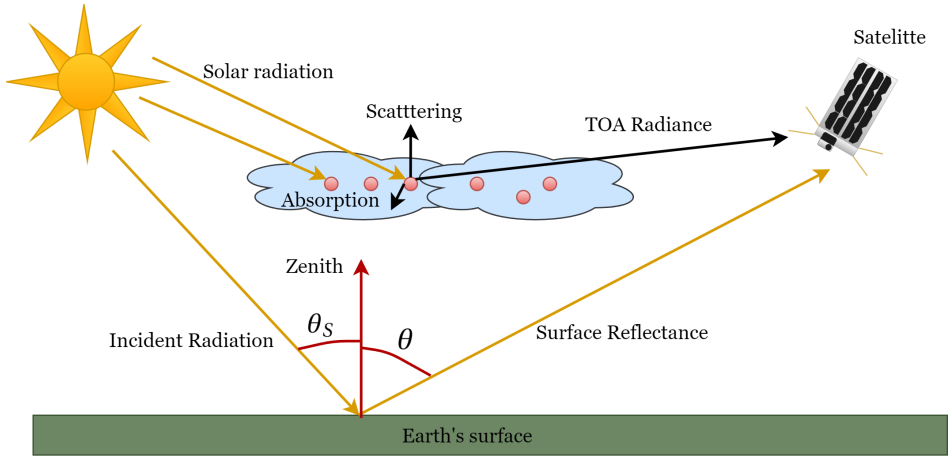



Figure 3.4: Illustration of the difference between radiance and surface reflectance. Where reflectance calibration is dependent on the solar zenith angle $\theta_S$.

When capturing an HSI with a satellite the main method the imager takes is by push-broom imaging. This method works by making a linear array perpendicular to the flight of the satellite. When this is done continuously over an area, you will get an HSI cube [34]. This same method applies also to HYPSO-1 [7]. The downside of this method, it can occur striping errors in the image, that are parallel with the push-broom direction. One other factor when capturing an HSI is the gain and exposure value from the imager. These errors and factors can affect the accuracy of the classified labeled image when capturing HSI around the Earth's surface [34].

A method that takes this into account is radiometric calibration. The purpose of radiometric calibration is to translate raw digital numbers ($DN$) to physical units that measure the radiance and reflectance intensity at the sensor [11]. The two methods of radiometric calibration are described in the sections below and are illustrated in the Figure 3.4.

### 3.2.1 Radiance

Radiance is described as the radiant flux emitted, that is transmitted by a surface in a given projected area. It is measured as watt per steradian square meter ($W \cdot sr^{-1} \cdot m^{-2}$) [8]. This is calculated by the Equation 3.6.

$$L(\lambda, x) = [DN(\lambda, x) - \alpha(\lambda, x)]\beta(\lambda, x), \tag{3.6}$$

where $L(\lambda, x)$ is the function of wavelenght $\lambda$ and x is the cross-track spatial location. $\alpha(\lambda, x)$ are the offset and $\beta(\lambda, x)$ are the gain in the $DN$s [11].

$$\alpha(\lambda, x) = \alpha_{dark}(\lambda, x) + \alpha_{ped}(x), \tag{3.7}$$

where the $\alpha_{dark}(\lambda, x)$ and $\alpha_{ped}(x)$ are the dark current and pedal shift respectively [11].

### 3.2.2 Reflectance

The reflectance is measured as the amount of solar light reflected on the Earth's surface. It is calculated by the Equation 3.8.

$$R_{sensor} = \frac{L(\lambda, x) \times \pi \times d^2}{E_{SUN} \times cos\theta_S} \tag{3.8}$$

$R_{sensor}$ is the reflectance, $d$ is the distance between the Earth and the Sun. $E_{SUN}$ is the mean solar exo-atmospheric irradiance, and the $\theta_S$ is the Solar zenith angle [6].

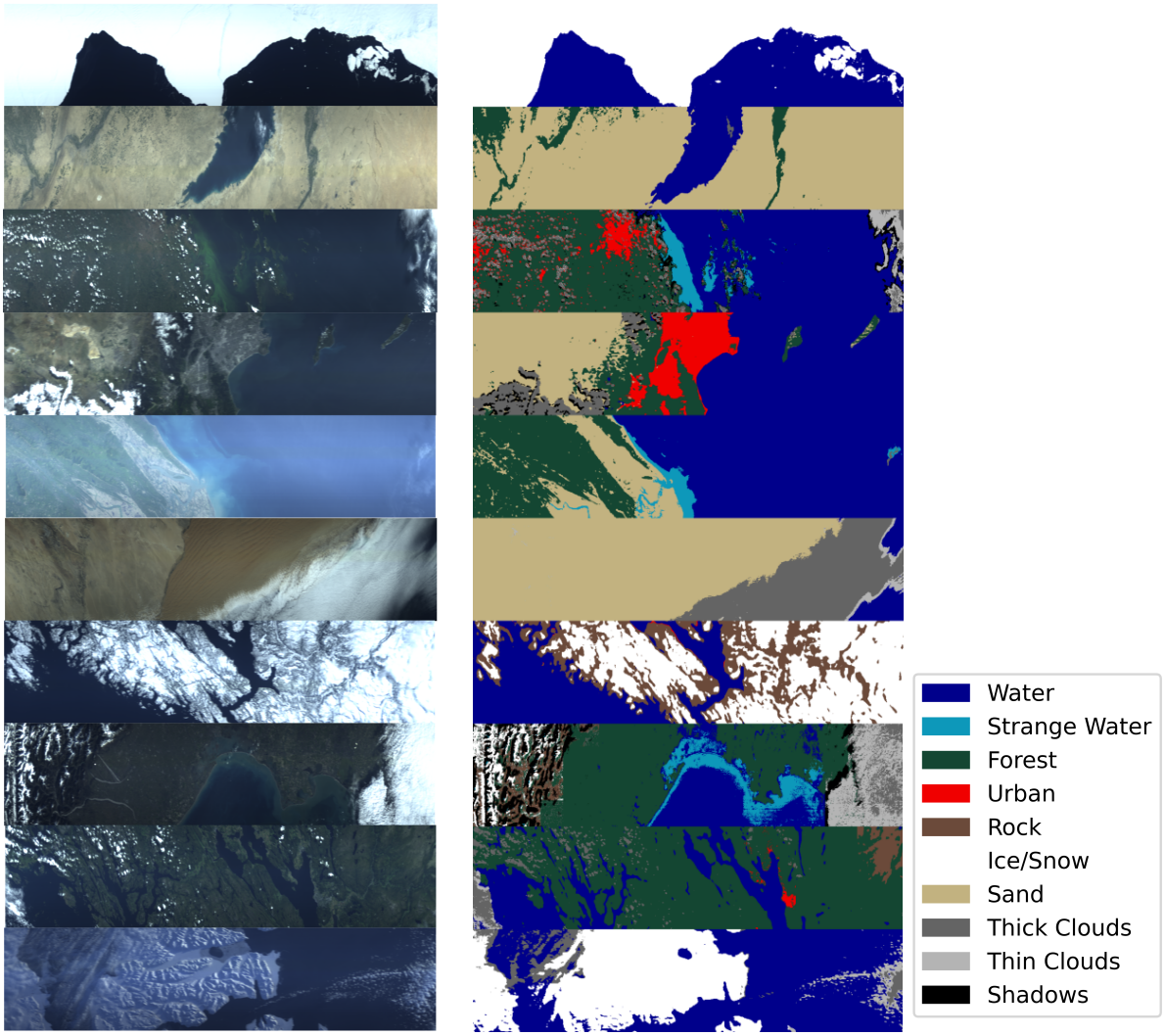
# Chapter 4

# Hyperspectral dataset and usage of the labels

In this chapter, it will be represented the HSI dataset used for training the onboard classification, and what challenges that occur when making these labels. Also, discuss what predicted labeled data can be used for working with autonomous agents and detecting errors in the capture.

## 4.1 HSI Dataset

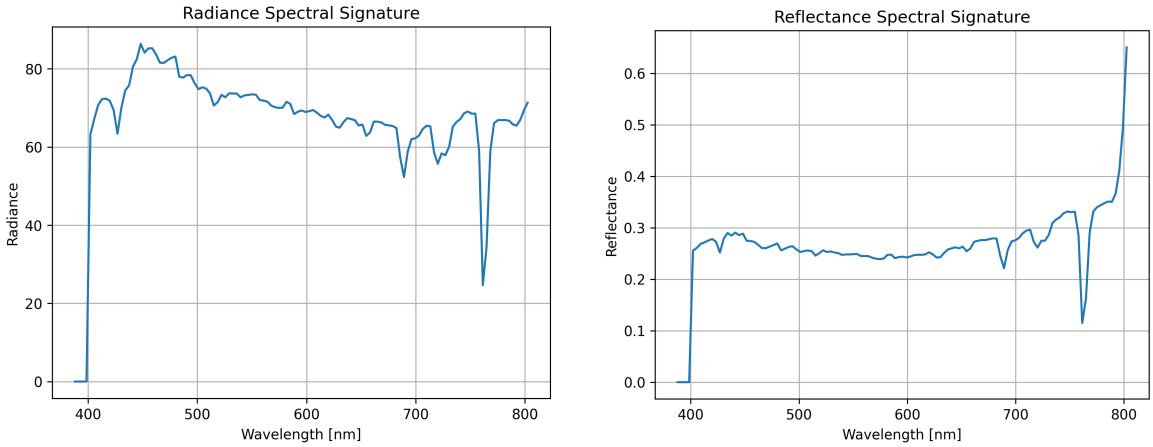
In machine learning, a dataset need to be in place when training and making parameters for the models. For the SVMBDT discussed in section 3.1 the samples are formatted as the input vector $x_i$ and associated labels vector $y_i$. For this project, a dataset with 10 HSI captures from the HYPSO-1 satellite with Ground Truth (GT) labels is produced. GT is telling what kind of surfaces are in each pixel of the HSI. The dataset is viewed at Figure 4.1.




(a) (b) (c)

Figure 4.1: (a) The 10 HSI captures represented in Red, green and blue (RGB). (b) GT labels of the 10 HSI. (c) Color mapping of the 10 different classes found in the labels.

For each of the 10 HSI cube, they are all radiance and reflectance radiometric calibrated. They are calibrated with the implementation found in subsection 5.2.1. The spectral signature of the dataset with radiance and reflectance is viewed in Figure 4.2.




(a) Radiance (b) Reflectance

Figure 4.2: The spectral signature of the training data from Figure 4.1, with (a) radiance and (b) reflectance calibrated.

The captures are taken on different surfaces of the Earth and timestamps. The date and the coordinates of the HSI are viewed at Table 4.1.

| **Area** | **Capture Date** | **Coordinates** | |
|---|---|---|---|
| | | **Latitude** | **Longitude** |
| Antarctica | 11/12-2022 | 74°40’20.0"S | 23°19’50.2"W |
| Tharthar, Iraq | 04/03-2023 | 35°29’28.0"N | 43°33’35.2"E |
| Kampala, Uganda | 04/02-2023 | 1°45’59.7"N | 32°50’42.6"E |
| Los Angeles, USA | 24/03-2023 | 35°28’04.0"N | 118°01’13.2"W |
| Sittwe Lake, Myanmar | 07/02-2023 | 21°35’32.4"N | 92°52’39.1"E |
| Gobabeb, Namibia | 14/02-2023 | 21°45’13.6"S | 14°50’37.9"E |
| Trondheim, Norway (Snow) | 27/03-2023 | 65°05’12.1"N | 10°47’24.6"E |
| Venice, Italy | 08/02-2023 | 46°50’05.5"N | 12°57’58.2"E |
| Trondheim, Norway | 23/08-2022 | 65°05’10.9"N | 11°08’38.5"E |
| Svalbard, Norway | 14/03-2023 | 78°33’55.5"N | 20°28’24.4"E |

Table 4.1: Different areas, capture date, and coordinates of the training HSI images in chronological order.

The classes in the training data consist of 10 classes. These consist of surfaces that are all over the Earth’s surface and are captured by HYPSO-1. These classes are described in the Table 4.2.

| Number | Class Name | Description |
|---|---|---|
| 0 | Water | Ocean water, rivers and lakes. |
| 1 | Strange Water | Water with algae bloom, pollution or are shallow. |
| 3 | Forest | Three forests and land vegetation. |
| 4 | Urban | City surface, roads, and buildings. |
| 5 | Rock | Mountains and rocky surfaces |
| 6 | Ice/Snow | Ice glaciers and snow-covered surface. |
| 7 | Sand | Sand deserts and beaches. |
| 8 | Thick Clouds | Dens clouds covering the Earth's surface. |
| 9 | Thin Clouds | Thinner clouds, where the Earth's surface can be viewed. |
| 10 | Shadows | Shadows produced by clouds or mountain terrain. |

Table 4.2: The number and description of the classes in the training data set.

### 4.1.1 Process for labeling training data

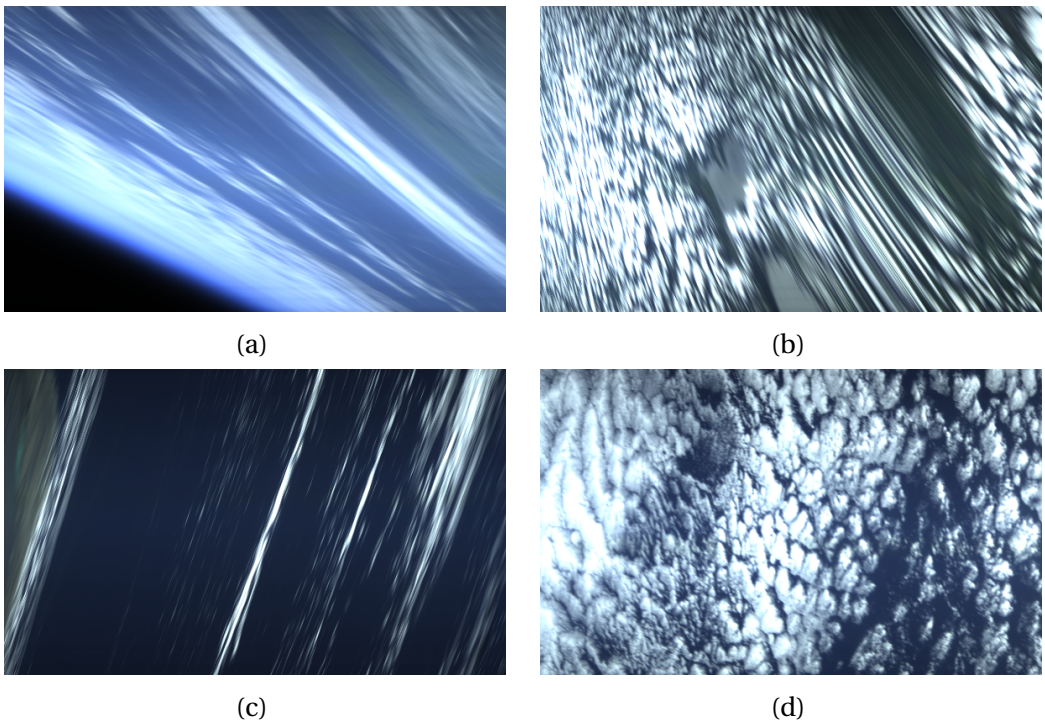
When machine learning algorithms are trained, a training set needs to be in place for the algorithms to make the right classification decisions. In subsection 3.1.4 it was introduced how linear separation could make the SVM algorithm make misclassifications. One other causes of misclassifications are errors in the training data. The errors that are in the data set could affect the MLs when training and will make a higher probability of false labeling in the images.

The labels can often be produced by using a supervised MLs together with a human by looking at the captured image and a map of the same area. One example of this is the data set shown in section 4.1. Some errors that can be viewed are in the Venice label, where no land surface is labeled as urban. Also note that the Trondheim (Snow) label doesn't have urban labels, due to the city surface covered with snow in that HSI. This is compared to the Trondheim labels.

When the labels are produced this way more errors can occur, due to not knowing the local knowledge of the capture area [14]. The solution for making fewer errors when producing the GT is by using one or more of the autonomous agents together with a satellite. This will make the labeling easier, due to more accurate analysis of a given area.

## 4.2 Catching errors in the HSI captures

The HYPSO-1 satellite has captured and downlinked over 1000 HSI since it was launched in 2022. Many of the captures have been successful, but some have errors in them. These errors can be capturing just clouds in the image, which are blocking the Earth's surface. Other errors are either pointing errors or the focus making the HSI distorted. Some examples of this HSI captured by HYPSO-1 can be viewed at Figure 4.3



(a) (b)

(c) (d)

Figure 4.3: Some captured HSI with a)-c) error distortions, and d) clouds blocking most of the surface area.

By producing labels and downlinking them before the whole HSI capture, the ground station can determine if the HSI should be downlinked or not. They can determine this by the shape and the classes produced in the labeled image. Some unsuccessful images that are labeled either with errors or with a lot of clouds in them are later explored in the subsection 6.2.4 and section 6.4.

## 4.3 An observational pyramid with a drone

When the HYPSO-1 satellite takes HSI over large areas, around an area of 100km × 100km is covered. Due to the spatial resolution of the image, the local area captured would not have a lot of details in them [6].

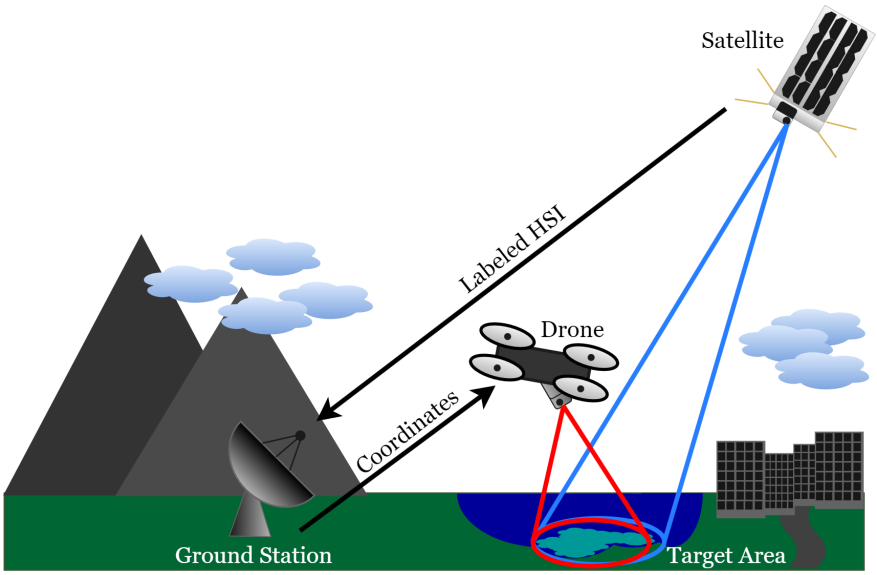



Figure 4.4: Satellite downlinking HSI label with georeferencing to determine which coordinate a drone should be sent for taking images for more detailed areas.

To have more detailed imagery and analysis of an area a drone or UAV can be deployed [6]. For the HYPSO project, HABs are target surfaces that want to be further analyzed. To detect the HABs the classified labeled HSI can be used, where some of the water are classified with the strange water class. Due to the label's small size, the coordinates where the UAV should be sent can be downlinked in seconds when the satellite is on the horizon of the ground station. The coordinates would then be produced with the labels and the georeferencing of the HSI. An illustration of this is shown in Figure 4.4. For HYPSO-2 with a SDR, it is then possible to send the coordinates directly from the satellite to the autonomous agents. Similar systems that send information from the satellite to remote units are the Iridium NEXT system. This is based on Internet of Things (IoT) services and has mega-constellations of satellites [9].

# Chapter 5

# Implementation and Design

In this chapter the implementation that is necessary for doing classification onboard the satellite is presented. The implementation is divided into three parts: Ground Training, Onboard Classification, and Decode Labels. A figure of how the implementation is linked together is viewed in Figure 5.1. The implementation that will be discussed below can be found in the Appendix A and at [30].

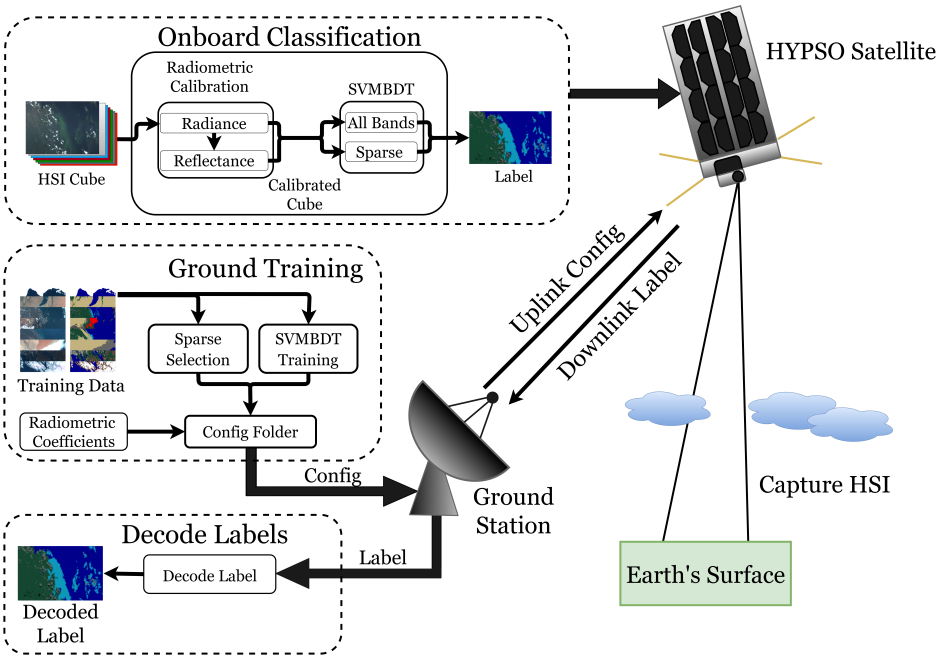



Figure 5.1: An illustration of the implementation divided into three parts: Ground Training, Onboard Classification, and Decode Labels.

## 5.1 Ground Training

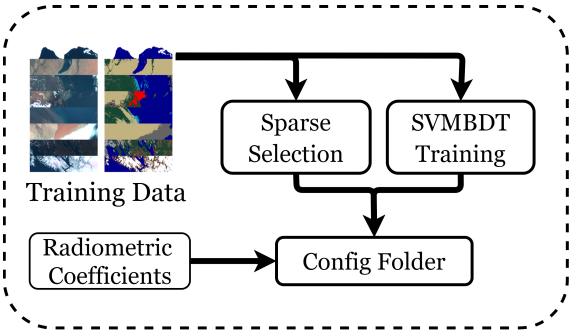



Figure 5.2: Illustration of the Ground Training.

The ground training has the task of making training data, doing the sparse selection, and saving the radiometric coefficients to the config folder. This is illustrated in Figure 5.2. The config folder has to be uplinked to the satellite so that the onboard classification module can load the necessary data for doing radiometric calibration and classification. The ground training functions that are described below are found in the folder **ground_training/**.

### 5.1.1 SVMBDT Training

The SVMBDT training aims to create the necessary parameters for each SVM model in the BDTs. The parameters for each SVM model are the two classes, intercept ($b$) and weights ($w_i$). To train the SVM models individually the functions **SvmDesionTreeTrain()** and **RetrieveSubData()** from the python program **SVMBDT.py** are utilized. This Python program was implemented with part of the project thesis from fall 2022 [28], and the implementation can be found at [29]. This is trained together with the Sklearn [26] function **SGDClassifier()** that trains the SVM models and produces the intercepts and weights.

For this project, three BDTs are constructed where they are trained in individual files.

These files are listed with the names:

- **svmbdt_training_bdt1.ipynb**
- **svmbdt_training_bdt2.ipynb**
- **svmbdt_training_bdt3.ipynb**

Each of the design decisions and structures is viewed and discussed in section 6.1.

For training the files the SVM models have 4 different processing modes. These modes are radiance, radiance sparse, reflectance, and reflectance sparse. These modes are trained individually and have their intercepts and weights.

To make the BDT design that the onboard classification module can read a list needs to be implemented. The general structure of this list is shown Figure 5.3

| Index | SVM Model Classes | Next index for left model | Next index for rigth model |
|---|---|---|---|
| 0 | 00,01 | 1 | 2 |
| 1 | 00,02 | 3 | 0 |
| 2 | 01,03 | 5 | 4 |
| | ⋮ | | |
| M-1 | M-2,M-1 | 0 | 0 |

Figure 5.3: The BDT general file structure with N size and M size classes.

### 5.1.2 Sparse Selection

The sparse selection has the task to analyze the training data and give a list of which bands should be considered when predicting with the SVMBDT. A binary file is then saved to the config folder, which tells what band number should be used. To make the band selection decision the Python program **ssmv.py** is utilized. This program is a part of the project Efficient Learning and Optimization Tools for Hyperspectral Imaging Systems (ELO-Hyp) [1]. The implementation of this can be found at [17].

### 5.1.3 Radiometric Coefficients

To calculate the radiometric calibration some calibration files are needed. For doing radiance the **rad_coeffs_FM_binx9.csv** and **spectral_coeffs_FM_order2.csv** found in the **calibration_files/** folder. The is provided by Radcalnet is the top-of-atmosphere reflectance data [6]. For reflectance, the **mean_extraterrestrial_solar_irradiance_chkur** that is the mean solar exo-atmospheric irradiance, that is also provided by RadCalNet [6].

### 5.1.4 Config File

The purpose of the config folder is to store all the data produced in the ground training module and later uplink the folder to the satellite. The onboard classification module loads the necessary data and uses it to calibrate and predict HSI. The folders for radiometric calibration are **calibration_files/** and **constans**, and for SVMBDT are **bdt/**, **sparse_bands/** and **svm_models/**. The config folder structure is viewed in Figure 5.4.

The benefit of having a config folder is that if the training data is improved and re-trained in the future, only the config folder needs to be uplinked to the satellite. The onboard classification module doesn't need to be updated and re-uplinked to the satellite. The config folder is found at **onboard_classification/onboard_classification**.

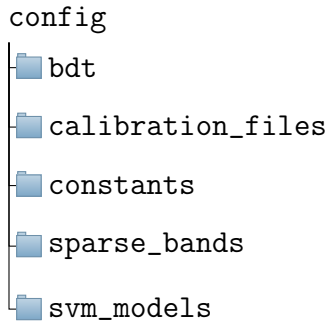



Figure 5.4: Config Folder Structure.

## 5.2 Onboard Classification

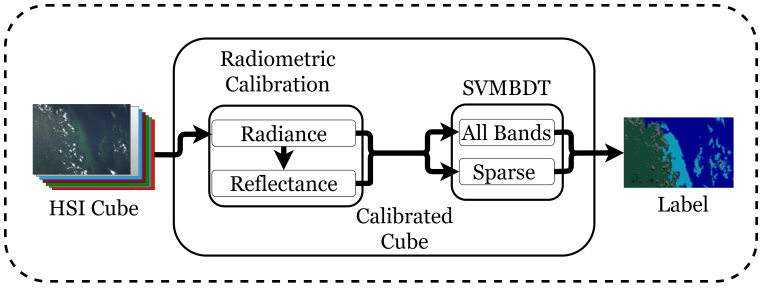



Figure 5.5: Illustration of onboard classification module.

The onboard classification module that is illustrated in Figure 5.5 combines the two implementations radiometric calibration and SVMBDT. The onboard classification first loads the target HSI Cube onboard the satellite and then starts the init functions, **radiometric_calibration_init()** and **svmbdt_init()**. These two functions will load the data from the config folder. After this, if no errors have occurred, it starts looping through the pixels of the HSI Cube and starts labeling them. First, the pixel is radiometric calibrated with the function **radiometric_calibration_pixel()** and then predicted with the function **svmbdt_predict()** and saves it in the struct **Labeled_Image**. This struct is then stored as a binary file with the function **labeled_image_bin()**. For every HSI Cube that is being captured captures the binary file will then be called **labels.bin**. This file is the downlinked labeled image and needs to be decoded to make a png image out of it. How the decoding is happening is shown in Figure 5.7. To make the onboard classification module execute, this console command needs to be executed:

```
./onboard_classification.exe <capture folder file path> <output file path>
    <config folder prefix> <bdt index> <processing mode> <local angels
   path>
```

The <capture folder file path> is the folder path where the HSI cube is stored. <output file path> what the path the labels binary file should be stored. <config folder prefix> is the config folder prefix path. The <bdt index> has the available number listed in the table below:

| Number | BDT |
|---|---|
| 1 | BDT 1 |
| 2 | BDT 2 |
| 3 | BDT 3 |

Table 5.1: <bdt index> options.

The <processing mode> options available are listed in the table below:

| Number | Processing mode |
|---|---|
| 0 | Radiance |
| 1 | Radiance Sparse |
| 2 | Reflectance |
| 3 | Reflectance Sparse |

Table 5.2: <processing mode> options.

With these options, the ground station can then decide what type of processing modes and BDTs structures that should be used for labeling the HSI captures.

### 5.2.1 Radiometric Calibration

The radiometric calibration implementation is taking inspiration from the paper [6], where HYPSO-1 HSI cubes were radiometrically calibrated on the ground. The goal of this thesis is to have radiometric calibration implemented onboard the satellite for this project. The radiometric calibration implemented can also be used for future projects where calibration of the HSI cubes is necessary.

Radiometric calibration that is discussed in section 3.2 will turn the digital values from the captured HSI cubes into physical values. The reason for using radiometric calibration in the classification is that the HSI captures are taken from different areas of the Earth and will have different radiance and reflectance. This will give the HSI a similar intensity, that fits the training data and can make the predictions of the labels more robust.

Radiance is calculated with the Equation 3.6. The function that does this is the **apply_radiometric_calibration()** with the code snippet where it is calculated shown as:

```
for(uint32_t i = 0; i < SPECTRAL_BAND_SIZE; i++){
    pixel[i] = pixel[i] - background_value;
    pixel[i] = pixel[i] * radiometric_calibration_coeff[i] / exp;
}
```

When a HSI cube is loaded to the onboard classification the folder where it is saved on the OPU on HYPSO-1 is **-hsi0**. The **exp** is found in the file **capture_config.ini** in the same folder. From the radiometric coefficients showed in subsection 5.1.3, the array **radiometric_calibration_coeff** values are loaded from the file **rad_coeffs_FM_binx9.csv**.

The cube can then are calibrated with reflectance as shown from the Equation 3.8. This is calculated in the function **radiance_to_reflectance()** and it's shown from the code snippet:

```
for(uint16_t i = 0; i < SPECTRAL_BAND_SIZE; i++){
    calibrated_pixel->X[i] =
    calibrated_pixel->X[i] *
    PI *
    earth_sun_distance * earth_sun_distance *  //Squared ^2
    esi->coeff[i] *
    la->coeff[local_angels_index][SOLAR_ZENITH_ANGLE_INDEX];
```

The data necessary to be loaded for this calibration is from the **constants/** the file **mean_extraterrestrial_solar_irradiance_thuillier** stored in the **esi** struct. The **ia** struct contains the solar zenith angels. On the ground these are calculated together with the HSI captures telemetry data. Due to this is not implemented and calculated onboard the satellite by the time of writing the thesis, the reflectance processing mode is not tested onboard the satellite but is tested in the computer test in section 6.2. The reflectance can then be further tested when the solar zenith angels calculations are implemented onboard the satellite.

### 5.2.2 SVMBDT

The SVMBDT consists of two parts the SVM prediction and the decision function BDT that has been explored in section 3.1. For the SVM the function **svm_linear()** is computing the Equation 3.2. A code snippet of the function is shown below:

```
for(uint16_t i = 0; i < SPECTRAL_BAND_SIZE; i++){
    pred += (X_i[i]*svm_model[3+i]) + svm_model[2];
}

if(pred >= 0){
    return -1; }
else{
    return 1; }
```

For calculating the SVM with sparse mode the function **sparse_svm()** is used. The band selection is loading from the folder in the Config Folder **sparse_bands**, where the file **sparse_bands1** is for BDT1. This applies the same to the two other BDTs structures found in the folder **sparse_bands/**.

As viewed in the Figure 5.3 the SVM says how the BDT is structured and which two classes are in tree nodes. When a HSI is predicting a pixel in that node the results are either 1 or -1. When the results are 1 it looks up the index number in the BDT list and jumps to the SVM model index with new SVM model classes. Same if the result is -1, but to the right. If the index number is 0, it stops in the tree node and set the class number in the struct array **Labeled_Image**. This will then iterate through all the pixels in the HSI cube.

## 5.3 Decode Labels

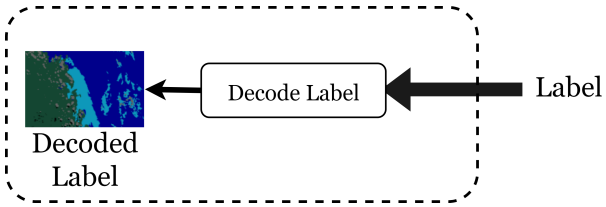



Figure 5.6: Illustration of decode labels.

When the Onboard Classification implementation has been predicted the **labels.bin** file is generated. The generated file size is dependent on the number of classes that are labeled in the image. For the default size HSI size on HYPSO-1 with 956x684 in pixels dimension, the sizes are shown in the table below:

| **N Classes** | **Labels File Size** |
|---|---|
| 1 to 2 | 82 kB |
| 3 to 4 | 163 kB |
| 5 to 16 | 327 kB |

Table 5.3: File size of the labels file structure with different N amount of classes in the labels with 956$x$684 pixels.

To make the labeled images so small the classes in the labels are sharing bytes. The labels are then transformed to the smallest value that is represented in $2^n$. For example, $2^2 = 4$ classes, and the image is labeled with the classes **{1,4,7,9}**. The labels in the **labels.bin** file will be transformed from **{1,4,7,9}** to **{0,1,2,3}**. These are stored in the data labels section of the file structure that is shown in Figure 5.7.

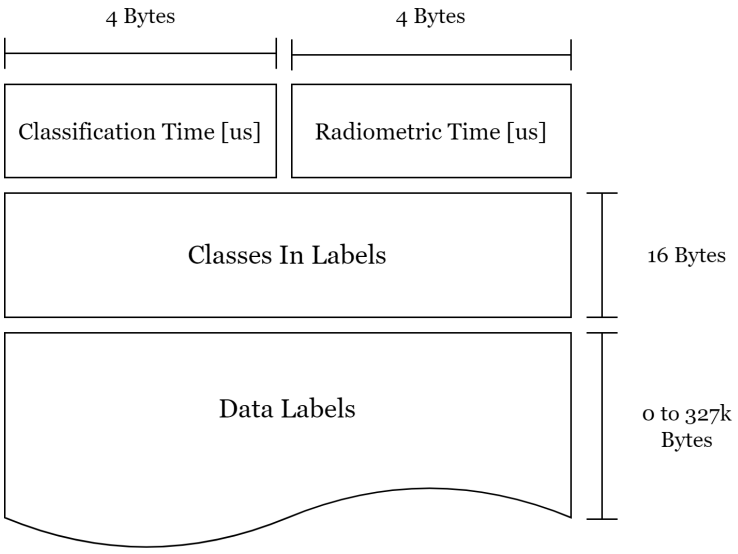



Figure 5.7: The labels file structure generated on the satellite and downlinked to the ground station.

To reconstruct the original labels of the HSI when on the ground, the Python program **decode_labels.py** is used. It uses the classes in labels in the binary file to make the labels to their original form. The program is found in the folder **decoding/** in the Appendix A and [30]. The three default dimensions that the HSI has is listed in the table below:

| **x dim** | **y dim** |
|---|---|
| 684 | 956 |
| 598 | 1092 |
| 537 | 1216 |

Table 5.4: Default HSI capture dimensions.

Since the labels are one-dimensional when stored in the binary file, the image dimensions of the HSI could be different from the default dimension, without affecting the labels. To decode the labels and create a png picture out of it, the Python program **decode_labels.py** is executed as shown below:

```
python3 decode_labels.py <compressed labels bin file path> <decoded labels
    save path>
```

Where <compressed labels bin file path> is the path for the **labels.bin** file and <decoded labels save path> is the path where the labeled png picture should be stored. When executing the program like this only the default dimension are supported. If in the future some other dimension of the HSI capture is taken, the program can then be executed as shown below:

```
python3 decode_labels.py <compressed labels bin file path> <decoded labels
    save path> <x dim> <y dim> <print metadata> <skew mode>
```

Where <print metadata> is the printing of the classes in labels, classification execution time, and loading execution time that is shown in Figure 5.7. <skew mode> is turning the images to bin-3 format.

# Chapter 6

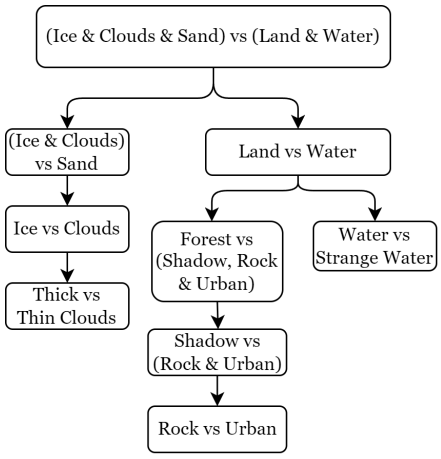
# Numerical Experiments

In this chapter the implementation that is presented in chapter 5 will be tested. The dataset is shown in section 4.1 and is used for training the three different BDT structures with four different processing modes each. The onboard classification is tested with different modes to test the overall accuracy and execution times. The test is performed on a computer to see which mode has the best performance to be executed onboard the HYPSO satellite. After the choice is made, the onboard classification is tested first on the LidSat and after uplinked and executed on HYPSO-1. The results from these tests will then be discussed.

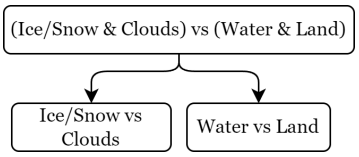
## 6.1 Binary Decision Tree Structures

This section will describe the design choices for the three different BDT structures. The goal of having different BDT structures is to see how the accuracy and execution time will differ from one another.

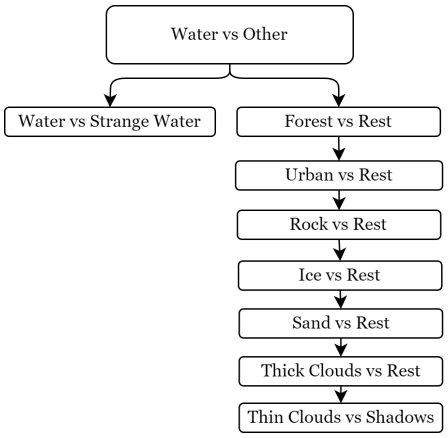
- The BDT 1 is designed first to separate clouds, sand, and ice/snow against water and land classes. It contains all the classes from the training data and the goal is to have a short travel path to the longest prediction. The longest prediction is on the rock vs the urban class, viewed in Figure 6.1 (a).
- The BDT 2 is to have a smaller tree structure, where only a limited of classes are labeled. It first separates ice and clouds against water and land, viewed in Figure 6.1 (b). This is also designed to see if this structure will have a shorter classification time versus the other two.
- The BDT 3 contains all of the classes in the training data and is structured to see if the classification time increases when the tree nodes have a long travel path to the last separation. The first separation is water versus the rest of the classes, where the rest of the classes have the longest travel path. This is viewed in Figure 6.1 (c).




(a) BDT 1




(b) BDT 2




(c) BDT 3

Figure 6.1: The three different BDT structures made for testing the onboard classification.

## 6.2 Testing implementation accuracy and execution time

Before the implementation is tested on the satellite hardware the BDT structure and type of processing mode need to be tested. To test this, the overall accuracy and classification execution time when predicting an HSI are taken into consideration. To test the overall accuracy the HSI from the training data is predicted and compared with the GT labels. The classification execution time is measured by the time it takes to radiometric calibrate the HSI and label it with the SVMBDT, which will be viewed in the subsections below.

Since the onboard classification that is explained in section 5.2 is designed to be executed onboard the satellite, a shell script is produced to test all the different BDTs with the different processing modes to run on a computer. This program is named **onboard_classifcation.sh** and can be found at Appendix A and [30] in the folder **onboard_classification/onboard_classifcation/src/**. The labeled binary files which are produced are decoded with the Jupyter notebook **decode.ipynb** found in the folder **decoding/**. These programs are executed on the computer with the specifications listed in Table 6.1.

| | |
|---|---|
| **Computer** | Lenovo Legion 7 Slim Pro |
| **Processor** | AMD Ryzen 7 5000 Series 3.2GHz |
| **Graphic Card** | Nvidia RTX 3060 |
| **Memory** | 16 GB RAM |
| **OS** | Windows 11 |

Table 6.1: Hardware and software the tests are executed on.

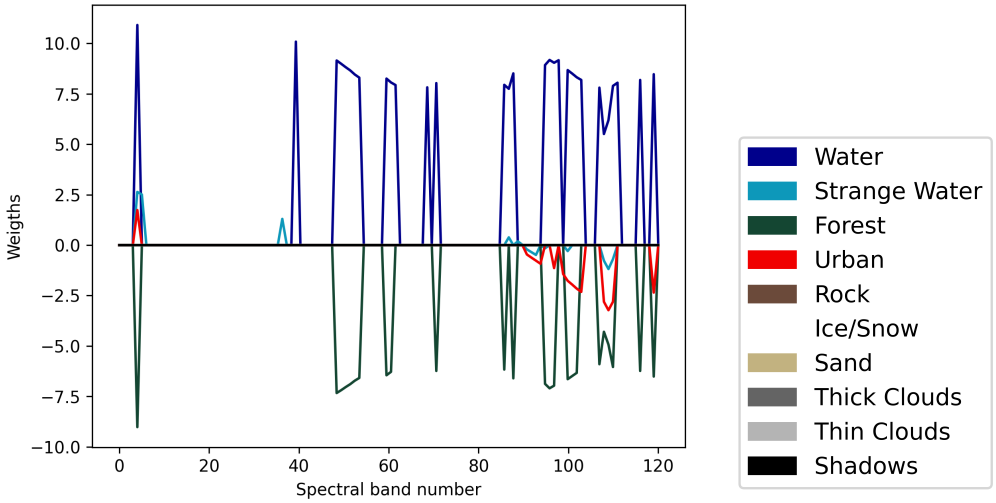
When these tests are executed the onboard classification memory usage is then monitored, and it's found that the peak memory usage is 1328 kB as shown in Table 6.2. As described in section 2.3 the OPU has available 0.5 GB of memory for new software uplinked to the satellite. The onboard classification uses well below that memory limit, making it possible to be executed onboard the satellite.

| **Program** | **Peak Memory Usage** |
|---|---|
| Onboard Classification | 1328 kB |

Table 6.2: Peak memory usage of the onboard classification program when separately classifying the HSI cubes.

Selecting the spectral bands that could be ignored when classifying with the SVMBDT, the Sparse SVM introduces in subsection 5.1.2 is utilized. The bands that are selected with the band weights are shown in Figure 6.2, where 40 spectral bands are selected. These bands will then be

used when doing the processing modes radiance sparse and reflectance sparse. The all bands mode uses all 120 spectral bands of the HSI cube.




(a) Sparse Selection (b) Legend

Figure 6.2: Sparse band selection of the HSI training data from Figure 4.1. (a) The weights produce for each spectral band. (b) Color mapping legend for each class weight.

Before the onboard classification is tested, the training weights are produced by the Jupyter notebooks presented in subsection 5.1.1. For doing the training the data set is loaded, where 10% random pixels are used. The training times with the different BDT structures and processing modes are shown in Table 6.3. Observed in the table the radiance training times are higher than the reflectance. This is due to reflectance being an easier model to solve, making it faster to converge than radiance.

| **BDT Number** | **Radiance** | | **Reflectance** | |
|---|---|---|---|---|
| | **All bands** | **Sparse** | **All bands** | **Sparse** |
| 1 | 39.01s | 52.60s | 8.08s | 6.49s |
| 2 | 22.47s | 35.96s | 2.23s | 2.04s |
| 3 | 51.93s | 43.22s | 5.20s | 4.59s |

Table 6.3: Training times of BDT 1, BDT 2, and BDT 3 for each type of processing mode.

In the section 2.5 it was mentioned that when building with GCC a -O3 flag can be added for better execution times. The onboard classification is built with and without the -O3 with the BDT 1. The result of this is shown in Table 6.4, where when using the -O3 flag makes it a significantly faster classification time. The -O3 flag will then be used for the test in the subsections below. How to build the onboard classification in the console with the -O3 flag is shown below:

```
gcc onboard_classification.c ../../radiometric/src/radiometric.c
../../svm_bdt/src/svmbdt.c -o onboard_classification.exe -O3
```

| **Without -O3** | | | |
|---|---|---|---|
| **Radiance** | | **Reflectance** | |
| **All bands** | **Sparse** | **All bands** | **Sparse** |
| 1.315s | 0.878s | 1.518s | 1.095s |

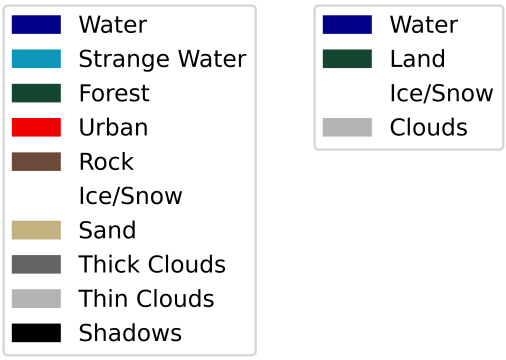
| **With -O3** | | | |
|---|---|---|---|
| **Radiance** | | **Reflectance** | |
| **All bands** | **Sparse** | **All bands** | **Sparse** |
| 0.278s | 0.194s | 0.314s | 0.238s |

Table 6.4: Mean classification time with BDT1 with all the processing modes, without and with the -O3 optimization flag.

When looking just a the radiometric calibration times, the execution time can be viewed in Table 6.5 with radiance and reflectance.

| **Radiance** | **Reflectance** |
|---|---|
| 0.143 s | 0.176 s |

Table 6.5: Mean Radiometric calibration execution time when testing on computer.



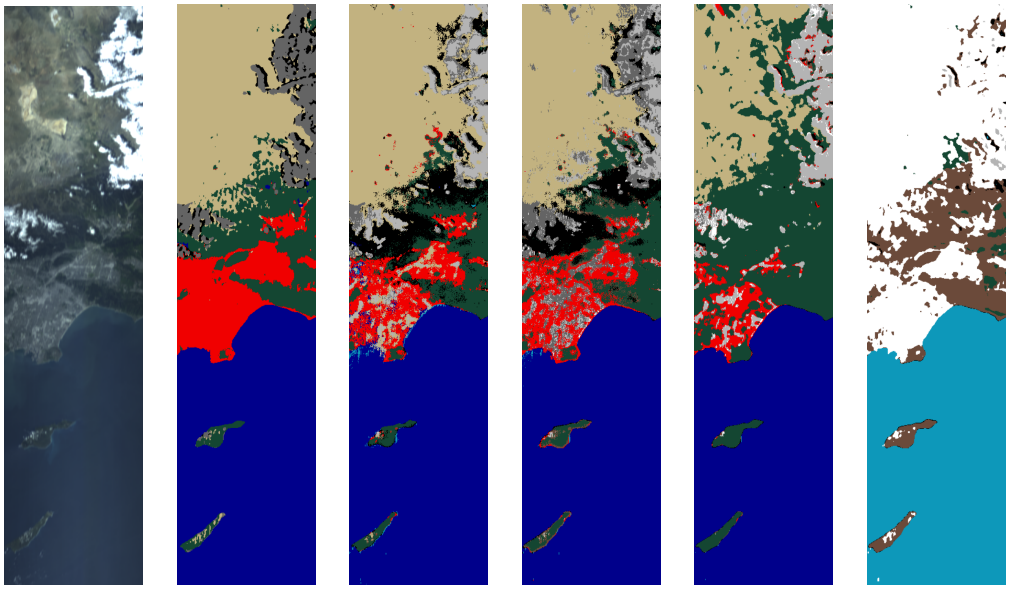

(a) (b)

Figure 6.3: Color mapping for the labeled results in the subsections below. (a) BDT 1 and 3, and (b) BDT 2.

### 6.2.1 BDT 1

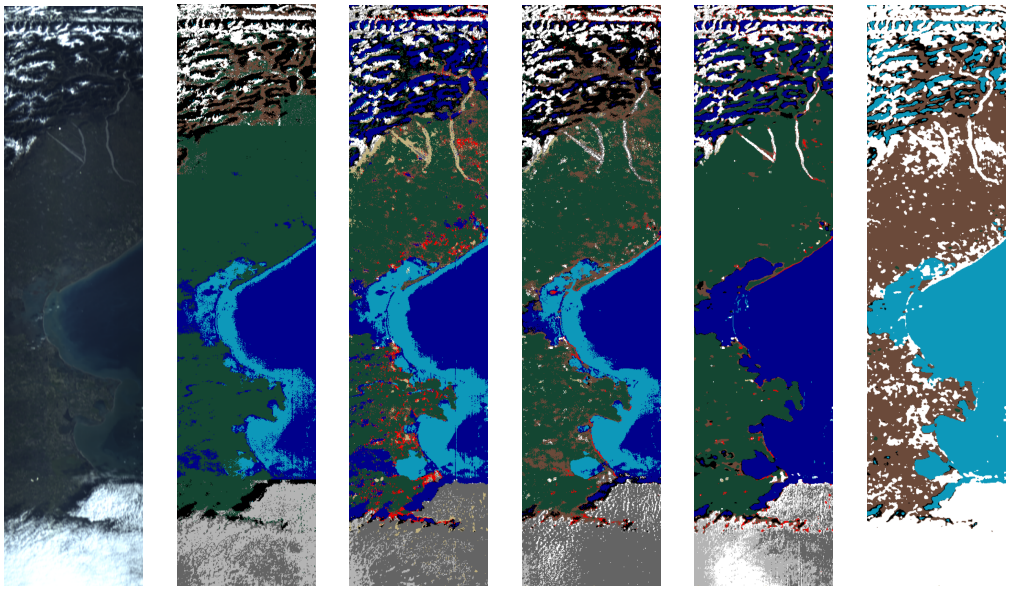
| Area | Radiance | | Reflectance | |
|---|---|---|---|---|
| | All bands | Sparse | All bands | Sparse |
| **South Pole [%]** | 99.01 | 97.93 | 99.08 | 60.52 |
| **Tharthar [%]** | 94.74 | 93.92 | 96.89 | 0.04 |
| **Kampala [%]** | 75.43 | 83.67 | 80.61 | 3.45 |
| **Los Angeles [%]** | 84.12 | 79.31 | 80.08 | 1.4 |
| **Sittwe [%]** | 91.65 | 88.96 | 84.39 | 4.77 |
| **Gobabeb [%]** | 93.39 | 88.66 | 96.83 | 0.59 |
| **Trondheim [%]** | 82.46 | 88.06 | 90.1 | 24.09 |
| **Trondheim, (Snow) [%]** | 80.55 | 83.33 | 84.43 | 48.83 |
| **Venice [%]** | 62.84 | 70.2 | 63.73 | 17.23 |
| **Svalbard [%]** | 75.34 | 74.83 | 75.76 | 35.26 |
| **Mean overall accuracy [%]** | 83.95 | 84.89 | 85.19 | 19.62 |
| **Mean loading time [ms]** | 40.692 | 40.64 | 43.852 | 43.677 |
| **Mean classification time [s]** | 0.278 | 0.194 | 0.314 | 0.238 |

Table 6.6: BDT 1 testing accuracy, loading, and execution time from the training set data.



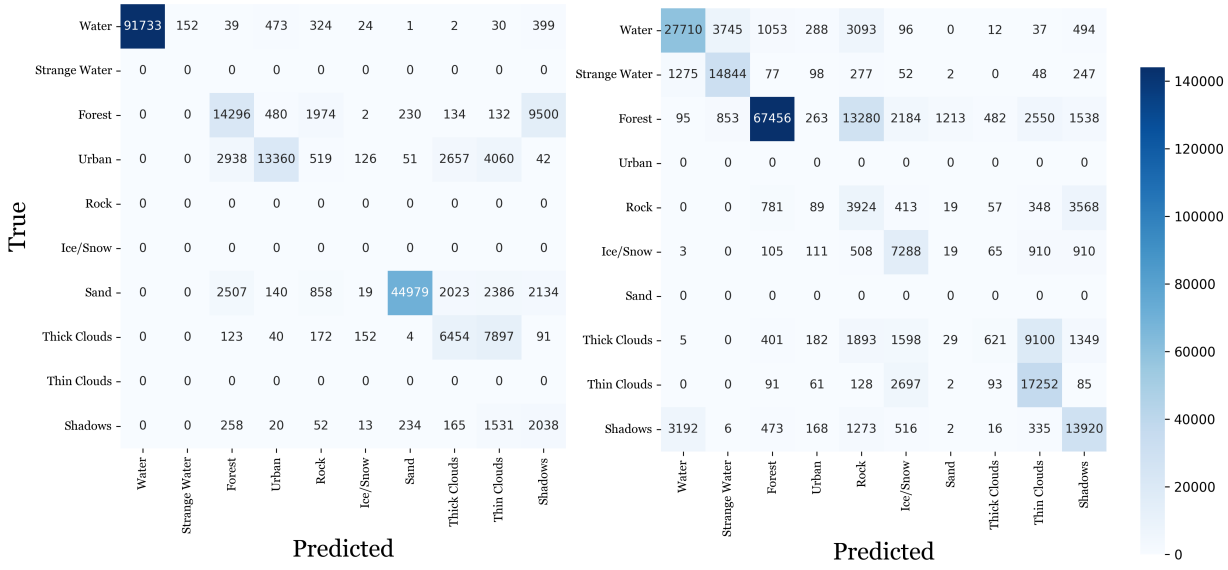
(a) (b) (c) (d) (e) (f)

Figure 6.4: Los Angeles with a) Capture, b) GT and c)-f) labeled predicted images in chronological order as in the table.



(a) (b) (c) (d) (e) (f)

Figure 6.5: Venice with a) Capture, b) GT, and c)-f) labeled predicted images in chronological order as in the table.



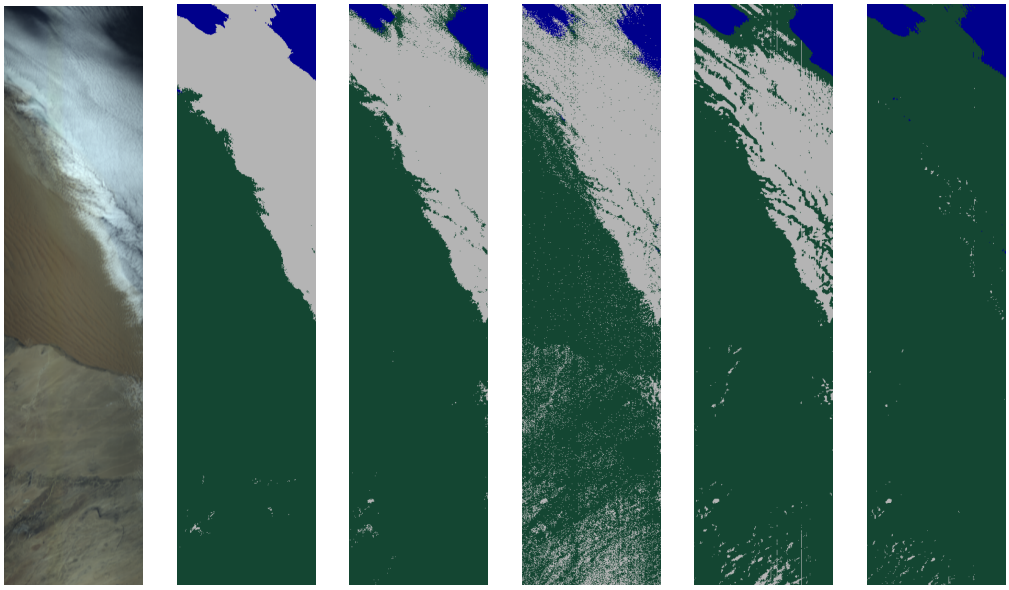

(a) Los Angeles (b) Venice (c)

Figure 6.6: Confusion matrix of the processing mode with radiance sparse, with (c) as the color bar with the number of pixels in each class.

### 6.2.2 BDT 2

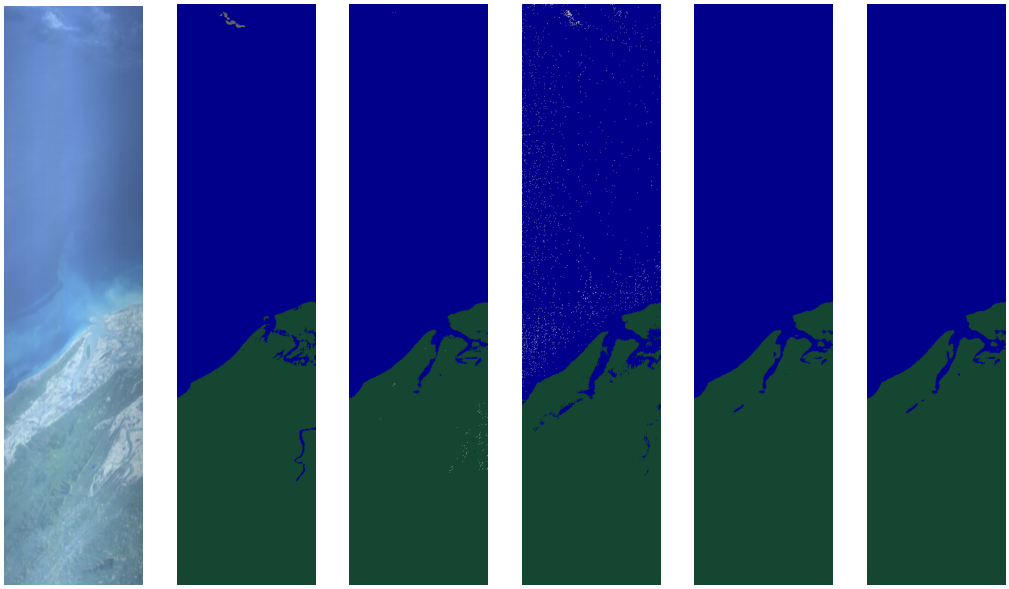
| Area | Radiance | | Reflectance | |
|---|---|---|---|---|
| | All bands | Sparse | All bands | Sparse |
| **South Pole [%]** | 98.46 | 93.78 | 98.86 | 98.95 |
| **Tharthar [%]** | 88.84 | 89.01 | 88.56 | 88.49 |
| **Kampala [%]** | 95.15 | 92.53 | 93.04 | 91.64 |
| **Los Angeles [%]** | 75.88 | 76.27 | 75.84 | 74.77 |
| **Sittwe [%]** | 98.99 | 99.02 | 98.39 | 98.29 |
| **Gobabeb [%]** | 91.97 | 67.15 | 95.48 | 79.19 |
| **Trondheim [%]** | 93.18 | 93.32 | 94.09 | 93.95 |
| **Trondheim, (Snow) [%]** | 84.28 | 79.79 | 84.33 | 81.31 |
| **Venice [%]** | 88.32 | 88.55 | 87.01 | 85.38 |
| **Svalbard [%]** | 57.98 | 67.19 | 57.91 | 45.78 |
| **Mean overall accuracy [%]** | 87.3 | 84.66 | 87.35 | 83.77 |
| **Mean loading time [ms]** | 39.165 | 39.483 | 43.428 | 42.578 |
| **Mean classification time [s]** | 0.23 | 0.161 | 0.27 | 0.192 |

Table 6.7: BDT 2 testing accuracy, loading, and execution time from the data of the training set.



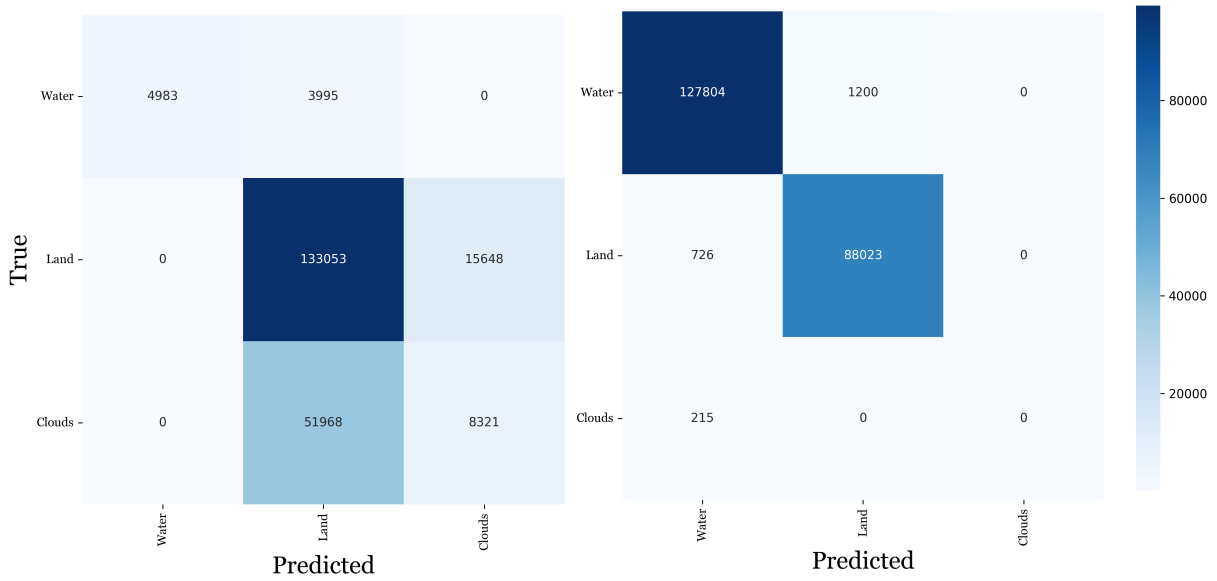
(a) (b) (c) (d) (e) (f)

Figure 6.7: Gobabeb with a) Capture, b) GT, and c)-f) labeled predicted images in chronological order as in the table.



(a) (b) (c) (d) (e) (f)

Figure 6.8: Sittwe with a) Capture, b) GT and c)-f) labeled predicted images in chronological order as in the table.



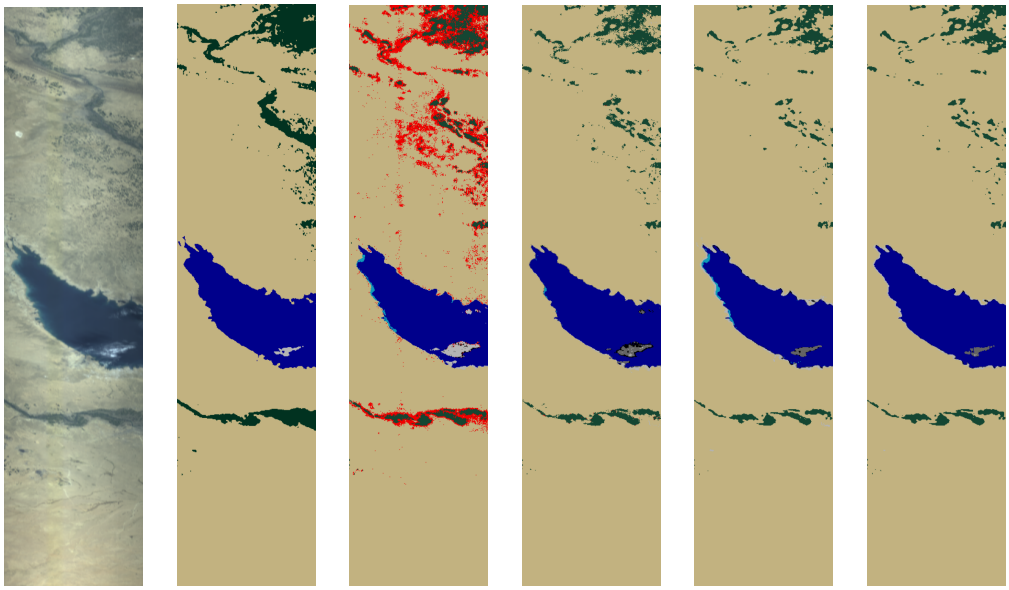

(a) Gobabeb (b) Sittwe (c)

Figure 6.9: Confusion matrix of the processing mode with radiance sparse, with (c) as the color bar with the number of pixels in each class.

### 6.2.3 BDT 3

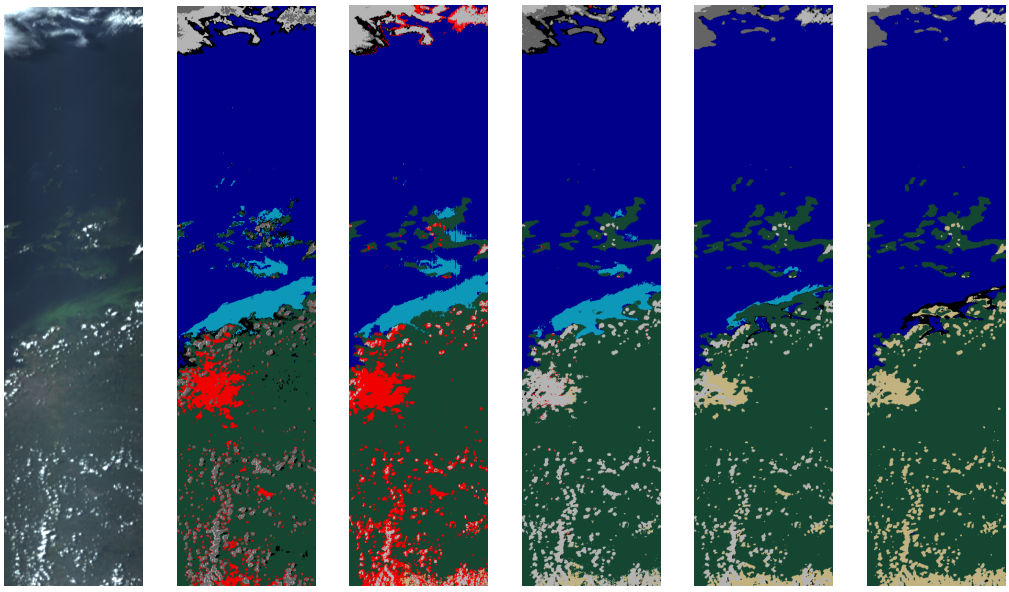
| Area | Radiance | | Reflectance | |
|---|---|---|---|---|
| | All bands | Sparse | All bands | Sparse |
| **South Pole [%]** | 98.98 | 98.63 | 98.32 | 97.65 |
| **Tharthar [%]** | 80.36 | 66.65 | 71.91 | 71.12 |
| **Kampala [%]** | 64.85 | 82.74 | 82.06 | 81.86 |
| **Los Angeles [%]** | 54.79 | 52.4 | 56.57 | 55.26 |
| **Sittwe [%]** | 94.7 | 86.72 | 87.27 | 86.4 |
| **Gobabeb [%]** | 89.57 | 84.05 | 97.69 | 97.15 |
| **Trondheim [%]** | 83.75 | 85.67 | 87.9 | 88.25 |
| **Trondheim, (Snow) [%]** | 75.92 | 68.96 | 77.15 | 74.9 |
| **Venice [%]** | 60.41 | 69.15 | 70.33 | 68.43 |
| **Svalbard [%]** | 50.4 | 50.4 | 50.34 | 50.22 |
| **Mean overall accuracy [%]** | 75.37 | 74.54 | 77.95 | 77.12 |
| **Mean loading time [ms]** | 39.491 | 40.654 | 43.271 | 43.641 |
| **Mean classification time [s]** | 0.291 | 0.177 | 0.344 | 0.209 |

Table 6.8: BDT 3 testing accuracy, loading, and execution time from the training set data.



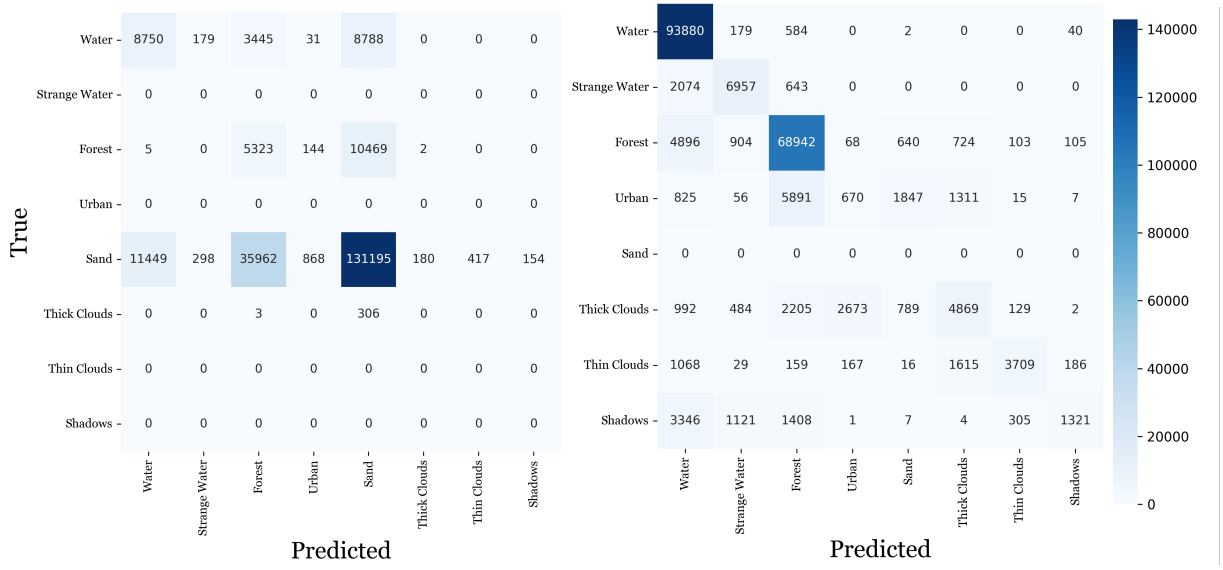
(a) (b) (c) (d) (e) (f)

Figure 6.10: Tharthar with a) Capture, b) GT, and c)-f) labeled predicted images in chronological order as in the table.



(a) (b) (c) (d) (e) (f)

Figure 6.11: Kampala with a) Capture, b) GT, and c)-f) labeled predicted images in chronological order as in the table.




(a) Tharthar (b) Kampala (c)

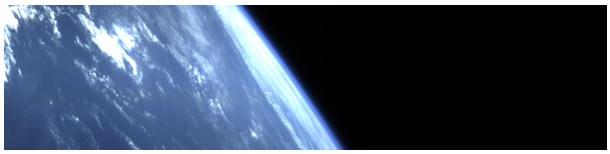
Figure 6.12: Confusion matrix of the processing with radiance sparse, with (c) as the color bar with the number of pixels in each class.

### 6.2.4 Deciding mode to be used onboard the satellite

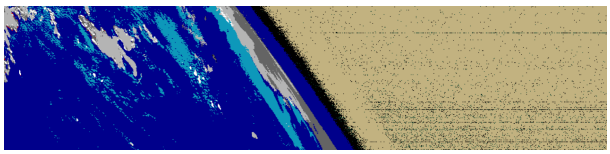
In this section, it will be discussed which BDT structure and processing mode that will be the best choice to use for classifying onboard the satellite.

From the results in the subsection 6.2.1, subsection 6.2.2, and subsection 6.2.3 it is shown that the BDT 3 has the worst performance compared to the BDT 1 and BDT 2 when measured by accuracy and classification time. Between these two structures, it is shown that BDT 2 has a slightly faster execution time than BDT 1, due to the lower amount of classes. The importance of the onboard classification is the model for every HSI around the Earth. The trade-off of having more classes is better, with the expense to have a slightly higher classification time. Between the processing modes in the BDT 1 structure, the best accuracy is with the reflectance all bands. With the reflectance sparse it is shown that has very low accuracy, which can be caused by overfitting in the nodes from the trained data that has been used. Due to the solar zenith angle not being implemented onboard the satellite as discussed in subsection 5.2.1 and having a much higher classification time, the radiance sparse could be a better choice. Even with 0.5% lower mean overall accuracy, it never gets worse than 70%. This value goes below with other processing modes.

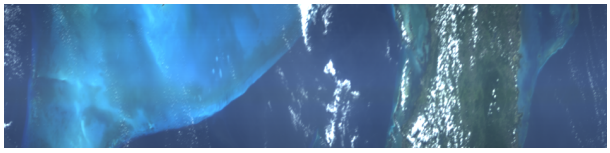
To further test the robustness of the BDT 1 radiance sparse, some HSI captured not in the data set is labeled. The labels in this test can't be measured due to not having GT, but the quality of the labels can be observed by comparing HSI in RGB representation. The different HSI captures with the labeling results are shown in Figure 6.13. From the results in the figure, the onboard classification with the mode chosen can be observed that it is overall robust. Due to the training data not containing a space class, it is labeled as sand as shown in Figure 6.13(a)-(b). The space capture is from an HSI that had pointing errors when captured, making the image contain space and the curvature of the Earth. The ground station can then know that space is labeled as sand and can determine by the labeled image that the HSI had pointing errors. The Florida capture is shown in Figure 6.13(c)-(d) that shallow water is labeled as strange water as expected. The moon capture, shown in Figure 6.13(e)-(f), labeling is not working for these types of scenarios. The Svalbard capture in Figure 6.13(g)-(h) shows that a lot of clouds are blocking much of the surface. The labels produced classified much of the image as clouds, which can give the ground station an idea of how big a percentage is clouded in the HSI. The last HSI capture Sogn in Figure 6.13(i)-(j), shows that the labeling is robust. Some errors are shown where some clouds are labeled as urban. With these tests done the onboard classification with mode BDT 1 radiance sparse is used to label the different HSIs in the test shown in section 6.3 and section 6.4.



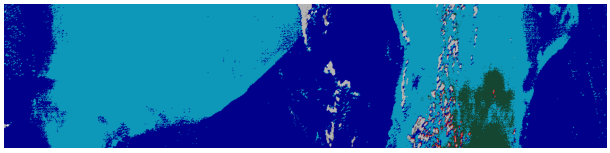
(a) Space Capture



(b) Space Labels

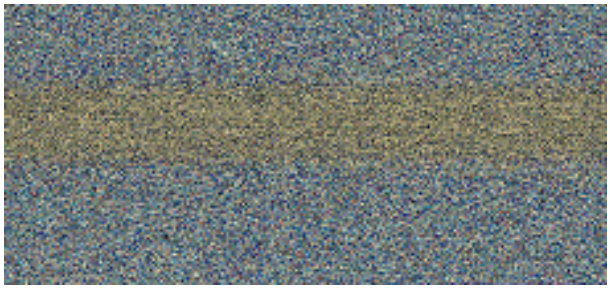


(c) Florida Capture

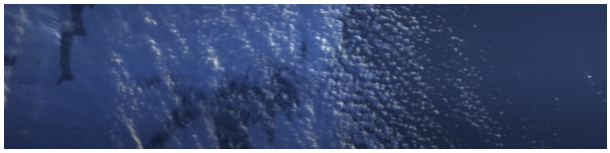


(d) Florida Labels

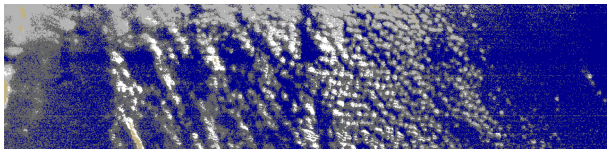
(e) Moon Capture



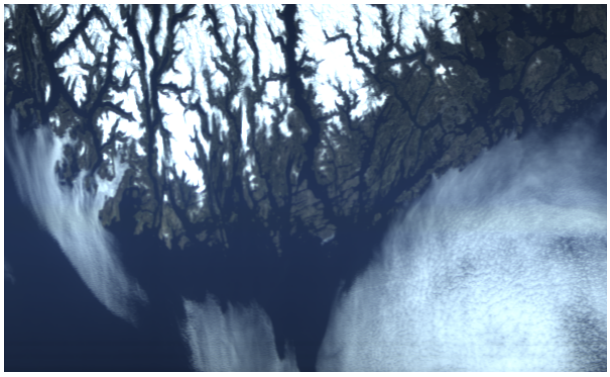
(f) Moon Labels



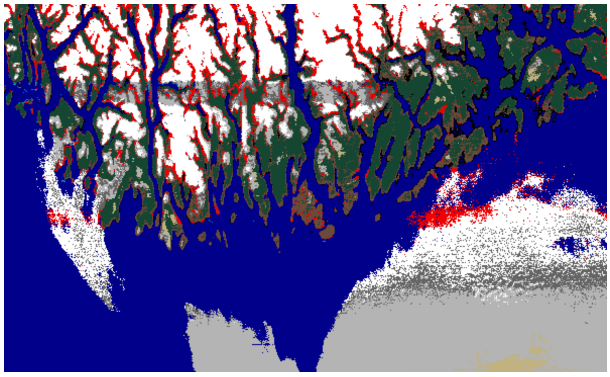
(g) Svalbard Capture



(h) Svalbard Labels



(i) Sogn Capture



(j) Sogn Labels

Figure 6.13: Different HSI capture in different areas and scenarios to test the onboard classification model's robustness with mode BDT 1 radiance sparse.

## 6.3 Testing on LidSat

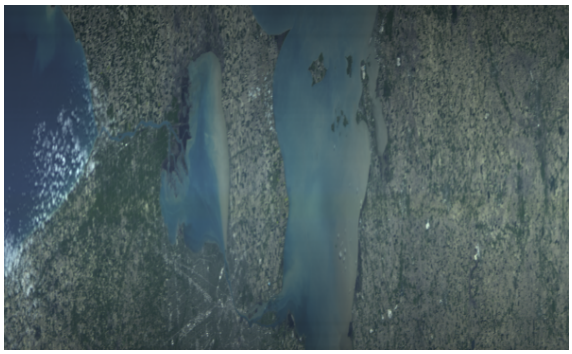
As mentioned in the section 2.3, new software that is developed for the satellite, needs to be tested first on the setup Lidsat before uplinking and executing on the satellite. To test the onboard classification on the Lidsat three different HSI captures are uploaded to it. The HSI captures are visualized in the Figure 6.14(a),(c), and (e). To execute the onboard classification the following console command is executed:

```
./onboard_classification.exe "capture_folder/" "labels.bin" "config/" "1"
   "1"
```

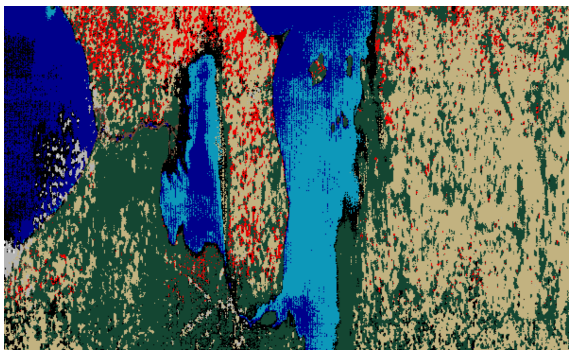
To test the classification time each HSI captures is tested two times, one without and one with the -O3 flag. The different classification execution times for the three HSI captures are shown in Table 6.9 and labeling of the HSI in Figure 6.14 (b), (d) and, (f). When these tests are performed, it is shown that -O3 stills have faster execution times than without. The -O3 flag will then be used for the build for onboard classification when uplinked to the HYPSO-1 satellite.

| **HSI Capture** | **Without -O3** | | **With -O3** | |
|---|---|---|---|---|
| | **Classification** | **Config** | **Classification** | **Config** |
| **Erie** | 33.14 s | 0.46 s | 8.57 s | 0.49s |
| **Vigo** | 36.06 s | 0.46 s | 9.52 s | 0.49 s |
| **Venice** | 35.68 s | 0.46 s | 9.12 s | 0.49 s |

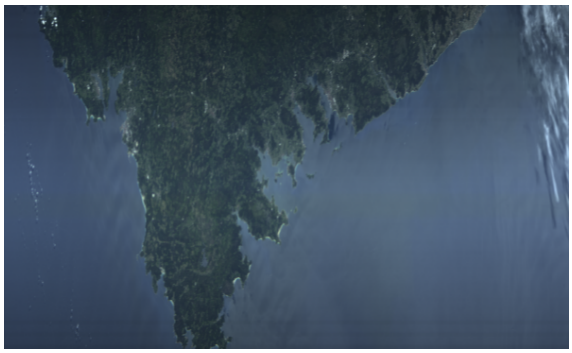
Table 6.9: Classification and Loading Execution Time of the different HSI captures tested on the LidSat.



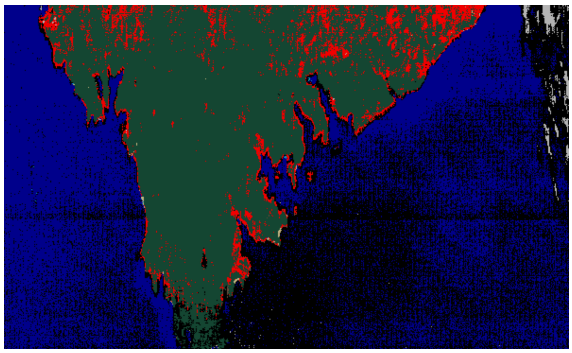
(a) Erie Capture



(b) Erie Labels

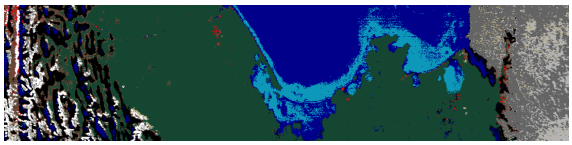


(c) Vigo Capture



(d) Vigo Labels

(e) Venice Capture



(f) Venice Labels

Figure 6.14: Lidsat test images, where RGB representation of the BDT cubes on the left and the corresponding labeled images on the right.

## 6.4 Testing on HYPSO-1

Now that the onboard classification is tested on satellite hardware, it can be uplinked to the HYPSO-1 satellite. This is done via the ground station in Trondheim, Norway where it's uplinked to HYPSO-1 under an orbital pass. To make the onboard classification execute after a HSI is captured a script is added to the OPU. After the onboard classification is executed and HYPSO-1 is again in the horizon of the ground station the labeled HSI is downlinked. The downlink time of the label versus the whole HSI cube is shown in Table 6.10.

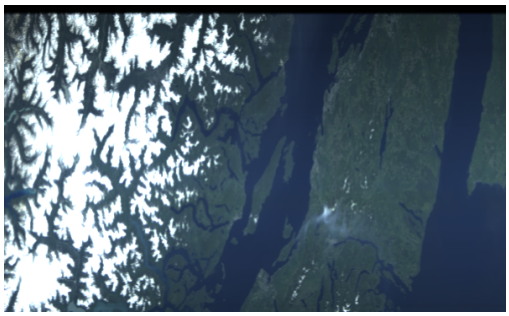
| **HSI Cube** | **Labeled HSI** |
|---|---|
| 10 min 24 s | 2.64 s |

Table 6.10: Downlink times of HSI Cube and labeled HSI from HYPSO-1 to the ground station.

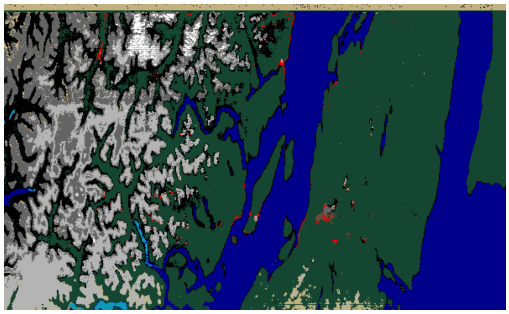
This procedure is done multiple times, and some of the HSI captures with the associated labels are shown in Figure 6.15. The different classification and loading execution times when executing onboard HYPSO-1 are shown in Table 6.11.

| **HSI Capture** | **Classification** | **Config** |
|---|---|---|
| **Griegeschelt** | 8.25 s | 0.49 s |
| **Delaware** | 8.45 s | 0.49 s |
| **Greenbay** | 8.04 s | 0.49 s |
| **Ingdalen** | 9.06 s | 0.49 s |

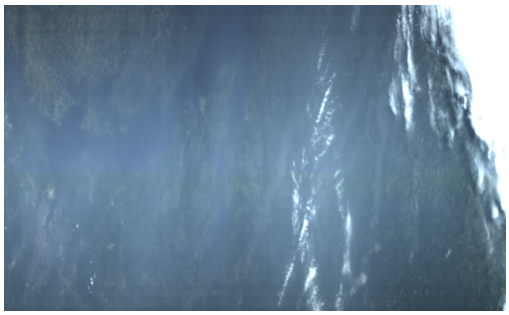
Table 6.11: Classification and config loading times when labeling the four captures onboard HYPSO-1.



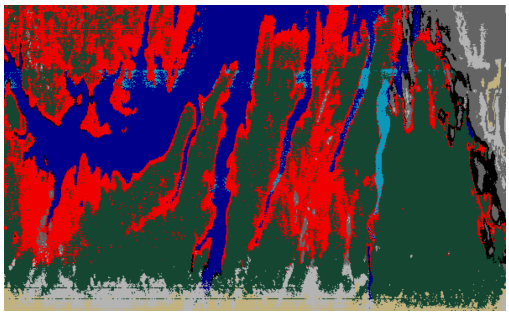
(a) Griegeschelt Capture



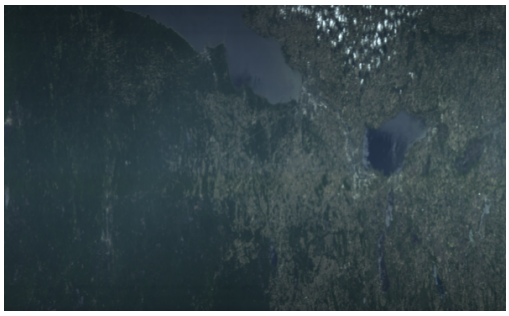
(b) Griegeschelt Labels



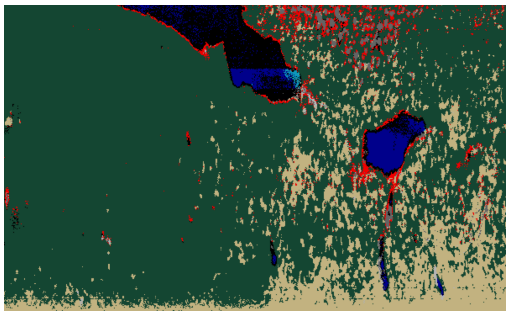
(c) Delaware Capture



(d) Delaware Labels

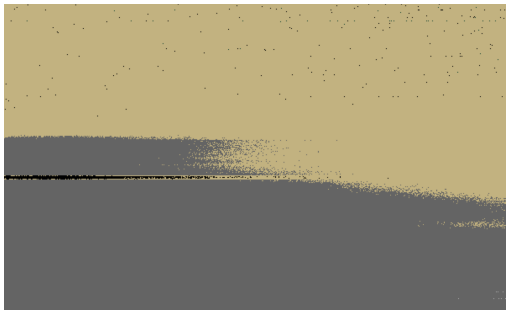


(e) Greenbay Capture



(f) Greenbay Labels

(g) Ingdalen Capture



(h) Ingdalen Labels

Figure 6.15: One of the first onboard classification labeled images downlinked from the HYPSO-1 satellite. To the left the HSI captures and to the right the associated classified labels.

## 6.5 Discussion

In this chapter the onboard classification module has been tested first on a computer, to test the different overall accuracy and execution times with the different processing modes and BDT structures. The mode and BDT chosen is BDT 1 radiance sparse, due to having one of the lowest execution times and robust labeling accuracy, but also having all the 10 classes.

From the labeling results on Figure 6.13 (b), (h) and Figure 6.15 (h), it is shown captures with pointing errors, distortions or mostly capturing clouds. The section 4.2 shows that many HSI captures have this problem, and may not be downlinked if known. With the labels produced onboard and having a short downlink time shown in Table 6.10, it can be used to help determine if the HSI capture should be downlinked or not.

In the Griegshelt and Delaware labels in Figure 6.15 there is labeled strange water that potentially can have algae blooms or pollution in it. In section 4.3 it was explained how the labels could be used to determine the coordinates where the autonomous agents should be sent. For HYPSO-1 that has not had a SDR, the coordinates must have been sent via the ground station and to one of the agents afterward. In the future when the HYPSO-2 satellite is launched into orbit this could be directly sent to the agents, with the areas labeled with strange water. If the HSI captured is containing algae blooms fish farms could be warned that the area has HABs in the water. Doing this could help reach the UN's sustainability goals that are mentioned in section 1.2. To better separate the strange water class, some subclasses of it could be made in the future for the data set. This could be the algae bloom class, pollution class, and shallow water class. Another class that could be added is a space class so that space is not labeled as sand. If this is changed the BDT in the onboard classification module needs to be re-designed and re-trained to make those labels in the HSI.

Even though the labeled HSI gives a good representation of how the HSI captures looks like, some misclassifications occur in the labels. Viewed in the Griegeschelt labels in Figure 6.15 the snow-covered areas are labeled as thin and thick clouds. Also errors in the Delaware labels where some cloud pixels are labeled as urban. And the Greenbay labels dark water is labeled as shadows. To have fewer errors in the labels, the data set showed in section 4.1, could be improved by further investigating the surfaces in the HSI captures.

To give a better understanding of how the labels can be utilized, the onboard classification module will be further executed onboard HYPSO-1 and the labels downlinked to the ground station.

# Chapter 7

# Conclusion

In this thesis, it has been implemented the classification module on the image processing pipeline on the HYPSO-1 satellite. Inside the module is implemented radiometric calibration to reduce stripping noise and factor in exposure of the HSI cube. After the HSI is calibrated the pixels are labeled with the ML algorithm SVMBDT. To train the ML algorithm the ground training module produces and saves the necessary training data and radiometric coefficients to a config file that onboard classification can read from. To train the models a data set with 10 HSI with GT is produced for this project. After the HSI is labeled, it is produced as a binary file. The binary file is then decoded by the decode labels module that produces a png image of the labels.

The implementations have been then tested first on a computer to test the accuracy and the execution time for the different BDT designs, radiometric calibration, and sparse selection. From this test, it is concluded the best mode for the classification onboard the HYPSO-1 satellite is BDT 1 radiance sparse. This is due to overall high accuracy and one of the lowest execution times compared to the other modes.

The onboard classification is then tested on target hardware Lidsat where three HSI was tested. Labels are produced and working as expected, but with a slower classification time, due to other hardware than the computer.

At the end the onboard classification is uplinked to the HYPSO-1 satellite. 4 HSI is labeled onboard and downlinked to the ground. The labeled images were decoded and was showing the expected behavior, with a similar execution time as the Lidsat. The labels it produced give a good indication if the HSI cube errors in them should be downlinked or not, even with some misclassifications.

## 7.1 Further Work

The onboard classification will be further executed onboard the HYPSO-1 satellite to give a better understanding of how the labels could be used. To verify that the strange water pixels could contain HABs, a test with one of the autonomous agents would verify this. For the HYPSO-2 with the SDR an implementation that calculates the coordinates of HABs with the labels and georeferencing of the HSI. They could then be directly sent from the satellite to the autonomous agents for more detailed capturing.

As shown subsection 4.1.1 labeling and making a training set for HSI is a difficult task. The labels that have been produced have some errors that are difficult to correct without local knowledge. To make the classification module more accurate the training data could be improved by doing some more research on the local areas. The data set could also be added more classes with subclasses of strange water so that the HABs could be separated by the pollution and shallow water.

For a better execution time for the onboard classification module, the sparse selection should be implemented on the radiometric calibration. The current implementation calibrates all the 120 spectral bands pixel-wise even when the SVMBDT is doing sparse mode.

# Bibliography


[1] Efficient learning and optimization tools for hyperspectral imaging systems (elo-hyp), `https://elohyp.wordpress.com/`. Accessed on 06/18-23.

[2] Savita Ahlawat and Amit Choudhary. Hybrid cnn-svm classifier for handwritten digit recognition. *Procedia Computer Science*, 167:2554–2560, 2020.

[3] Adrián Alcolea, Mercedes E Paoletti, Juan M Haut, Javier Resano, and Antonio Plaza. Inference in supervised spectral classifiers for on-board hyperspectral imaging: An overview. *Remote Sensing*, 12(3):534, 2020.

[4] Mohammad Azimi-Pour, Hamid Eskandari-Naddaf, and Amir Pakzad. Linear and nonlinear svm prediction for fresh properties and compressive strength of high volume fly ash self-compacting concrete. *Construction and Building Materials*, 230:117021, 2020.

[5] Sivert Bakken, Roger Birkeland, Joseph L Garrett, P Amund R Marton, Milica Orlandić, Evelyn Honoré-Livermore, Dennis D Langer, Cecilia Haskins, and Tor A Johansen. Testing of software-intensive hyperspectral imaging payload for the hypso-1 cubesat. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 258–264. IEEE, 2022.

[6] Sivert Bakken, Marie B Henriksen, Roger Birkeland, Dennis D Langer, Adriënne E Oudijk, Simen Berg, Yeshi Pursley, Joseph L Garrett, Fredrik Gran-Jansen, Evelyn Honoré-Livermore, et al. Hypso-1 cubesat: First images and in-orbit characterization. *Remote Sensing*, 15(3):755, 2023.

[7] Sivert Bakken, Evelyn Honoré-Livermore, Roger Birkeland, Milica Orlandić, Elizabeth F Prentice, Joseph L Garrett, Dennis D Langer, Cecilia Haskins, and Tor A Johansen. Software development and integration of a hyperspectral imaging payload for hypso-1. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 183–189. IEEE, 2022.

[8] AJ Batista-Leyva. Radiometry and photometry: Two visions of one phenomenon. *Revista Cubana de Física*, 36(1):66–72, 2019.

[9] Roger Birkeland, Gara Quintana-Diaz, Evelyn Honoré-Livermore, Torbjörn Ekman, Fernando Aguado Agelet, and Tor A Johansen. Development of a multi-purpose sdr payload for the hypso-2 satellite. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–11. IEEE, 2022.

[10] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[11] John W Chapman, David R Thompson, Mark C Helmlinger, Brian D Bue, Robert O Green, Michael L Eastwood, Sven Geier, Winston Olson-Duvall, and Sarah R Lundeen. Spectral and radiometric calibration of the next generation airborne visible infrared spectrometer (aviris-ng). *Remote Sensing*, 11(18):2129, 2019.

[12] Giacomo Curzi, Dario Modenini, and Paolo Tortora. Large constellations of small satellites: A survey of near future challenges and missions. *Aerospace*, 7(9):133, 2020.

[13] Magnus Danielsen. System integration and testing of on-board processing system for a hyperspectral imaging payload in a cubesat. Master's thesis, NTNU, 2020.

[14] Arthur Elmes, Hamed Alemohammad, Ryan Avery, Kelly Caylor, J Ronald Eastman, Lewis Fishgold, Mark A Friedl, Meha Jain, Divyani Kohli, Juan Carlos Laso Bayas, et al. Accounting for training data error in machine learning applied to earth observations. *Remote Sensing*, 12(6):1034, 2020.

[15] P Ezhil et al. Experimental analysis of optimization flags in gcc. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(7):1875–1879, 2021.

[16] JL Garrett, NK Singh, TA Johansen, and I Necoara. Accelerating support vector machines for remote platforms by increasing sparsity. In *2022 12th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5. IEEE, 2022.

[17] Joseph Garrett. Ssp-ls. `https://github.com/ELO-Hyp/SSP-LS/tree/new-diag`, 2023.

[18] Mariusz E Grøtte, Roger Birkeland, Evelyn Honoré-Livermore, Sivert Bakken, Joseph L Garrett, Elizabeth F Prentice, Fred Sigernes, Milica Orlandić, J Tommy Gravdahl, and Tor A Johansen. Ocean color hyperspectral remote sensing with high resolution and low latency—the hypso-1 cubesat mission. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–19, 2021.

[19] Xiao-li Hao and Huan Liang. A multi-class support vector machine real-time detection system for surface damage of conveyor belts based on visual saliency. *Measurement*, 146:125–132, 2019.

[20] Hui Huang, Xi'an Feng, Suying Zhou, Jionghui Jiang, Huiling Chen, Yuping Li, and Chengye Li. A new fruit fly optimization algorithm enhanced support vector machine for diagnosis of breast cancer based on high-level features. *BMC bioinformatics*, 20:1–14, 2019.

[21] Jingfu Li. Iot security analysis of bdt-svm multi-classification algorithm. *International Journal of Computers and Applications*, 45(2):170–179, 2023.

[22] Bing Lu, Phuong D Dao, Jiangui Liu, Yuhong He, and Jiali Shang. Recent advances of hyperspectral imaging technology and applications in agriculture. *Remote Sensing*, 12(16):2659, 2020.

[23] Wenjing Lv and Xiaofei Wang. Overview of hyperspectral image classification. *Journal of Sensors*, 2020, 2020.

[24] NanoAvionics. Innovative ocean research from ntnu to ride on the nanoavionics m6p nano-satellite bus, `https://nanoavionics.com/`, 2018.

[25] Feiping Nie, Wei Zhu, and Xuelong Li. Decision tree svm: An extension of linear svm for non-linear classification. *Neurocomputing*, 401:153–159, 2020.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Abdul Razaque, Mohamed Ben Haj Frej, Muder Almi'ani, Munif Alotaibi, and Bandar Alotaibi. Improved support vector machine enabled radial basis function and linear variants for remote sensing image classification. *Sensors*, 21(13):4431, 2021.

[28] Jonas Gjendem Røysland. Support vector machine with binary decision tree for classification onboard a satellite. *NTNU*, 2022.

[29] Jonas Gjendem Røysland. Svm_thesis_project. `https://github.com/jonasroy/SVM_THESIS_PROJECT`, 2022.

[30] Jonas Gjendem Røysland. onboard-pipeline-modules. `https://github.com/NTNU-SmallSat-Lab/onboard-pipeline-modules/tree/classification`, 2023.

[31] PM Salgado-Hernanz, M-F Racault, JS Font-Muñoz, and G Basterretxea. Trends in phytoplankton phenology in the mediterranean sea based on ocean-colour remote sensing. *Remote Sensing of Environment*, 221:50–64, 2019.

[32] Bijo Sebastian and Pinhas Ben-Tzvi. Support vector machine based real-time terrain estimation for tracked robots. *Mechatronics*, 62:102260, 2019.

[33] Signe Stroming, Molly Robertson, Bethany Mabee, Yusuke Kuwayama, and Blake Schaeffer. Quantifying the human health benefits of using satellite information to detect cyanobacterial harmful algal blooms and manage recreational advisories in us lakes. *GeoHealth*, 4(9):e2020GH000254, 2020.

[34] Min Wang, Ke-ming Yang, and Guo-ping Wang. Relative radiometric calibration of yaw data without calibration field of hyperspectral images based on harmonic analysis. *International Journal of Remote Sensing*, 41(14):5429–5442, 2020.

[35] Marie Weiss, Frédéric Jacob, and Grgory Duveiller. Remote sensing for agricultural applications: A meta-review. *Remote sensing of environment*, 236:111402, 2020.

[36] WHISPERS. Workshop for hyperspectral image and signal processing: Evolution in remote sensing, `https://www.ieee-whispers.com/`, 2023. Accessed on 06/18-23.

[37] Lei Yao, Zhanpeng Fang, Yanqiu Xiao, Junjian Hou, and Zhijun Fu. An intelligent fault diagnosis method for lithium battery systems based on grid search support vector machine. *Energy*, 214:118866, 2021.

# Appendix A

# Code

The implementation that is described in chapter 5 and used in chapter 6 is found in the attachments in the zip folder **Implementation/**.

# Appendix B

# WHISPERS paper

Proceeding paper submitted to WHISPERS conference 2023. The paper is named **HYPERSPECTRAL CLASSIFICATION ON-BOARD THE HYPSO-1 CUBESAT** and is found in the pages below.

# HYPERSPECTRAL CLASSIFICATION ON-BOARD THE HYPSO-1 CUBESAT

*Jonas G. Røysland*[1], *Dennis D. Langer*[2,4], *Simen Berg*[3],
*Milica Orlandić*[1,4], *Joseph L. Garrett*[3,4†]

[1]Department of Electronic Systems
[2]Department of Marine Technology
[3]Department of Engineering Cybernetics
[4]Center for Autonomous Marine Operations and Systems
Norwegian University of Science and Technology
O.S. Bragstads Plass 7034 Trondheim, Norway

## ABSTRACT

***Index Terms*—** Machine Learning, Real-time Classification, Remote Sensing, Support Vector Machines, Embedded Systems

## 1. INTRODUCTION

The vast amounts of data created by hyperspectral payloads and the limited satellite downlink bandwidth together encourage the use of onboard processing for hyperspectral remote sensing missions. Because tasks such as compression and sensor corrections offer the largest immediate gain, passing more data through the same bandwidth, they were the first to be run onboard.

In recent years the popularity of small satellites has exponentially increased after the launch cost has been lowered. This has opened up more possibilities for researching the Earth's surface with satellites. This is called remote sensing and is about monitoring the characteristic of an area by capturing the reflected solar light from the surface. This is visualized in **Fig. 1**.

One of the satellite projects doing this is HYPSO at NTNU in Trondheim. On the 13th of January 2022, their first satellite HYPSO-1 with a hyperspectral imager was launched into Low Earth Orbit (LEO) [2]. The data it produces are Hyperspectral Images (HSI) which are images that contain high amounts of spectral bands (120 on HYPSO-1). The benefit of HSI is that have more information in the electromagnetic spectrum, which makes it more capable to identify objects and materials on surfaces. One of the main goals of HYPSO with HSI is to analyze and monitor harmful algae blooms (HABs) in waters, that could harm the local ecosystem.

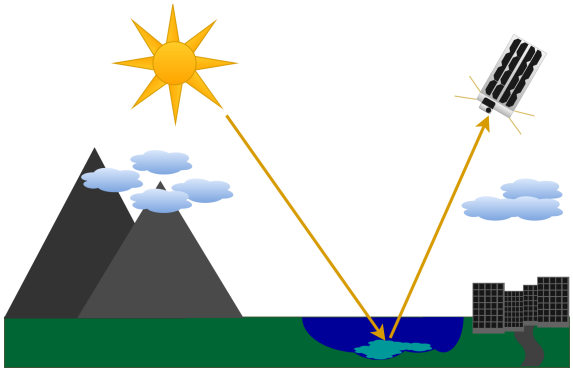


**Fig. 1**: Illustration of remote sensing where solar light is reflected on the Earth's surface and hitting an imager on a satellite [1].

Due to the limited bandwidth between the satel-

The research leading to these results has received funding from the NO Grants 2014 – 2021, under Project ELO-Hyp, contract no. 24/2020.

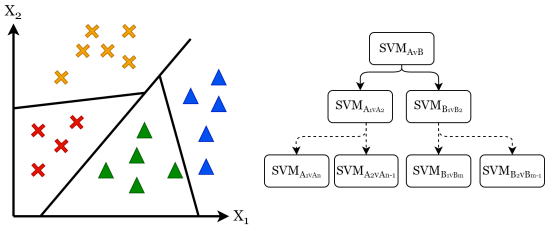
†joseph.garrett@ntnu.no

lite and the ground station, and the size of HSI (151MB)[3], it takes at least 90 minutes (2 orbital passes) to downlink one HSI [2]. This is less likely to make able to work with other real-time systems as Auntomus drones to have a local analysis of an area. One solution to make the information of the HSI a lot smaller is classified labels that are produced onboard the satellite. Example of this is illustrated in **Fig. 3** . The labels will then contain classes (for example algae blooms) that say what types of surfaces are located on the pixels. With image dimensions of 956x684, the labels would have a size of 327kB with a max of 16 classes.

To make the prediction when labeling onboard, Support Vector Machine (SVM) with Binary Decision Tree (BDT) has been implemented. SVM is a supervised learning model that does binary separation. The least computational separation in SVMs is linear kernels that are mathematically shown as:

$$\sum_{i=1}^{n} w_i \cdot x_i + b = \pm 1 \tag{1}$$

,where $x_i$ input data, $w_i$ weight, $b$ intercept and $n$ number of features. The separation is visualized in **Fig. 2** (a).




(a) Feature space (b) Binary Decision Tree

**Fig. 2**: (a) SVM separating two classes and then two subclasses in feature space with the BDT. (b) General BDT structure where each node has an SVM model.

The BDT is the SVMs decision function that makes it possible to do multiclass predictions. The BDT makes it possible to design which of the classes should be separated from each other, and where the subclasses should go. This makes it robust when doing multiclassification, and the majority of the errors are done in the subclasses [4]. A general tree structure is shown in **Fig. 2** (b). The time complexity of SVMs is $\mathcal{O}(n)$ [5], making it more predictable how the execution time will be.

Due to the high amount of spectral bands ($n$ features) and the execution are proportional to it. To reduce the $n$, Sparse SVM is utilized for the band selection. Making only the most significant spectral bands chosen [6]. This then will affect the number of calculations in Equation 1.

When the HSI is captured, it is taken place in different areas of the Earth and can have a different exposure value. This could affect the prediction for the SVMBDT making more errors due to this difference. To solve this problem, radiometric calibration has been implemented. This makes the digital values in the images into physical values. The first part of the calibration is done by looking at the scattering of the solar lights, and it's called radiance. It's mathematically expressed as:

$$L(\lambda, x) = [DN(\lambda, x) - \alpha(\lambda, x)]\beta(\lambda, x), \tag{2}$$

where $DN(\lambda, x)$ are the digital numbers in the image, $L(\lambda, x)$ is the function of wavelenght $\lambda$ and x is the cross-track spatial location. $\alpha(\lambda, x)$ are the offset and $\beta(\lambda, x)$ are the gain in the $DN$s.

The second part of the calibration is called reflectance. This factor in the distance from the Sun and Earth and the solar zenith angle of the reflected solar light. The mathematical expression is shown:

$$R_{sensor} = \frac{L(\lambda, x) \times \pi \times d^2}{E_{SUN} \times cos\theta_S} \tag{3}$$

$R_{sensor}$ is the reflectance, $d$ is the distance between the Earth and the Sun. $E_{SUN}$ is the mean solar exo-atmospheric irradiance, and the $\theta_S$ is the Solar zenith angle [2].

## 2. LABELING, TRAINING, AND TESTING

To make the concept described above three parts are implementation. These are ground training, onboard classification and decode labels. The ground training loads training data and trains models to

make the SVM model weights $w_i$ and intercepts $b$, and the sparse selection. These and the radiometric coefficients are stored in a config folder that is stored onboard the satellite. The onboard classification loads then targeted the HSI cube and the config data when it is executed. The ground station has the choice to make decisions the cube should be radiometrically calibrated with radiance or reflectance. When the cube is calibrated the pixels are then predicted with the SVMBDT with the choice of using all the bands or the sparse selection. The predicted labels are then stored as a compressed labeled image. When the satellite is in the horizon of the ground station the labels can be downlinked and decoded by the decode labels part. An illustration of how all three parts a linked together are shown in Figure 3.

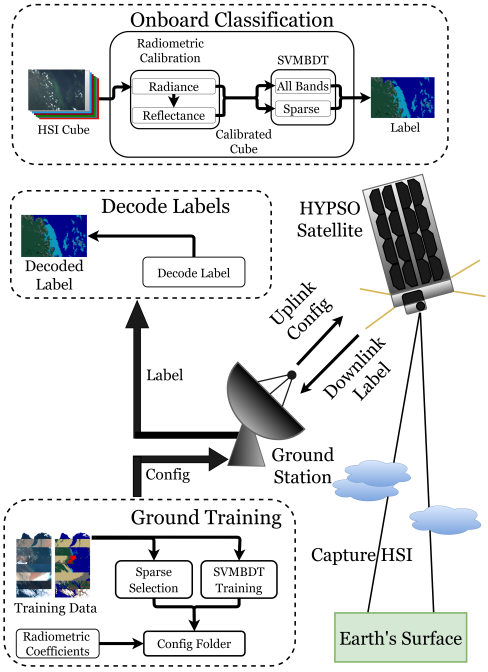
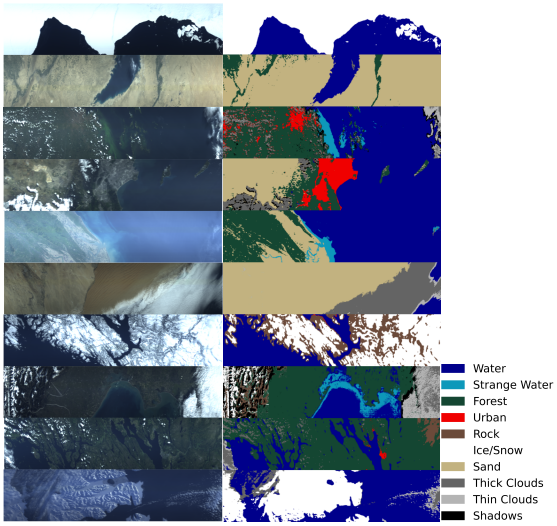



**Fig. 3**: Illustration of how the ground training, onboard classification, and decode labels are working together [1].

For making the config data that are necessary for the onboard classification to the predictions, the HSI training sets are made with labeled ground truth. The training data with legend are shown in **Fig. 4**.




(a) HSI Captures (b) Labels (c) Legend

**Fig. 4**: Labeled HSI training set with (a) HSI captures in RGB representation and (b) the associated labels of the HSI captures.

The implementation that is made is tested on a personal computer to verify the accuracy and the different prediction times with different processing modes. Since the same model should predict all the HSI around the Earth, a BDT structure is designed to aim for separating cloud/ice/sand, water, and land surface. These have different subclasses where water has strange water, clouds have thick and thin clouds, and the land mass is forest, urban, and rock. The shadow class is the shadows produced by clouds and terrain. The strange water contains algae bloom, pollution, and shallow water. The BDT constructed for this is shown in **Fig. 5**.

The implementation is trained with 10% of the training data and tested on the computer. The results from this with four different processing modes are shown in **Fig. 6**.

The overall processing method with the best accuracy is reflectance but also has the longest classification time. The best trade of accuracy and also having the best classification time are Radiance Sparse.

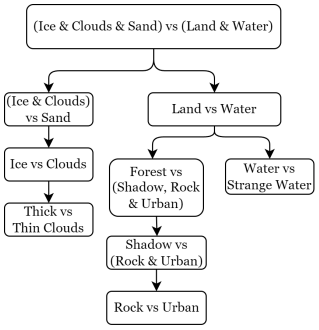



**Fig. 5**: BDT 1 Structure

| Area | Radiance | | Reflectance | |
|---|---|---|---|---|
| | All | Sparse | All | Sparse |
| **Antarctica [%]** | 99.01 | 97.93 | 99.08 | 60.52 |
| **Tharthar [%]** | 94.74 | 93.92 | 96.89 | 0.04 |
| **Kampala [%]** | 75.43 | 83.67 | 80.61 | 3.45 |
| **Los Angeles [%]** | 84.12 | 79.31 | 80.08 | 1.4 |
| **Sittwe [%]** | 91.65 | 88.96 | 84.39 | 4.77 |
| **Gobabeb [%]** | 93.39 | 88.66 | 96.83 | 0.59 |
| **Trondheim [%]** | 82.46 | 88.06 | 90.1 | 24.09 |
| **Trondheim, Snow [%]** | 80.55 | 83.33 | 84.43 | 48.83 |
| **Venice [%]** | 62.84 | 70.2 | 63.73 | 17.23 |
| **Svalbard [%]** | 75.34 | 74.83 | 75.76 | 35.26 |
| **Mean accuracy [%]** | 83.95 | 84.89 | 85.19 | 19.62 |
| **Loading time [ms]** | 40.692 | 40.64 | 43.85 | 43.68 |
| **Classification time [s]** | 0.27 | 0.18 | 0.30 | 0.218 |

**Fig. 6**: Results from computer test where the 10 training HSI are tested by accuracy, loading time, and classification time.

Due to this, the processing method is the choice that would be run when doing the in-flight test.

Before the onboard classification is tested onboard the HYPSO-1 satellite, the implementation needs to be tested on target hardware. On NTNU Smallsatlab there is a setup called LidSat, where the goal is to test software on satellite target hardware. The test is performed with mean loading and classification time shown as:

| Execution time | |
|---|---|
| **Loading Config Data** | **Classification** |
| 0.49s | 8.46s |

**Fig. 7**: LidSat mean loading and classification execution times when predicting HSI.

## 3. IN-FLIGHT TESTS

Now the implementation has been tested on the target hardware it is ready to be tested onboard the satellite. The onboard classification build together with the config folder is uplinked to the HYPSO-1 satellite via the ground station. To test the implementation some HSI captures that are stored on the satellite are tested. The results of the in-flight onboard labels are shown in **Fig. 8**, after they are decoded on the ground.

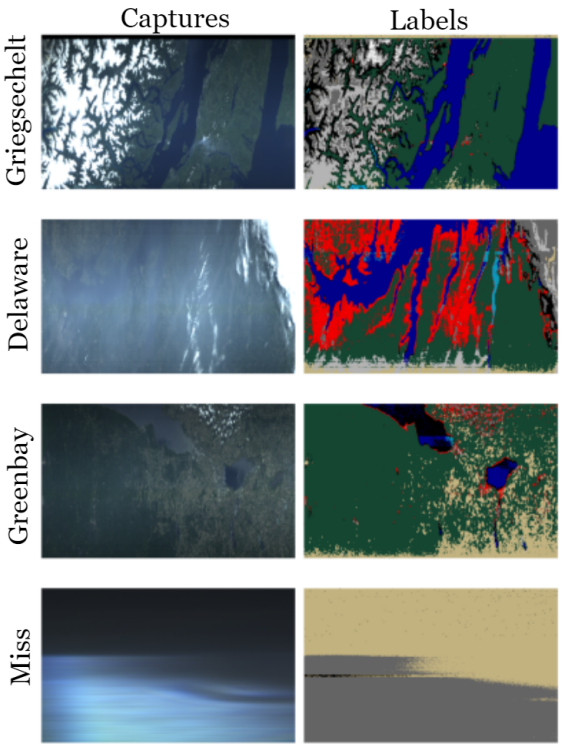



**Fig. 8**: One of the in-flight test results where to the left are the captures and to the right labels.

The figure above shows that the labels have the same shapes as the original HSI captures. The labels are robust when labeling between land, cloud, and water. The errors in the labeled image are for example ice/snow classified as clouds, from the Griegsechelt labels. From the Miss (Pointing Error) Capture, it can be viewed that the capture wasn't successful to aim at the Earth's surface. The bottom pixel is clouds and the top is space. Due to the SVMBDT model doesn't have space labels from the training data, the space pixels are predicted as sand. From the four labels that have been downlinked, it can successfully say that the labels can give a good estimate if the HSI captures are good enough to be downlinked or not.

Further, the onboard classification will be labeling every HSI captures that is taken. The labels will then be downlinked before the whole HSI captures so that the ground station can determine if the capture will be downlinked or not.

| **HSI Capture** | **Classification Time [s]** |
|---|---|
| Griegsechelt, Canada | 8.25 |
| Delaware, USA | 8.45 |
| Greenbay, USA | 8.04 |
| Miss (Pointing Error) | 9.06 |

**Fig. 9**: HYPSO-1 classification table.

## 4. CONCLUSION

This paper has explored the design of the onboard classification on the HYPSO-1 satellite for labeling HSI captures. For the prediction SVMBDT is chosen for its low execution time, but overall robustness. Since the captures are taken over all of the Earth with different exposure in the images, radiometric calibration is utilized before doing the prediction. The SVMBDT is trained with 10 HSI training labels and is tested for overall accuracy and execution when doing the test. The processing mode chosen to run on the satellite is Radiance Sparse with the lowest classification time with an 85% overall accuracy. The onboard classification is tested on target hardware, and then later uplinked to the HYPSO-1 satellite. The implementation is working as intended and the labeling gives a good representation of the original HSI captures. The ground station then has a labeled image to determine if the HSI capture should be downlinked or not.

## 5. REFERENCES


[1] Jonas Gjendem Røysland, *Real-time classification onboard the HYPSO-1 satellite*, Norwegian University of Science and Technology, 2023.

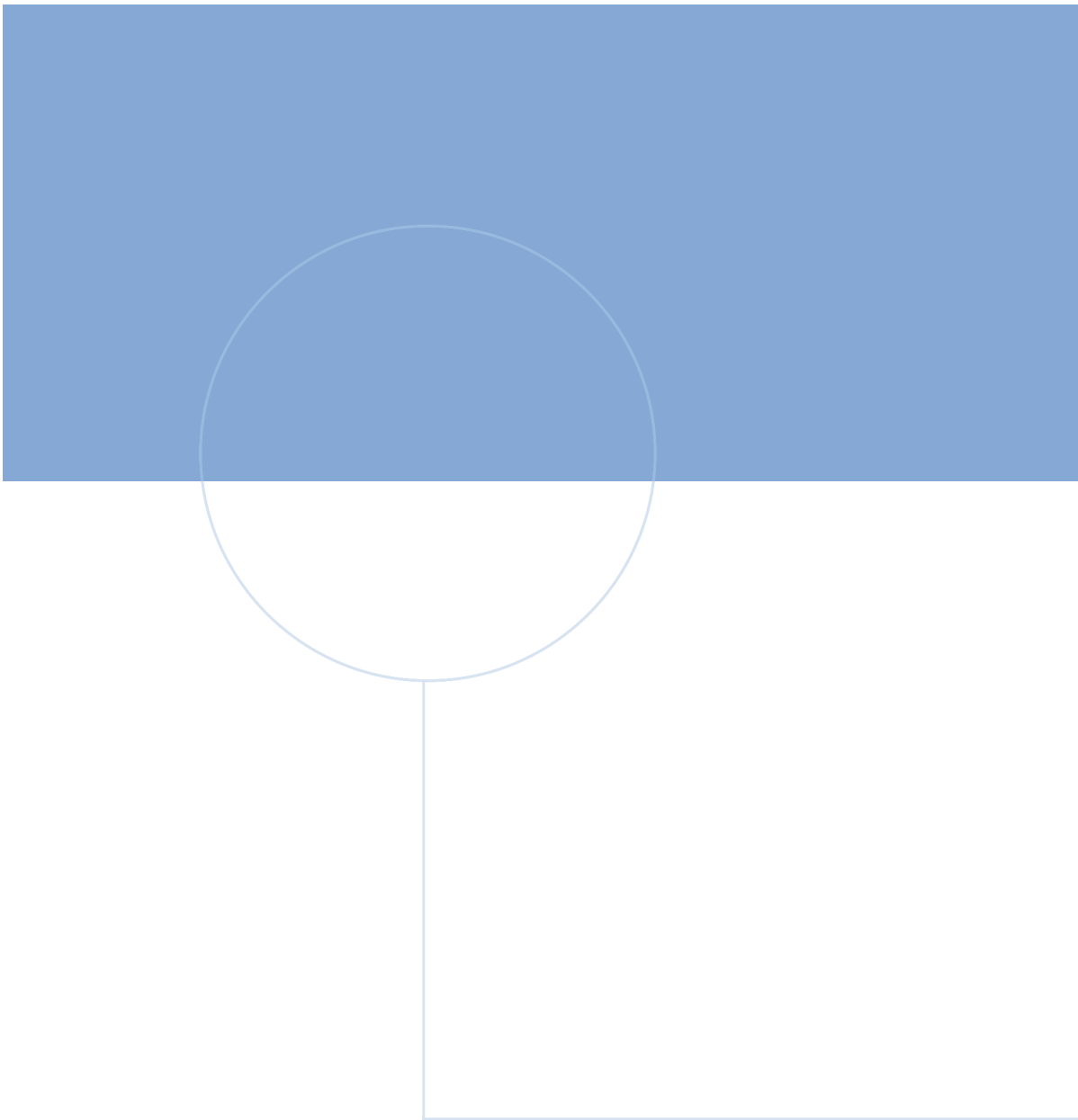
[2] Sivert Bakken, Marie B Henriksen, Roger Birkeland, Dennis D Langer, Adriënne E Oudijk, Simen Berg, Yeshi Pursley, Joseph L Garrett, Fredrik Gran-Jansen, Evelyn Honoré-Livermore, et al., "Hypso-1 cubesat: First images and in-orbit characterization," *Remote Sensing*, vol. 15, no. 3, pp. 755, 2023.

[3] Sivert Bakken, Evelyn Honoré-Livermore, Roger Birkeland, Milica Orlandić, Elizabeth F Prentice, Joseph L Garrett, Dennis D Langer, Cecilia Haskins, and Tor A Johansen, "Software development and integration of a hyperspectral imaging payload for hypso-1," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 183–189.

[4] Jingfu Li, "Iot security analysis of bdt-svm multi-classification algorithm," *International Journal of Computers and Applications*, vol. 45, no. 2, pp. 170–179, 2023.

[5] Abdul Razaque, Mohamed Ben Haj Frej, Muder Almi'ani, Munif Alotaibi, and Bandar Alotaibi, "Improved support vector machine enabled radial basis function and linear variants for remote sensing image classification," *Sensors*, vol. 21, no. 13, pp. 4431, 2021.

[6] JL Garrett, NK Singh, TA Johansen, and I Necoara, "Accelerating support vector machines for remote platforms by increasing sparsity," in *2022 12th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, 2022, pp. 1–5.



NTNU

Norwegian University of
Science and Technology